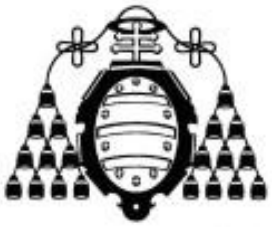
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE MASTER

"SERVIDOR MULTIMEDIA CON CONTROL REMOTO ANDROID"

DIRECTORA: María del Puerto Paule Ruíz

AUTOR: Jesús Sánchez Sanzo

Agradecimientos

En primer lugar he de agradecer este proyecto a mi directora de proyecto María del Puerto, sin cuyo apoyo y orientación no habría sido posible, y que también me ha puesto en contacto con la empresa en cuyo entorno se ha desarrollado el proyecto.

En segundo lugar agradecer a mis amigos, familiares y compañeros del gremio, a los que tantas dudas he preguntado durante el proceso de desarrollo y documentación sin dejar de animarme a acabarlo.

Agradecer a mi amigo Adair el ser la fuente de la idea del proyecto, y esperar que pueda llegar a serle útil la solución desarrollada.

Parte del código y plugins que han sido utilizados y adaptados para desarrollar el proyecto, han sido creados originalmente por otros desarrolladores como Muhammad Ichsan o Sunit Katkar a los que no conozco en persona pero quiero agradecer su labor.

Cómo no, agradecer a Redondo, autor de la plantilla de Documentación, sin la cual la organización de la información en un buen documento habría llevado diez veces más tiempo y esfuerzo.

Agradecer a Alberto y Maxi, mis compañeros en la empresa en la que he realizado las prácticas y el proyecto, que me han orientado y compartido su experiencia conmigo para resolver las dudas de implementación y orientar el desarrollo.

He de agradecer a Macamen la ayuda y orientación en el tramo final del desarrollo de la interfaz de usuario señalando los defectos de usabilidad a corregir y aspectos a mejorar, consiguiendo mejorar enormemente la usabilidad y adecuación de la misma.

Por último agradecer a mis padres que nunca han escatimado en tiempo, dinero ni dedicación en mi formación.

Gracias a todos.

Resumen

El proyecto consiste en un sistema informático que implementa un servidor web en el que los usuarios pueden guardar nombres y ubicaciones de sus recursos multimedia favoritos disponibles en la red, así como ordenarlos y organizarlos en listas de reproducción.

Estos medios y listas podrán ser reproducidos desde la propia página web del sitio, y la reproducción se podrá controlar remotamente utilizando una aplicación móvil desarrollada a tal efecto; es decir, el usuario reproduce videos en el navegador de su ordenador y los controla desde el teléfono móvil.

El sistema no guarda los recursos multimedia en sí, sino sólo sus ubicaciones lo que permite al usuario utilizar cualquier recurso disponible públicamente en internet sin tener que previamente cargarlo en nuestro servidor.

Cuando el usuario quiere crear y guardar un nuevo medio, se le facilita un sistema de búsqueda o sugerencia que encuentra recursos ya disponibles en la plataforma Youtube y rellena automáticamente el formulario de creación.

Como la reproducción es controlada remotamente desde el dispositivo móvil, el navegador web ha de reflejar respuestas a órdenes que se han enviado desde otro dispositivo, el modelo de comunicación difiere de la típica petición-respuesta por parte del navegador, lo que requiere una manera de iniciar una comunicación por parte del servidor hacia el cliente web. Para esto se han utilizado WebSockets, que permiten mantener una comunicación síncrona con el cliente web, y actualizar el estado de reproducción según las órdenes enviadas por el usuario desde su aplicación móvil.

Una ventaja de este modelo es la posibilidad de controlar desde un mismo dispositivo, todos los reproductores que el usuario haya iniciado con su cuenta en distintos ordenadores. Se podría por ejemplo, controlar y sincronizar la reproducción de música de toda una cadena de tiendas situadas en distintos puntos geográficos, desde un solo dispositivo móvil.

Además el sistema incorpora gestión de usuarios, con registro, control anti creación automática, recuperación y cambio de contraseñas, etc.

Tanto el entorno web como la aplicación móvil están internacionalizados siendo los idiomas disponibles el español y el inglés.

Palabras Clave

Servidor multimedia, Control remoto, Websockets, Spring, GV-NIX, Android, Java

Abstract

This project consists of a computer system that implements a web server where users can store names and locations of their Favorite multimedia resources available on the Internet as well as sort and organize them into playlists.

These multimedia resources and lists may be played from the web page of the site itself, and playback can be controlled remotely using a mobile application developed for this purpose; that means the user plays videos in the web browser of his computer and controls them from remotely using his mobile phone.

The system does not save multimedia resources themselves, just their locations allowing the user to use any resource publicly available on the internet without having to first upload it to our server.

When the user wants to create and save a new media, he has the option to use the suggestion tool to find resources on the Youtube platform and automatically fill in the form of creation to add them to the system.

As the playback is remotely controlled from the mobile device, the web browser needs to react to some responses to orders that have not been sent from the browser itself. This means the communication model differs from the typical request-response from the browser, which requires a way to start a communication from the server to the web client. WebSockets have been used for this propose, because this technology enables to maintain a synchronous communication between the server and the Web client, and update the status of playback according to commands sent by the user from his mobile application.

An advantage of this model is the option to control using a single device, all players that the user has logged in with his account on different computers. It could, for example, control and synchronize music playback for a chain of stores located in different geographical locations from a single mobile device.

In addition the system includes user management, with signup, anti-bot creation control, password change and recovery, etc.

Both the web environment and the mobile application are internationalized being languages available Spanish and English.

Keywords

Multimedia server, Remote control, Websockets, Spring, GV-NIX, Android, Java

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

En esta memora se hace un resumen del proyecto dirigido a cualquier público, para tratar de hacer sumario de los aspectos más importantes del mismo y que se pueda entender de qué trata el proyecto

En el presente proyecto se pretende desarrollar un sistema informático que implemente un entorno web que permita a sus usuarios guardar la ubicación de sus videos y audios favoritos, así como controlar a través de su dispositivo móvil lo que se reproduce en el navegador del ordenador.

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La reproducción de contenidos multimedia es una actividad enormemente extendida tanto en el ámbito cotidiano como profesional.

Ver videos o escuchar música es algo que hacemos constantemente en nuestro tiempo de ocio, pero estos contenidos multimedia también son usados en las tiendas, bares, peluquerías, como música de fondo y aunque su negocio no tenga una relación directa con ellos, sí que tienen influencia a la hora de infundir determinados estados de ánimo o sensaciones en sus clientes, como puede ser la tranquilidad con música lenta y calmada, o la agitación y prisa con la música más movida que suele sonar en las tiendas cuando se acerca la hora del cierre.

El uso de internet, hace que el consumo de recursos multimedia ya no necesite que sus usuarios lleven consigo su música o videos, sino que con una conexión a internet el usuario pueda disponer de ellos en cualquier lugar.

La idea del proyecto surge cuando un amigo que ya utiliza este tipo de servicios de música online mientras atiende a los clientes de la tienda en la que trabaja, me cuenta sus dificultades con este tipo de programas. Por una parte la necesidad de instalar nuevo software en los ordenadores de la tienda, o acceder a sitios web que puedan estar filtrados, y por otra la necesidad de acercarse al ordenador desde el que se controla la reproducción de la música.

Esto nos hace pensar, ¿No sería estupendo disponer de un sitio donde poder guardar nuestros recursos multimedia, organizarlos y reproducirlos a nuestro gusto, y controlar la reproducción cómodamente a distancia a través de nuestro teléfono móvil? Ese es pues el objetivo del proyecto que describe este documento.

Con él se podría no sólo acceder desde cualquier sitio a una inmensa cantidad de recursos multimedia, sino controlar la música de un establecimiento desde un dispositivo móvil sin tener que dejar desatendidos a los clientes ni un segundo, o en el caso del ámbito doméstico reproducir contenidos cómodamente utilizando nuestro dispositivo móvil como mando a distancia.

Además este modelo de control permitiría no sólo controlar un reproductor, sino cualquier número de ellos simultáneamente, desde el mismo dispositivo, sincronizando la reproducción por ejemplo, de todos los ordenadores de una tienda, o todas las tiendas de una cadena, por muy distantes que éstas estén situadas.

1.2 Resumen de Todos los Aspectos

A continuación se hace un pequeño resumen de todos los aspectos que se verán en este documento:

- **Introducción**: Aquí se justifica la motivación del desarrollo del proyecto, así como sus objetivos, situación actual, y alternativas o soluciones actuales a los problemas que el proyecto pretende resolver.

- **Aspectos teóricos:** En esta sección se describirán brevemente aquellos conceptos, herramientas y tecnologías más importantes usados y/o relacionados con el proyecto**.**

- **Planificación del proyecto**: En esta sección se describe el plazo de ejecución del proyecto y de qué forma se repartirá el tiempo y el trabajo para llevarlo a término.

- **Análisis**: Este apartado contendrá toda la especificación de requisitos y la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

- **Diseño del sistema**: En este apartado se detallará el diseño del sistema construido a partir del análisis realizado en el apartado anterior. Cómo se estructurará el código, la base de datos, la interfaz, así como qué pruebas se realizarán para comprobar que el programa funcione correctamente.

- **Implementación del sistema**: En esta sección se describe como se ha llevado a cabo la implementación del proyecto.

- **Desarrollo de las pruebas**: aquí se muestran los resultados de las diferentes pruebas que hemos realizado al proyecto para comprobar su adecuación con los objetivos.

- **Manuales del sistema**: Diferentes manuales para el los usuarios del sistema, tanto instalación y despliegue de la aplicación, como uso, administración y desarrollo.

- **Conclusiones y ampliaciones**: Conclusiones del sistema qué hemos elaborado, del proceso de aprendizaje durante su elaboración y posibles ampliaciones a realizar en el futuro sobre el mismo.

- **Presupuesto**: Presupuesto del coste de desarrollo del proyecto. En esta sección se hace un cálculo de presupuesto en el caso de que el cliente fuera un cliente real y no se tratara de una aplicación hecha para el ámbito académico.

- **Referencias bibliográficas**: Libros, artículos y sitios de internet usados de alguna forma durante el desarrollo del proyecto y su documentación.

- **Apéndices**: Aquí se incluyen un glosario, listado del contenido entregable junto a este documento, javadoc y el código fuente de la aplicación.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

La reproducción de música y videos es una actividad muy habitual tanto a nivel cotidiano como profesional, en tiendas, locales de ocio, etc. Cada vez cobra más importancia el consumo de estos medios procedentes de la red, pues con una simple conexión a internet, se tiene acceso a una colección de recursos incomparablemente más amplia de la que cualquiera podría guardar en un medio de almacenamiento convencional.

No obstante la ubicación de los recursos en la red es muy diversa, repartida en servidores de todo el mundo, por lo que ofrecer un medio de guardar, recopilar todas estas ubicaciones podría resultar útil.

Si además se ofrece la opción de reproducirlos en el mismo sitio web y controlar remotamente la reproducción, se pone a disposición de los usuarios la libertad de tener sus recursos disponibles y localizados en cualquier parte donde haya una conexión a internet.

Un ejemplo ilustraría bien la situación:

Consideremos el caso de un dependiente que trabaja en una cadena de tiendas de calzado, y que debe controlar la música de fondo del establecimiento, pero estar siempre disponible para atender a sus clientes.

Con el sistema desarrollado en este proyecto puede controlar la reproducción de música desde cualquier parte de la tienda mediante su dispositivo móvil, sin tener que acercarse hasta el ordenador desde el cual se reproduce, por lo cual no pierde ningún tiempo de estar disponible para sus clientes.

Si otro día ese mismo dependiente ha de trabajar en otra tienda de la cadena, no necesita llevarse consigo la música para reproducir, sino que con solo iniciar sesión en el sitio web la aplicación tendrá disponibles todos sus recursos igual que en la primera ubicación.

Si por cualquier motivo fuera responsable de la música de todas las tiendas de la cadena, no tendría más que iniciar su sesión en la aplicación en todos los ordenadores que reproducen la música, y podría controlar y sincronizar todos, remotamente, sin restricciones de distancia.

2.2 Objetivos del Proyecto

El objetivo principal de este proyecto es el desarrollo de un sistema informático que permita a los usuarios recopilar y organizar recursos multimedia disponibles en la red para luego reproducirlos. Esta reproducción podrá ser controlada remotamente por mediante dispositivos móviles.

Objetivos detallados:

1. Facilitar y acomodar la reproducción de videos y audios disponibles en la red.
2. Permitir el control de remoto de un reproductor multimedia sin necesidad de emparejamientos previos entre los dispositivos implicados.
3. Permitir la sincronización de varios reproductores con independencia de su ubicación.
4. Contemplar el despliegue distribuido de la aplicación en varios servidores.
5. Tener en cuenta aspectos de usabilidad de cara a los usuarios.
6. Sencillez en sus funciones y facilidad en su manejo.
7. Contemplar aspectos de seguridad.

Otros objetivos:

- Formación y mejora en el manejo de frameworks de desarrollo utilizados en la actualidad como Spring, y frameworks de desarrollo rápido de aplicaciones como Spring Roo.
- Aprendizaje de tecnologías de comunicación y sincronización actuales como WebSockets.
- Desarrollo de habilidades relacionadas con programación para dispositivos móviles y formación en tecnologías relacionadas tales como Android y phonegap.
- Madurez en el rendimiento y calidad del trabajo realizado.

2.3 Estudio de la Situación Actual

Aunque en la estudio realizado no se han encontrado soluciones que enfoquen la funcionalidad exactamente de la misma manera que la aplicación objetivo del proyecto, a continuación se presentan distintas alternativas que ofrecen funcionalidades relacionadas con el mismo, que si están disponibles actualmente.

De hecho una de las alternativas, concretamente VLC Player está muy relacionada con el proyecto, ya que se hace uno de su plugin web para la reproducción de los elementos multimedia, haciendo uso de su API en javascript desde el cliente web.

2.3.1 Evaluación de Alternativas

Aquí se presentan las alternativas estudiadas tratando de describir su funcionamiento, ventajas e inconvenientes.

2.3.1.1 Sistema 1: Youtube TV

El sitio web de reproducción de videos más utilizado en la actualidad, pone a disposición de sus usuarios a través www.youtube.com/tv#/ un reproductor de videos, cuyo control remoto es posible emparejar con un dispositivo móvil utilizando en éste la aplicación homónima.

Esta funcionalidad fue pensada para poder utilizar las nuevas Smart TVs capaces de acceder a internet para hacer el papel de reproductor de videos de Youtube, y controlar esta reproducción mediante en dispositivo móvil con la aplicación correspondiente instalada, y previo emparejamiento de ambos dispositivos.

La funcionalidad de control remoto viene ya incluida con la aplicación de reproducción de videos que YouTube ofrece.

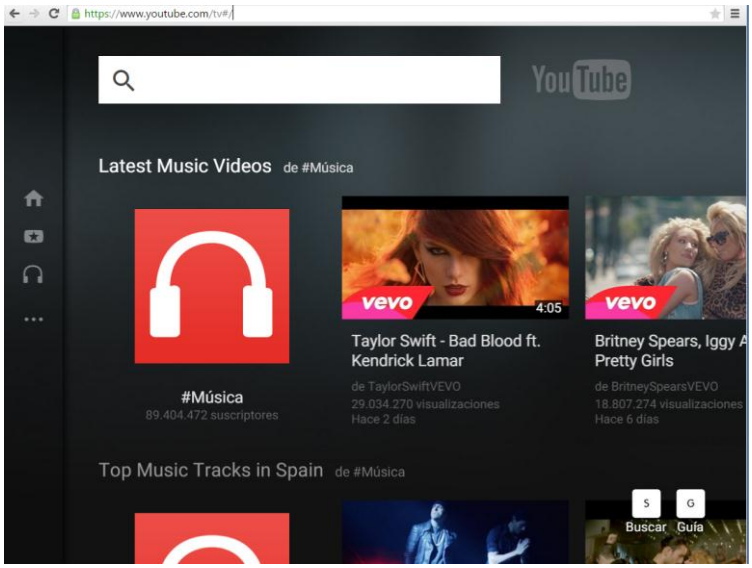


Figura 2.1. Vista del de la interfaz web de YouTube TV

Figura 2.2. Vista del reproductor web YouTube TV

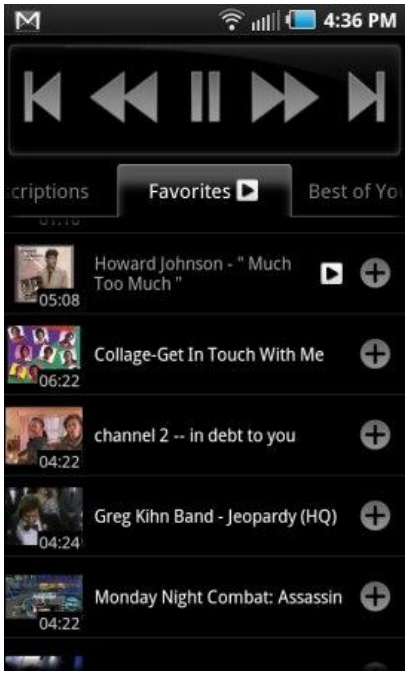


Figura 2.3. Vista del interfaz de control remoto móvil de YouTube

Como ventajas se podrían señalar:

- Es de manejo intuitivo.
- Gratuito.
- Muy extendido.
- Soporte multiplataforma y multidispositivo.

Entre sus inconvenientes:

- Requiere un proceso de emparejamiento entre el dispositivo reproductor y el control remoto.
- No permite controlar más de un dispositivo simultáneamente, ni más de un control remoto.
- Los recursos disponibles para reproducir han de estar alojados únicamente en los servidores de Youtube.

También conviene señalar que este sistema ofrece una enorme cantidad de opciones por lo que puede desorientar inicialmente a los usuarios.

2.3.1.2 Sistema 2: VLC Remote

El conocido reproductor multimedia multiplataforma famoso por la gran variedad de formatos que acepta dispone también de una aplicación de control remoto desde dispositivos móviles.

No obstante este no sería un reproductor web, sino una versión de escritorio, por lo que aunque sí que podría reproducir cualquier topo de archivo, local o remoto, la información acerca de listas de reproducción, etc. dependerán de cada máquina particular, y habría que trasportarlas de manera independiente si se quisiera reproducir la misma lista de reproducción en otro ordenador.

Figura 2.4. Vistas del interfaz de control remoto móvil de VLC Remote

Como ventajas se podrían señalar:

- Es de sencillo manejo, la interfaz móvil es muy clara y sencilla.
- Gratuito.
- Gran cantidad de formatos distintos aceptados.
- Fuentes locales y remotas.

Entre sus inconvenientes:

- Requiere un proceso de emparejamiento entre el programa reproductor y la aplicación control remoto.

- No permite controlar más de un dispositivo simultáneamente, ni más de un control remoto.
- Es una aplicación de escritorio que ha de estar instalada en cada ordenador que se quiera controlar como reproductor.
- Las listas de reproducción y recursos están ligados a la máquina en la que está instalado el reproductor.

2.3.1.3 Sistema 3: Pogoplug PC

Es una utilidad para Windows que pone los medios almacenados en el ordenador disponibles para otros PCs o dispositivos móviles. Se eligen los directorios a compartir y se accede a ellos a través del navegador o los programas específicos para cada dispositivo.

La finalidad de la aplicación es más bien poner a disposición de otros dispositivos los recursos almacenados en la máquina servidora, que el de ofrecer una reproducción remota. Esta se podría conseguir conectando un dispositivo reproductor que acceda a los recursos multimedia y controlando éste a su vez remotamente.

El programa tiene un coste de 29.95 $ e incluye 5 Gb de almacenamiento en la nube.

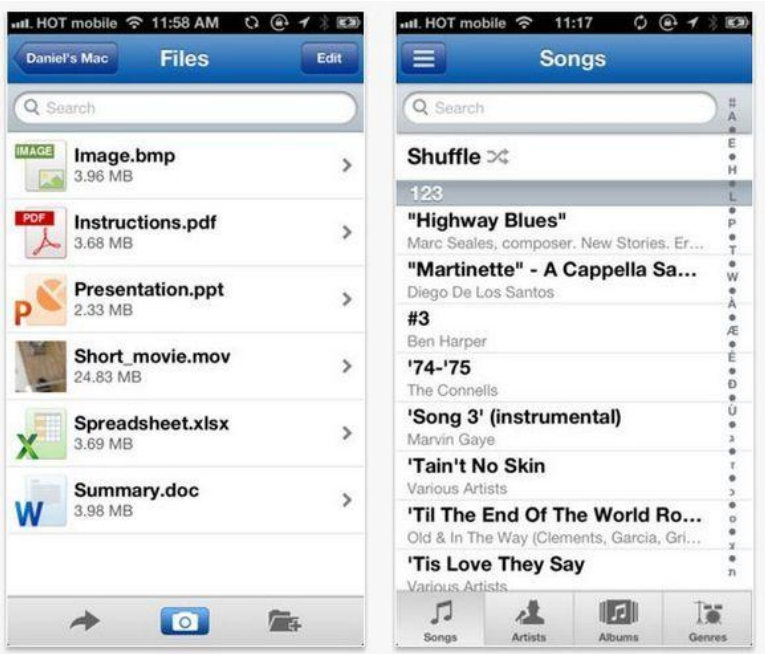


Figura 2.5. Vistas del interfaz de control remoto Pogoplug

Como ventajas se podrían señalar:

- Es de sencillo manejo.
- Gran cantidad de formatos.

Entre sus inconvenientes:

- No está pensado para la reproducción remota de multimedia, necesitando equipo y configuración adicionales.
- No es gratuito.
- Acciones limitadas.
- Solo disponible para Windows.

2.3.1.4 Sistema4: Microsoft Remote Desktop

Se trata de una aplicación de escritorio remoto completo para sistemas Windows, con ella se puede controlar completamente el equipo objetivo desde el dispositivo móvil como si de su versión de escritorio se tratase.

Así pues se teniendo un reproductor multimedia instalado en el ordenador objetivo, se podría controlar la reproducción, aunque la facilidad de uso se vería comprometida debido a un tamaño muy pequeño de los botones y controles, que se replicarían a escala en la pantalla del dispositivo móvil, sensiblemente más pequeño que una pantalla de ordenador standard.

Como ventajas se podrían señalar:

- Manejo ya conocido por ser idéntico a un escritorio ya utilizado por el usuario
- Gratuito.
- Acciones y configuraciones tan amplias como una interfaz de escritorio.
- Formatos aceptados dependiente del reproductor instalado en el PC.
- Fuentes locales y remotas dependiendo del reproductor instalado.

Entre sus inconvenientes:

- Solo aplicable a sistemas Windows que tengan habilitada la opción de control remoto.
- No permite controlar más de un dispositivo simultáneamente, ni más de un control remoto.
- Dificultad de uso por la pequeña escala de la interfaz.
- Los medios y listas de reproducción están ligados a la máquina que se controle.

Figura 2.6. Vistas del interfaz de control remoto móvil de Microsoft Remote Desktop

2.3.2 Resumen de alternativas

Característica	Youtube TV	VLC Remote	Pogoplug PC	Microsoft Remote Desktop
Fácil manejo	✓	✓	✓	✗
Gratuito	✓	✓	✗	✓
Gran cantidad de formatos	✗	✓	✓	✓
Multitud de orígenes de archivos	✗	✓	✗	✓
Reproducción remota	✓	✓	✗	✓
Sistema web	✓	✗	✓	✗
Recursos y listas disponibles globalmente	✓	✗	✓	✗
Control y sincronización de varios reproductores	✗	✗	✗	✗
Libre de proceso de emparejamiento	✗	✗	✓	✗

Capítulo 3. Aspectos Teóricos

En esta sección se describirán brevemente aquellos conceptos, herramientas y tecnologías usados y relacionados con el proyecto.

3.1 WebSocket

WebSocket es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. La API de WebSocket está siendo normalizada actualmente por el W3C, aunque el protocolo WebSocket ya fue normalizado por la IETF en 2011.

La tecnología está diseñada para ser implementada en navegadores y servidores web, pero puede utilizarse por cualquier aplicación cliente/servidor. Debido a que las conexiones TCP comunes sobre puertos diferentes al 80 son habitualmente bloqueadas por los administradores de redes, el uso de esta tecnología proporcionaría una solución a este tipo de limitaciones proveyendo una funcionalidad similar a la apertura de varias conexiones en distintos puertos.

En el lado del cliente, WebSocket está ya implementado en Mozilla Firefox 8, Google Chrome 4 y Safari 5y en Internet Explorer 10.

La comunicación por WebSockets se inicia con un *handshake* en el que cliente y servidor se ponen de acuerdo para establecer la comunicación bidireccional y se finaliza cuando uno de los lados cierra el canal de conexión. Esto se puede apreciar en el siguiente esquema de funcionamiento.

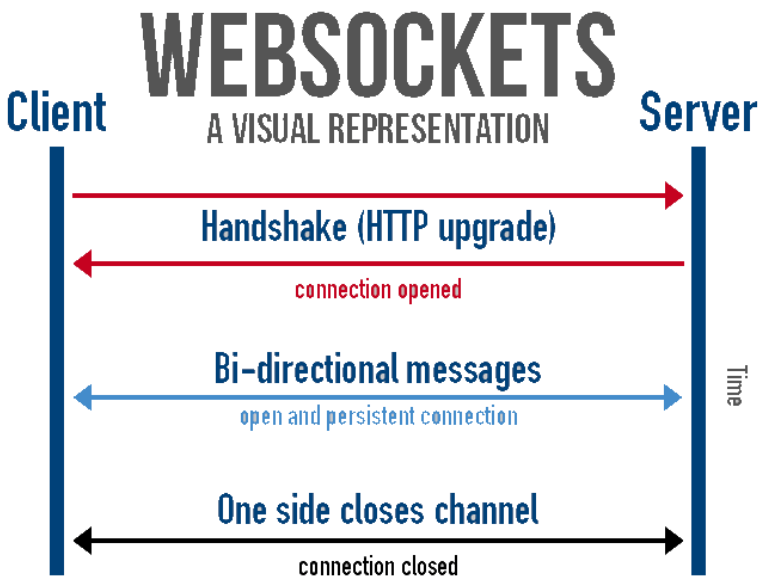


Figura 3.1. Esquema de funcionamiento de WebSockets

La necesidad de esta tecnología de comunicación para el proyecto radica en que las peticiones o eventos a los que se ha de dar respuesta, como puede ser pausar o parar un video, pasar al siguiente medio de reproducción, no siempre tienen su origen en el navegador web, sino que pueden ser originados por la aplicación de control remoto. En este caso, el control remoto enviará al servidor su petición, y sería el servidor el que tendría que notificar al cliente web que hay eventos nuevos a los que debe responder. Por lo tanto es necesario un medio por el cual el servidor pueda iniciar una comunicación con el cliente web, sin precisar de una petición previa por su parte.

Una alternativa a este modelo sería la utilización de mensajes AJAX para la técnica de *polling* (o encuesta periódica), en donde el cliente web preguntase regularmente al servidor si hay nuevos eventos a los que reaccionar, como se puede apreciar en el siguiente esquema.

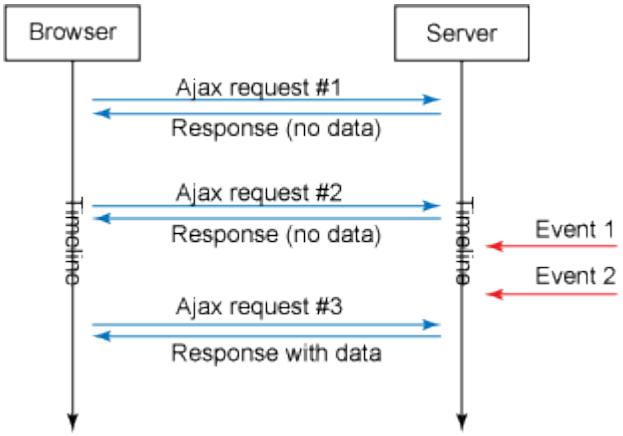


Figura 3.2. Esquema de funcionamiento de Ajax

La desventaja de este sistema sería la cantidad de peticiones y respuestas sin información, lo cual supondría tráfico inútil, y el retraso de la respuesta, pues el cliente web no sería notificado en el momento que se produce el evento, sino en la siguiente comunicación periódica que intercambiase con el servidor.

No obstante si que se han empleado comunicaciones AJAX para otros aspectos de la aplicación como es el caso de actualizaciones de las tablas de datos, inserción de nuevos elementos, ya que permite comunicar e intercambiar sólo la información necesaria para poder actualizar la información que se ha modificado en la página sin tener que refrescar la página completa.

3.2 STOMP

De las siglas en inglés Simple (or Streaming) Text Oriented Message Protocol, protocolo de mensajes orientado a texto simple, o difusión de texto. Es un protocolo de comunicación mediante mensajes independiente del lenguaje, lo que permite que se pueda conectar un cliente y un servidor que implementen el protocolo aunque estén escritos en distintos lenguajes de programación.

En el proyecto, se ha empleado este protocolo sobre los WebSockets para la comunicación entre servidor y cliente web en lo relativo a órdenes de reproducción multimedia.

Se ha utilizado una implementación en javascript para la parte cliente; y en java, como parte de framework *Spring* 4 para el servidor.

Los clientes realizan una petición para suscribirse a un canal de comunicación único para cada usuario por el que se intercambian los mensajes bidireccionales. Resaltar que el canal en único para el usuario, pero no para el cliente web, es decir, un usuario podría tener abiertos múltiples clientes web, y todos ellos se suscribirían al mismo canal. Esto es lo que permite la sincronización entre todos ellos, y que un único usuario controle la reproducción remota de tantos reproductores web como tenga conectados a la aplicación.

Adicionalmente, de manera periódica se intercambian "heartbit" o pulso, para comprobar que los extremos de la comunicación siguen activos, y cerrar la comunicación en caso de que uno de ellos se desconecte fortuitamente.

3.3 Spring Roo / gvNix

Spring Roo es una herramienta de desarrollo rápido de aplicaciones en java presentada en 2009 que aplica los principios de *convention-over-configuration* (convención por encima de configuración) primando lo que es comúnmente esperado y aceptado, sobre una configuración exhaustiva.

Su objetivo es mejorar la productividad en el desarrollo ofreciendo la generación automática de código a través de una interfaz de comandos donde el desarrollador describe el modelo de los datos con los que debe trabajar, la manera de mostrarlos, etc.

Estos comandos se traducen en código que implementa las necesidades básicas descritas, haciendo uso de AspectJ, programación orientada por aspectos, extensión del lenguaje java que agrupa el código no solo en clases java, sino también en archivos .aj asociados a una de esas clases y que se ocupan de un aspecto de su funcionalidad, como puede ser la persistencia, configuración etc.

Los aspectos son controlados automáticamente por el framework Spring roo y se adaptan a los cambios que hace el programador en las clases. Para modificar el funcionamiento de un método implementado en un método implementado en un archivo de aspecto es necesario llevar su código a su clase java asociada.

Esta herramienta está integrada dentro del framework de desarrollo Spring, una infraestructura conceptual y tecnológica con módulos y artefactos útiles para facilitar el desarrollo de aplicaciones Web den Java, de las más utilizadas actualmente.

GvNIX es una herramienta de desarrollo basada en *Spring roo* y patrocinada por la Dirección General de Tecnologías de la Información (DGTI) de la Conselleria de Hacienda y Administración Pública de la Generalitat Valenciana.

La principal ventaja de vgNIX respecto a herramienta originaria Spring Roo y el motivo por el que se ha elegido emplearla respecto a la primera es la incorporación de plugins y funcionalidades, e integración con otras tecnologías, entre las que cabe destacar jQuery,

Bootstrap, Jasper Reports, Datables. Con lo cual consigue unos resultados de generación mucho más adaptativos, dinámicos y versátiles.

Un resumen de las tecnologías empleadas por este framework se presenta en el siguiente cuadro.

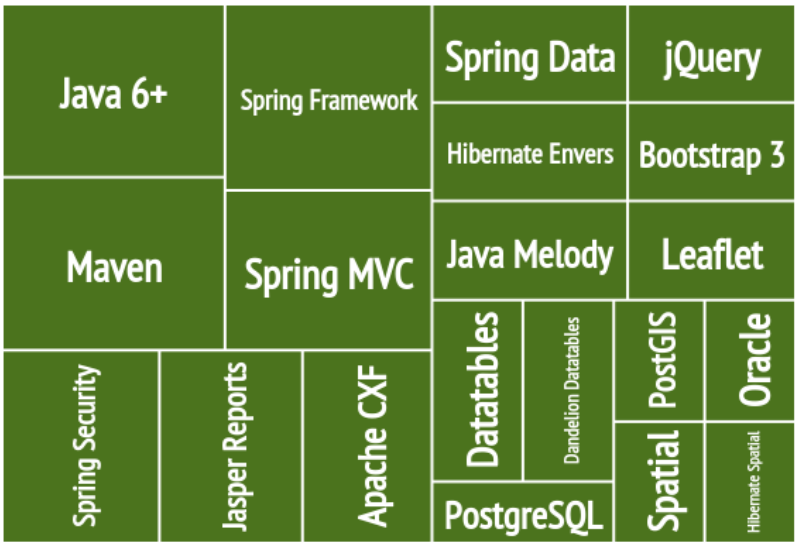


Figura 3.3. Cuadro de tecnologías empleadas por el framework gvNIX

Esta herramienta ha sido utilizada sobre todo en las primeras fases del proyecto pues facilitaba la puesta en marcha y construcción de una primera versión funcional, *scafolding (*andamiaje web) de las opciones principales de inserción, modificación y borrado de las entidades de la aplicación.

La versión de gvNIX utilizada en el proyecto es la 1.4, basada en Spring Roo 1.3.0.

3.4 VLC-Web Plugin

El plugin web VLC es un complemento para navegadores, similar a Flash que permite la reproducción dentro del navegador de todos los videos que el reproductor VLC media player puede leer. De hecho el funcionamiento básico es la inserción de un reproductor nativo en la página web. El plugin viene incluido al descargar la versión de escritorio del reproductor VLC.

La API de este plugin permite controlar su funcionamiento mediante llamadas a funciones javascript, por lo que se ha empleado en el proyecto para que combinado con la comunicación realizada mediante websockets, se puedan transmitir y sincronizar los eventos y peticiones relacionadas con la reproducción de contenidos multimedia.

Google Chrome ha decidido bloquear en su navegador los plugins que hacen uso de NPAPI (Netscape Plugin Application Programing Interface), como es el caso de VLC, por lo que a partir del 14 de abril de 2015 es necesario habilitar manualmente su utilización en una página de configuración. El soporte será retirado completamente a partir de septiembre de 2015.

Se han estudiado otras opciones de plugins de reproducción web, como 14 de abril de 2015 puede ser *WebChimera*, también basa den VLC, pero finalmente se ha optado por el primero debido a la sencillez de uso y la gran variedad de formatos aceptados. No obstante para futuras versiones y considerando el bloqueo de plugins NPAPI por parte de Chrome, es probable que se sustituya este plugin por otro que no requiera este tipo de APIs y por lo tanto si tenga soporte en navegador más extendido. Este hecho se ha tenido ya en cuenta en el desarrollo tratando de modularizar el código de manera que el cambio afecte a la menor cantidad de archivos posibles.

Capítulo 4. Planificación del Proyecto

En esta sección se describe la planificación seguida para el desarrollo del proyecto.

Con las pertinentes modificaciones para adaptarlo a las características y tamaño del proyecto el modelo es parecido a RUP (versión 4.1). RUP *(Rational Unified Process)* es un proceso de desarrollo de software que constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Es un modelo de proceso software orientado a objetos que utiliza el lenguaje de modelado UML. El proceso definido por RUP es iterativo, ya que se planifican iteraciones al final de las cuales se obtiene una versión más refinada del producto. Se centra en la arquitectura y su proceso está dirigido por los casos de uso, dando gran importancia a la comunicación con el cliente. Su estructura dinámica adaptable a las necesidades de cada proyecto y es por ello por lo que se ha seguido un proceso similar al definido en este modelo.

En la siguiente tabla se muestran las tareas de las que consta todo el proceso de realización del proyecto, nivel de la tarea, tiempo planificado para llevarlas a cabo, así como la descripción de la propia tarea. Cada día de duración se correspondería con 5 horas de trabajo.

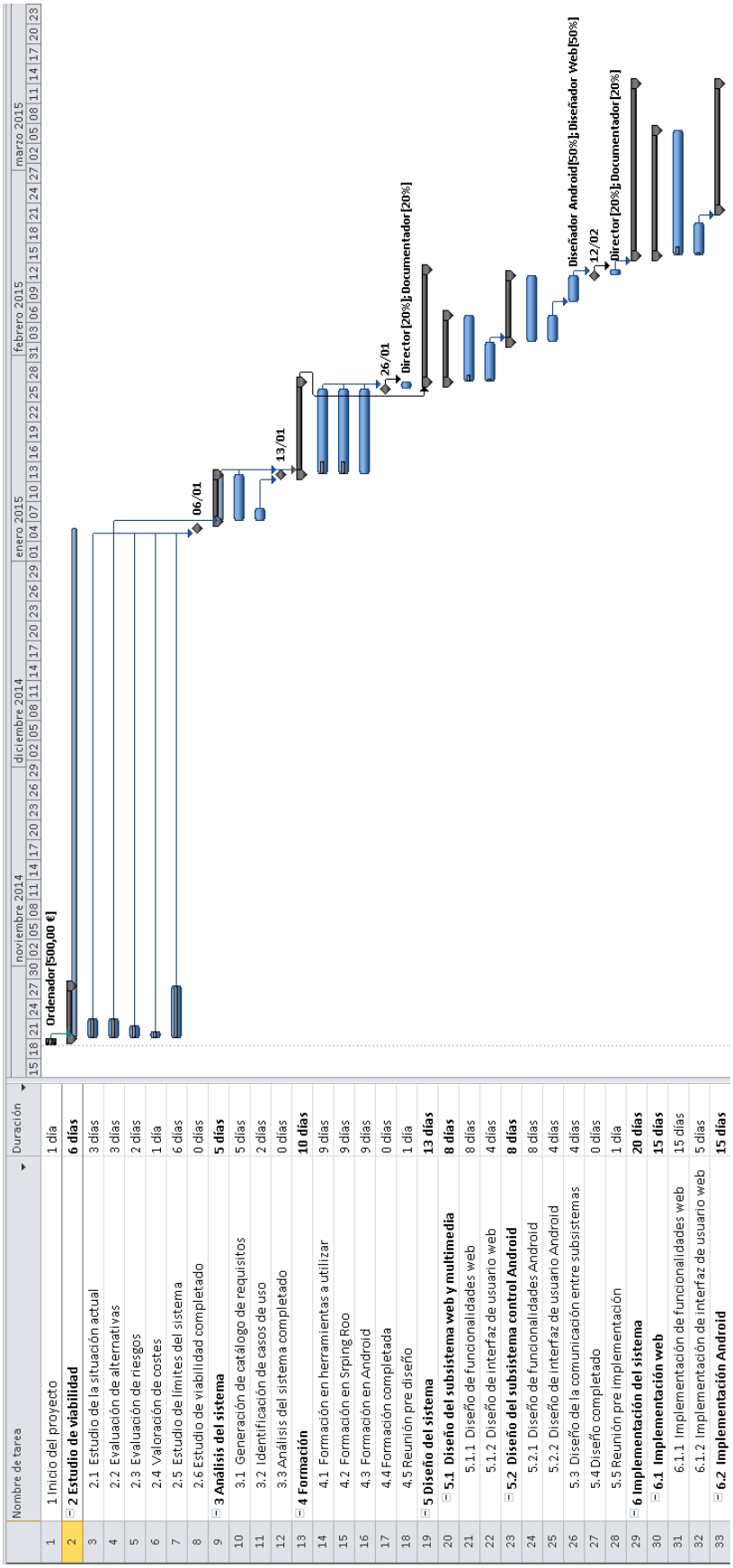
4.1 Listado de tareas

Nº	Tarea	Nivel	Duración	Descripción
1	Inicio proyecto	1	1 día	Puesta en marcha del proyecto
2	Estudio de viabilidad	1	6 días	Planteamiento y valoración de si el proyecto es factible en el tiempo disponible.
3	Estudio de la situación actual	2	3 días	Estudio de las tecnologías y utilidades que dan solución o son utilizadas para resolver el problema que el proyecto pretende resolver
4	Evaluación de alternativas	2	3 días	Evaluar las distintas opciones de cara a plantear una solución al problema que trata el proyecto.
5	Valoración de riesgos	2	2 días	Estudio de las distintas fuentes de riesgo que podrían relevantes proyecto y sus efectos sobre el desarrollo del mismo.
6	Valoración de costes	2	1 día	Estudio de los costes y presupuesto para la realización del proyecto
7	Estudio de límites del sistema	2	6 días	Establecimiento de los límites del proyecto, que se pretende implementar, y qué no.
8	Estudio de viabilidad completado	2	0 días	Hito fin de estudio de viabilidad
9	Análisis del sistema	1	5 días	Valoración de la información disponible y propuesta de los rasgos generales de la solución.
10	Generación de catálogo de requisitos	2	5 días	Recopilación de la serie de características que la solución a implementar debe cumplir
11	Identificación de casos de uso	2	2 días	Búsqueda y recopilación de los distintas acciones y escenarios en que se usará la solución buscada
12	Análisis del sistema completado	2	0 días	Hito fin de análisis del sistema

13	Formación	1	10 días	Adquisición y ampliación de conocimientos que permitan llevar a cabo la solución buscada
14	Formación en herramientas a utilizar	2	9 días	Familiarización con los entornos de desarrollo y herramientas a utilizar
15	Formación en Spring Roo	2	9 días	Ampliación de conocimientos en la herramienta de desarrollo rápido de aplicaciones Spring Roo
16	Formación en Android	2	9 días	Ampliación de conocimientos en la tecnología para desarrollo de aplicaciones en dispositivos móviles Android
17	Formación completada	2	0 días	Hito fin de formación
18	Reunión pre-diseño	1	1 día	Reunión con el director del proyecto previa al al diseño para poner en común las líneas generales del mismo
19	Diseño del sistema	1	13 días	Planteamiento y descomposición de las distintas partes que componen la solución a desarrollar
20	Diseño del subsistema web y multimedia	2	8 días	Diseño de la parte de la solución que llevará la gestión de usuarios, gestión de elementos multimedia, y reproducción multimedia
21	Diseño de funcionalidades	3	8 días	Diseño de las utilidades de que dispondrá la parte web de la aplicación
22	Diseño de interfaz de usuario	3	4 días	Diseño de la comunicación del sistema web con el usuario.
23	Diseño del subsistema control Android	2	8 días	Diseño de la parte de control remoto mediante dispositivo móvil
24	Diseño de funcionalidades	3	8 días	Diseño de las utilidades de que dispondrá la parte móvil de la aplicación
25	Diseño de interfaz de usuario	3	4 días	Diseño de la interfaz de comunicación entre el dispositivo móvil y es usuario
26	Diseño de la comunicación entre subsistemas	2	4 días	Diseño de la comunicación entre los distintos subsistemas constituyentes de la solución.
27	Diseño completado	2	0 días	Hito fin de diseño
28	Reunión pre-implementación	1	1 día	Reunión con el director del proyecto previa a la implementación para poner en común detalles sobre la misma y revisar el diseño
29	Implementación del sistema	1	20 días	Construcción y codificación del sistema diseñado
30	Implementación web	2	15 días	Construcción y codificación de la parte web de la aplicación
31	Implementación de funcionalidades web	3	15 días	Construcción y codificación de las funcionalidades que ofrecerá la aplicación en su parte web
32	Implementación de interfaz de usuario web	3	5 días	Construcción y codificación de la interfaz web con el usuario
33	Implementación Android	2	15 días	Construcción y codificación de la parte móvil de la aplicación
34	Implementación de funcionalidades	3	15 días	Construcción y codificación de las funcionalidades que ofrecerá la aplicación en su parte de dispositivo móvil
35	Implementación de interfaz de usuario	3	5 días	Construcción y codificación de la interfaz de comunicación con el usuario del dispositivo móvil
36	Implementación de comunicación entre subsistemas	2	7 días	Construcción y codificación del sistema de comunicación entre los subsistemas de la aplicación
37	Fin de implementación	2	0 días	Hito fin de implementación
38	Reunión pre-pruebas	1	1 día	Reunión con el director de proyecto para revisar los detalles de implementación y poner en común las líneas de realización de pruebas

39	Pruebas	1	21 días	Comprobación de que la aplicación implementada funciona como debería.
40	Pruebas de funcionalidades	2	7 días	Comprobación de que las funcionalidades funcionan correctamente
41	Pruebas de integración	2	9 días	Comprobación de la integración entre los distintos componentes
42	Pruebas de la interfaz de usuario	2	5 días	Comprobación de que la interfaz de usuario cuenta con las facilidades de utilización que son deseables y necesarias para el uso correcto de la aplicación
43	Pruebas completadas	2	0 días	Hito de fin de pruebas
44	Corrección de errores	1	8 días	Subsanación de los resultados de las pruebas que contravengan las expectativas o funcionamientos inesperados.
45	Introducción de mejoras	1	9 días	Incorporación a la aplicación de aquellas modificaciones que hayan sido apreciadas como susceptibles de mejorar la experiencia del usuario manejando la aplicación
46	Reunión post-mejoras	1	1 día	Reunión con el director par comentar los resultados de las pruebas, las correcciones realizadas y las mejoras implementadas
47	Generación de documentación y manuales	1	13 días	Recopilación y redacción de toda la información disponible sobre el proceso de desarrollo de la aplicación y su modo de utilización.
48	Reunión sobre documentación	1	1 día	Reunión con el director para revisar la documentación redactada
49	Preparación de la presentación inaugural del proyecto	1	2 días	Resumen de la información de la documentación para la presentación a los clientes de las utilidades que ofrece la aplicación y su modo de uso.
50	Documentación completada	1	0 días	Hito de fin de documentación
51	Presentación inaugural del proyecto	1	1 día	Exposición a los clientes de las utilidades de la aplicación y ventajas respecto a las opciones anteriores a la existencia de la aplicación.
52	Proyecto inaugurado	1	0 días	Hito de inauguración de proyecto
53	Implantación	1	2 días	Puesta en marcha de la aplicación desarrollada en un entorno de explotación
54	Cierre de proyecto	1	0 días	Hito de clausura del proceso de desarrollo de la primera versión final de la aplicación

La distribución de tareas de la tabla anterior se puede ver también en el siguiente diagrama de Gantt



	Nombre de tarea	Duración
1	1 Inicio del proyecto	1 día
2	**2 Estudio de viabilidad**	**6 días**
3	2.1 Estudio de la situación actual	3 días
4	2.2 Evaluación de alternativas	3 días
5	2.3 Evaluación de riesgos	2 días
6	2.4 Valoración de costes	1 día
7	2.5 Estudio de límites del sistema	6 días
8	2.6 Estudio de viabilidad completado	0 días
9	**3 Análisis del sistema**	**5 días**
10	3.1 Generación de catálogo de requisitos	5 días
11	3.2 Identificación de casos de uso	2 días
12	3.3 Análisis del sistema completado	0 días
13	**4 Formación**	**10 días**
14	4.1 Formación en herramientas a utilizar	9 días
15	4.2 Formación en Spring Roo	9 días
16	4.3 Formación en Android	9 días
17	4.4 Formación completada	0 días
18	4.5 Reunión pre diseño	1 día
19	**5 Diseño del sistema**	**13 días**
20	**5.1 Diseño del subsistema web y multimedia**	**8 días**
21	5.1.1 Diseño de funcionalidades web	8 días
22	5.1.2 Diseño de interfaz de usuario web	4 días
23	**5.2 Diseño del subsistema control Android**	**8 días**
24	5.2.1 Diseño de funcionalidades Android	8 días
25	5.2.2 Diseño de interfaz de usuario Android	4 días
26	5.3 Diseño de la comunicación entre subsistemas	4 días
27	5.4 Diseño completado	0 días
28	5.5 Reunión pre implementación	1 día
29	**6 Implementación del sistema**	**20 días**
30	**6.1 Implementación web**	**15 días**
31	6.1.1 Implementación de funcionalidades web	15 días
32	6.1.2 Implementación de interfaz de usuario web	5 días
33	**6.2 Implementación Android**	**15 días**

Figura 4.1. Diagrama de Gantt del desarrollo del proyecto (parte 1)

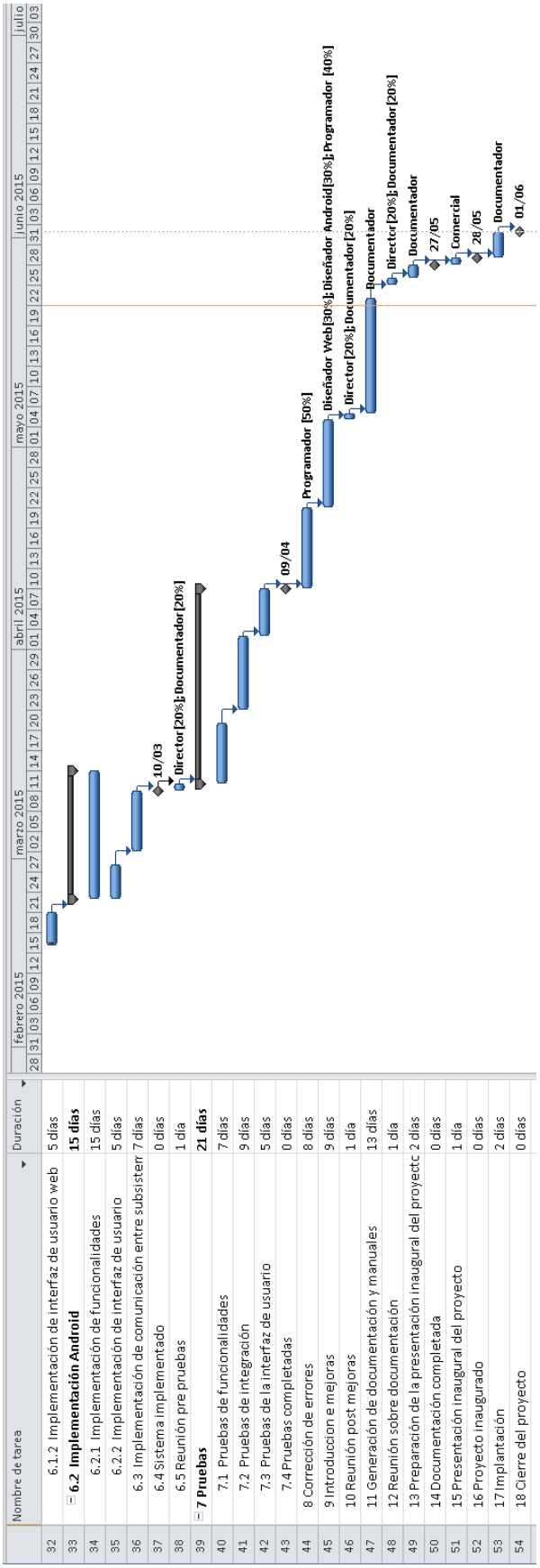


Figura 4.2. Diagrama de Gantt del desarrollo del proyecto (parte 2)

Como se puede ver en la tabla y los diagramas, el proyecto cuenta con 54 tareas (9 de ellas hitos), con una duración entre 1 y 21 días, siendo la tarea más corta la puesta en marcha del proyecto y la más larga la correspondiente a las pruebas del sistema, aunque esta última está compuesta de subtareas de menor nivel. Las tareas no divididas más largas se corresponden con las implementaciones de las funcionalidades de los distintos módulos, de 15 días de duración cada uno.

La mayor cantidad de tareas simultáneas tiene lugar al principio del proyecto cuando se realiza el estudio de la situación actual, estudio de alternativas, evaluación de riesgos, costes y estudio de límites del sistema; cinco en total.

Se cuentan 38 tareas críticas con un máximo de simultaneidad de 3 que tiene lugar en el periodo de formación en las tecnologías de desarrollo.

Hay pocas tareas con holgura para su realización, una de 3 días y dos de 5, por lo que su holgura media sería de 4,3 días.

El proyecto tiene una duración total estimada de 161 días.

Capítulo 5. Análisis

Este apartado contendrá toda la especificación de requisitos y toda la documentación del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

5.1.1 Determinación del Alcance del Sistema

El proyecto Mediaserver, tiene por objetivo permitir a sus usuarios gestionar sus colecciones de elementos multimedia, y controlar su reproducción remotamente, por lo que se compone de dos parte principales:

- Por una parte un sistema web que permitiría la gestión de usuarios, elementos multimedia, listas de reproducción. Este sistema web, tendrá a su vez dos vistas diferenciadas:
    - Una vista de usuario standard que incluirá opciones para el registro, configuración, y manejo de elementos multimedia y listas de reproducción.
    - Una vista de administración, que además de las opciones del usuario standard permitirá la gestión de todos los usuarios de la aplicación, creación directa de nuevos usuarios, asignación de roles, modificación de datos, y control general de la aplicación.
- Por otra parte una aplicación Android desde la que los usuarios puedan controlar qué es lo que se reproduce en el navegador o navegadores web que hayan iniciado sesión con su cuenta.

5.2 Requisitos del Sistema

5.2.1 Obtención de los Requisitos del Sistema

A continuación se muestra el catálogo de requisitos que debe satisfacer el sistema desarrollado.

Los requisitos funcionales serán identificados como RFxxx para hacer referencia a ellos, y los no funcionales como RNFxxx.

5.2.1.1 Requisitos funcionales

Código	Nombre Requisito	Descripción del Requisito
RF1.1	Insertar Usuario	Una vez leídos y validados sus datos, se añadirá un usuario al sistema.
RF1.2	Datos de usuario a guardar	Además de la dirección de correo electrónico y contraseña, que actuarán como credenciales, se guardará el nombre y apellidos del usuario.
RF1.3	Control anti registro automatizado	El proceso de registro incorporará sistemas de control como el uso de recaptchas y confirmación por correo para la activación de la cuenta.
RF1.4	Confirmación de contraseña	La contraseña se pedirá por duplicado en el registro para evitar posibles errores fortuitos, que luego dificulten una autenticación correcta.
RF1.5	Cambio de contraseña	El usuario podrá cambiar su contraseña de inicio de sesión
RF1.6	Restablecimiento de contraseña	Si el usuario olvida su contraseña dispondrá de un servicio de restablecimiento de contraseña mediante el cual recibirá un correo con un enlace a una página donde podrá establecer una nueva contraseña.
RF1.7	Bloqueo de usuarios	Un usuario administrador podrá bloquear el acceso de un determinado usuario al sistema.
RF2.1	Añadir medios	El usuario debe poder añadir a su lista de medios un nuevo medio identificado por un nombre y una dirección donde se ubica.
RF2.2	Sugerencia de medios	El sistema pondrá a disposición de los usuarios una herramienta que les facilite la búsqueda de ubicaciones de recursos a partir del nombre especificado, evitando que tengan que salir del sitio de la aplicación para buscarlos.
RF2.3	Vista de medios	El usuario podrá seleccionar distintas cantidades de medios que desea ver en la misma página.

RF2.4	Filtrado de medios	El usuario podrá filtrar los medios mostrados según una cadena de texto de manera que le sea fácil encontrar un medio concreto dentro de los que ya ha añadido.
RF2.5	Búsqueda de medios por tipo	Se podrá realizar una búsqueda entre los medios insertados por el tipo de medio: audio o video.
RF2.6	Eliminar medios	El usuario debe poder eliminar un medio o un conjunto de ellos de su lista de medios.
RF2.7	Edición de medios	El usuario podrá editar tanto el nombre como la dirección de un medio una vez insertado.
RF2.8	Origen de medios	Los medios podrán estar alojados en cualquier lugar de internet accesible públicamente.
RF3.1	Añadir lista de reproducción	El usuario debe poder añadir una lista de reproducción a su conjunto de listas, identificada por un nombre.
RF3.2	Eliminar lista de reproducción	El usuario debe poder eliminar una lista de reproducción o una selección de ellas de su conjunto de listas.
RF3.3	Editar de listas de reproducción	El usuario podrá editar El nombre de las listas de reproducción que haya creado
RF3.4	Insertar de medios en una lista de reproducción	El usuario podrá añadir a una lista de reproducción medios de entre los que haya creado en su lista de medios
RF3.5	Edición de componentes de una lista de reproducción	El usuario podrá editar que medios componen una lista de reproducción
RF3.6	Eliminar medios de una lista de reproducción	El usuario podrá eliminar de una lista de reproducción un medio o conjunto de medios.
RF4.1	Reproducción de medios y listas de reproducción	El usuario podrá reproducir sus medios y listas de reproducción en la página que el sistema crea para visualizar el medio o lista concreto.
RF4.2	Control remoto	El usuario podrá modificar parámetros como el estado, avance y volumen de la reproducción de manera remota empleando la aplicación móvil asociada.
RF4.3	Sincronización	El usuario podrá controlar la reproducción simultáneamente en todos los reproductores web asociados a su cuenta.

FF4.4	No emparejamiento	El control remoto no necesitará un emparejamiento previo con el navegador web que está reproduciendo los medios para poder controlarlo.
RF5.1	Inicio de sesión	El usuario deberá iniciar sesión antes de crear, o reproducir medios y listas de reproducción.
RF5.2	Cierre de sesión	El usuario podrá decidir finalizar la sesión en cualquier momento, deteniendo sus actividades en el sistema
RF6	Supervisión de medios listas y asignaciones	Un usuario administrador podrá ver y editar cualquier medio o lista creados por los usuarios así como controlar qué medios están incluidos en las listas de reproducción para poder corregir comportamientos inadecuados.
RF7	Roles de usuario del sistema	Un usuario recién registrado dispondrá por defecto del rol USER con los permisos y acciones básicas de la aplicación. Este rol podrá ser modificado pon un usuario administrador.
RF8	Internacionalización	El usuario podrá cambiar el idioma de la aplicación en cualquier momento mediante una opción de configuración. Por defecto se escogerá el idioma del navegador.
RF9	Ayuda	La aplicación contará con un sistema de ayuda que orientará al usuario para solventar las dudas sobre su correcta utilización.

5.2.1.2 Requisitos no funcionales

Código	Nombre Requisito	Descripción del Requisito
RNF1	Multiplataforma	El servidor web debe funcionar correctamente desplegado en un contenedor de aplicaciones web sobre sistemas operativos Windows y Linux
RNF2.1	Aplicación Web	Se debe crear una sistema web accesible a través de navegadores web
RNF2.2	Estándar UNE 139803:2012	Se debe seguir el estándar UNE 139803:2012 de Requisitos de Accesibilidad para contenidos en la web
RNF2.3	Navegador Web	Para utilizar la aplicación web, será necesario el uso de un navegador web compatible con HTML5, CSS3, javascript, etc.
RNF3	Java 1.7	El equipo en el que se despliegue el servidor debe tener instalada una versión de java 7 o superior.

RNF4	Android 4.1 JellyBean	La aplicación móvil requerirá que el dispositivo en el que se instale funcione con una versión de Android 4.1 o superior.
RNF5	MySQL	El sistema precisa para su funcionamiento del acceso local o remoto a una base de datos MySQL configurada para el manejo y guardado de los datos de la aplicación.
RNF6	Usabilidad	El procedimiento de utilización sistema debe ser fácil de aprender para los usuarios, y ofrecer sistemas de ayuda para resolver las posibles dudas que puedan surgir
RNF7	Seguridad	Se deben contemplar aspectos de seguridad de manera que los datos de los usuarios se almacenen y manejen de forma segura previniendo un posible robo de información.
RNF8	Tiempo de respuesta	El tiempo de respuesta a las peticiones de control remoto, una vez completada la carga inicial del medio o lista, no ha de ser superior a 2 segundos.

5.2.2 Identificación de Actores del Sistema

En este caso solo se han identificado cuatro tipos de actores tipos de actores atendiendo a su estado de autenticación, rol de usuario, y medio de acceso web o móvil.

- Usuario anónimo: no ha iniciado sesión en el sistema.
- Usuario registrado: ha iniciado correctamente sesión en la aplicación web.
- Usuario administrador: tiene un rol de usuario que le capacita para realizar acciones de administración.
- Usuario móvil: Accede al sistema desde la aplicación móvil para controlar la reproducción abierta en su sesión web.

5.2.3 Especificación de Casos de Uso

A continuación se presentan los casos de uso esperados en el sistema, que representan una secuencia de pasos simples llevados a cabo por los actores anteriormente descritos para conseguir los objetivos que permite la aplicación; primero en forma de diagramas de casos de uso y después su correspondiente explicación textual de cada caso.

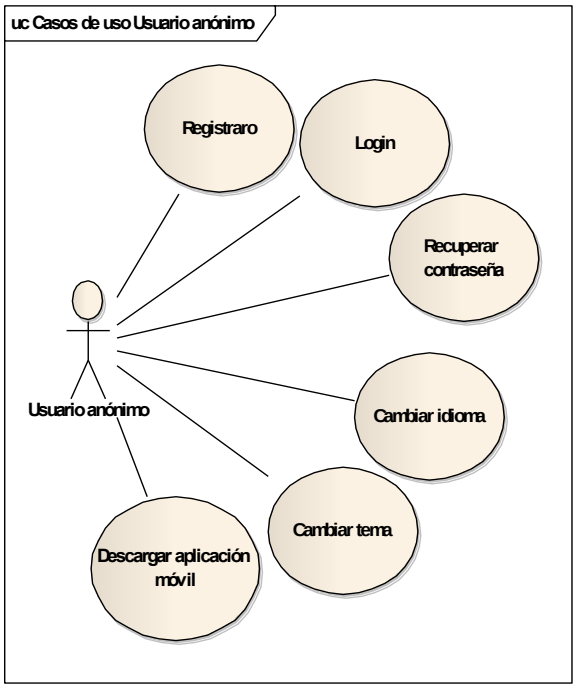


Figura 5.1. Caso de uso 1.Usuario anónimo

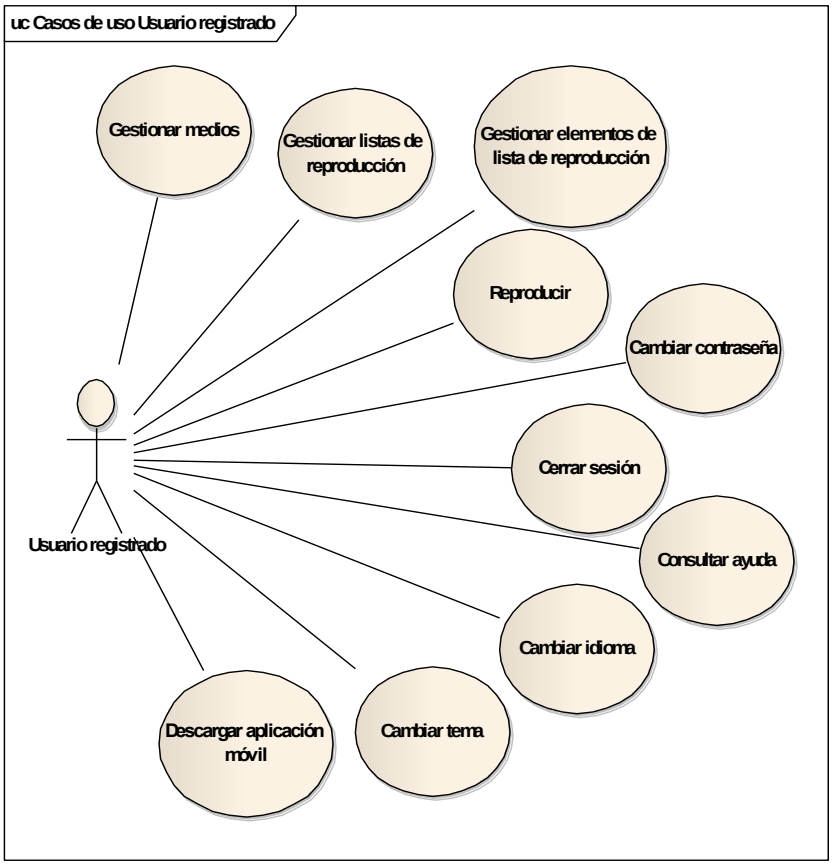


Figura 5.2. Caso de uso 2: Usuario registrado

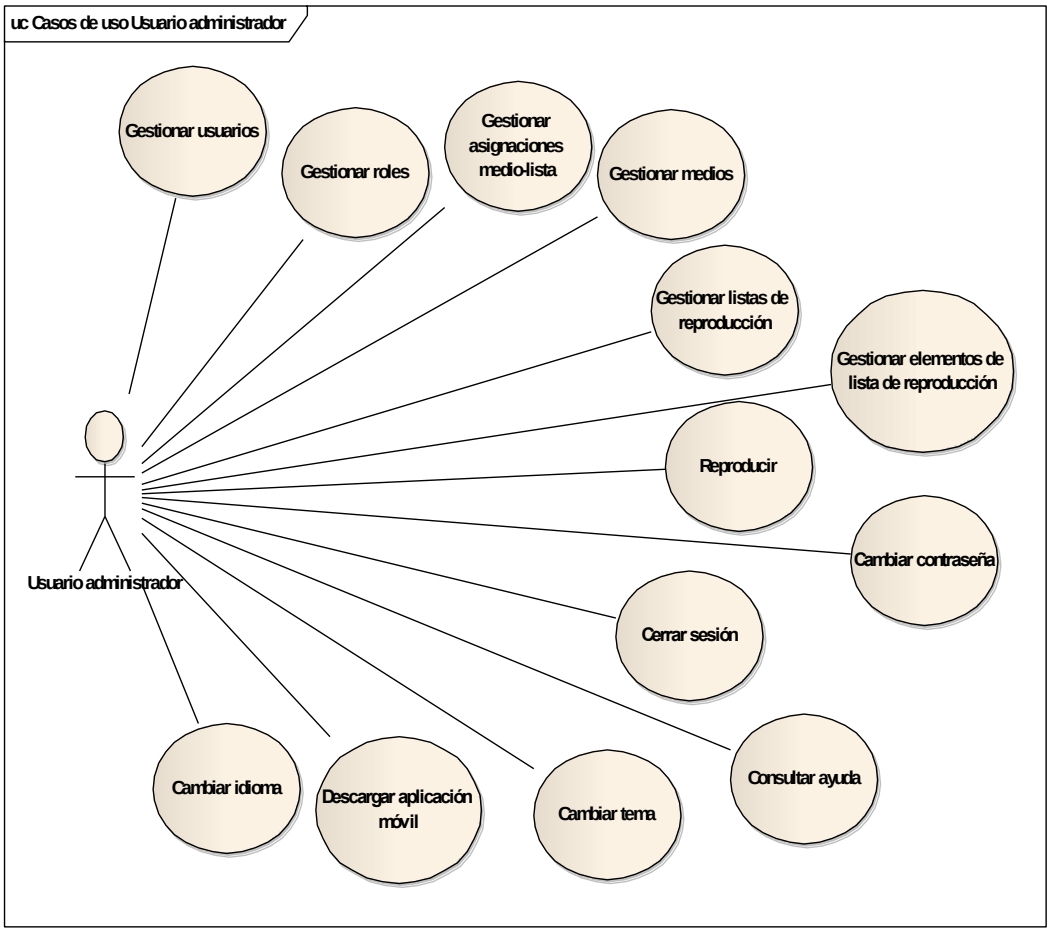


Figura 5.3. Caso de uso 3: Usuario administrador

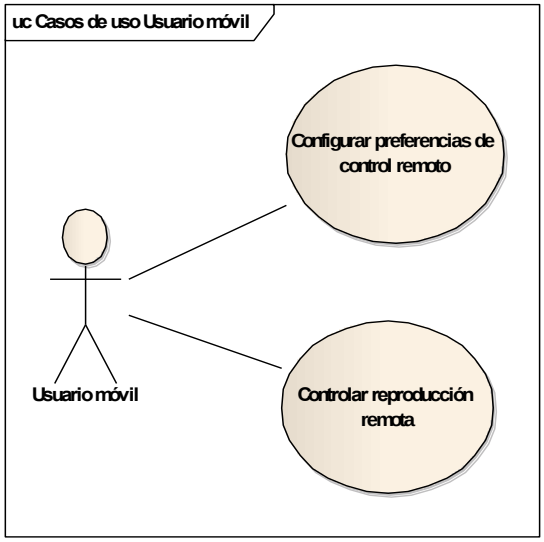


Figura 5.4. Caso de uso 4: Usuario móvil

Nombre del Caso de Uso
Registro
Descripción
El usuario abre la página de registro y rellena los formularios con sus datos, así como el

recaptcha con el texto de la imagen, o sonido reproducido. Si todo está correcto el sistema enviará a su dirección de correo un email de confirmación con una URL a la que el usuario debe acceder con el navegador. Tras esto, el usuario tendrá una cuenta en el sistema y podrá proceder a loguearse.

Nombre del Caso de Uso
Recuperar contraseña
Descripción
El usuario accederá a la página de recuperación de contraseña, donde se le preguntará por su dirección de correo. Tras introducirla correctamente se le enviará por correo un enlace a la página donde podrá establecer una nueva contraseña para su cuenta. Tras abrir el enlace y completar el correspondiente formulario con su nueva contraseña, esta quedará establecida y podrá utilizarla para loguearse.

Nombre del Caso de Uso
Login
Descripción
El usuario introducirá en la página de login sus credenciales para iniciar sesión y acceder a todas las funciones disponibles como usuario registrado.

Nombre del Caso de Uso
Cambiar idioma
Descripción
El usuario seleccionará en las opciones el idioma en que desea que se le muestre la interfaz y a partir de ese momento, esta cambiará al idioma deseado

Nombre del Caso de Uso
Descargar aplicación móvil
Descripción
El usuario accederá a una página desde la que podrá descargar la aplicación móvil de control remoto para instalarla en su dispositivo móvil. Una vez instalada y configurada con los datos de su cuenta, ya podrá controlar la reproducción de los medios y listas de su sesión web.

Nombre del Caso de Uso
Cambiar tema
Descripción
El usuario seleccionará un tema de entre los disponibles: estándar y alternativo, lo cual establecerá un esquema de colores en la aplicación al gusto del usuario.

Nombre del Caso de Uso
Gestionar medios
Descripción
El usuario accederá a la lista de medios de su cuenta, donde podrá: • Crear nuevos medios, introduciendo un nombre y una ubicación del recurso, o utilizando el sistema de sugerencia y búsqueda, donde buscará un nombre y

seleccionará uno de los resultados, que rellenará automáticamente el formulario para añadir un nuevo medio.
- Editar los datos de medios ya creados en su lista de medios.
- Eliminar medios ya creados de su lista de medios.

Nombre del Caso de Uso
Gestionar listas de reproducción
Descripción
El usuario accederá a la página de listas de reproducción donde podrá crear nuevas listas de reproducción a partir de un nombre, editar el nombre de las ya existentes o eliminarlas. Dentro de la visión de cada lista de reproducción, el usuario podrá añadir, editar o eliminar los medios que se reproducirán dentro de esa lista concreta.

Nombre del Caso de Uso
Gestionar elementos de lista de reproducción
Descripción
El usuario accederá a la página de una lista de reproducción concreta, donde podrá añadir, editar o eliminar los medios que se reproducirán dentro de esa lista.

Nombre del Caso de Uso
Reproducir
Descripción
El usuario accederá a la página para mostrar un medio o lista de reproducción determinada, y tendrá a su disposición un reproductor web con controles para poder reproducir el medio o lista en cuestión.

Nombre del Caso de Uso
Cambiar contraseña
Descripción
El usuario accederá a la página de cambio de contraseña donde tras completar el formulario con los datos de su contraseña actual, y contraseña nueva, se procederá a establecer la que será su contraseña de inicio de sesión a partir de ese momento.

Nombre del Caso de Uso
Cerrar sesión
Descripción
El usuario finalizará su sesión pasando a ser de nuevo un usuario anónimo

Nombre del Caso de Uso
Consultar ayuda
Descripción
Se accede a la ayuda de la aplicación

Nombre del Caso de Uso
Gestionar usuarios

Descripción
Un usuario administrador accederá al listado de usuarios donde podrá revisar los usuarios del sistema, editar sus datos, eliminarlos o crear usuarios nuevos. También podrá bloquear su acceso, o activar manualmente su cuenta, (activación que normalmente se lleva a cabo mediante el email de confirmación en el registro).

Nombre del Caso de Uso
Gestionar roles
Descripción
Un usuario administrador puede cambiar el rol de un usuario para concederle permisos de administrador o usuario estándar.

Nombre del Caso de Uso
Gestionar asignaciones medio-lista
Descripción
Un usuario administrador puede revisar que medios pertenecen a qué listas, y a que usuario pertenecen dichas listas para comprobar que se está haciendo un uso correcto del sistema, y realizar modificaciones en caso necesario.

Nombre del Caso de Uso
Configurar preferencias de control remoto
Descripción
Un usuario de la aplicación móvil debe introducir ciertos parámetros en la configuración de su aplicación como son sus credenciales y dirección del servidor para que esta se pueda comunicar correctamente con el servidor y las reproducciones de su cliente web reciban las órdenes del control remoto.

Nombre del Caso de Uso
Controlar reproducción remota
Descripción
Mediante los controles de la aplicación móvil una vez configurada, el usuario podrá controlar diversos parámetros de la reproducción de medios y listas de reproducción en el reproductor web, como son el estado de avance, el volumen, el cambio de medio hacia atrás o adelante, parada, cambio a pantalla completa, etc.

5.3 Identificación de los Subsistemas en la Fase de Análisis

El objetivo de esta sección es analizar el sistema para poder descomponerlo en sistemas más pequeños (subsistemas) que faciliten su posterior análisis.

5.3.1 Descripción de los Subsistemas

Los subsistemas en que se ha dividido la aplicación son los siguientes:

- Subsistema de gestión y reproducción web: aglutinará las funcionalidades de gestión de las entidades de la aplicación, medios, listas, usuarios así como su respaldo en la base de datos. También manejará la reproducción de medios y listas.

- Subsistema de comunicación: Será el encargado de comunicar las órdenes del sistema remoto Android con los reproductores web así como de sincronizar estos entre sí.

- Subsistema de control remoto Android: Encargado de enviar las peticiones de control remoto del usuario al sistema de comunicación del servidor.

5.3.2 Descripción de los Interfaces entre Subsistemas

Una vez identificados los subsistemas, debemos también describiremos cómo será la comunicación entre los mismos. Como veremos en el siguiente esquema lo más significativo de la comunicación se produce entre los subsistemas de comunicación y los otros subsistemas.

Por una parte la comunicación con el subsistema web se realiza mediante WebSockets con el protocolo STOMP sobre HTTP, para disponer de una comunicación bidireccional.

Por otra parte la comunicación con el subsistema de control Android se realiza mediante mensajes HTTP de petición-respuesta.

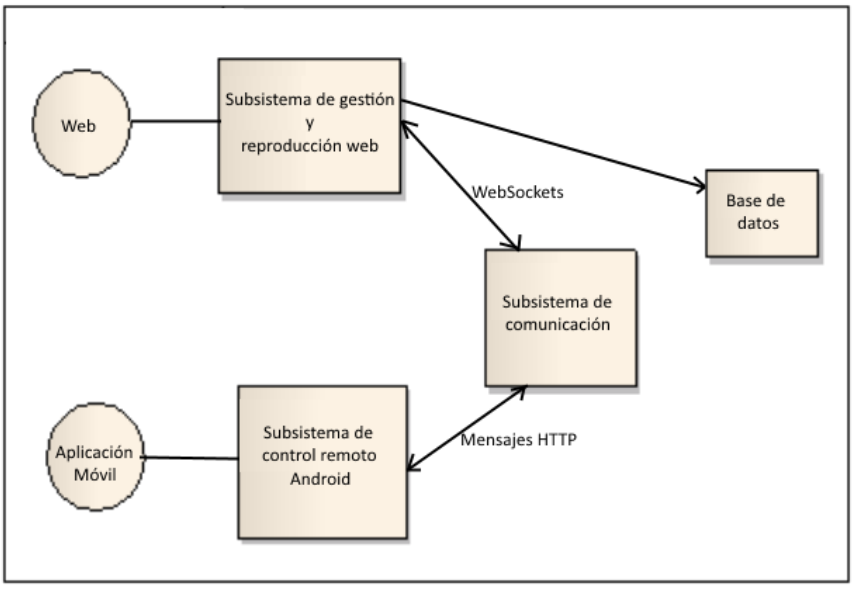


Figura 5.5. Interfaces entre los subsistemas

5.4 Diagrama de Clases Preliminar del Análisis

5.4.1 Diagrama de Clases

El siguiente diagrama se muestra los paquetes y clases ya identificados como necesarios en la fase de análisis. Se centra fundamentalmente en el subsistema de gestión, que es el que manejará las entidades centrales de la lógica de la aplicación.

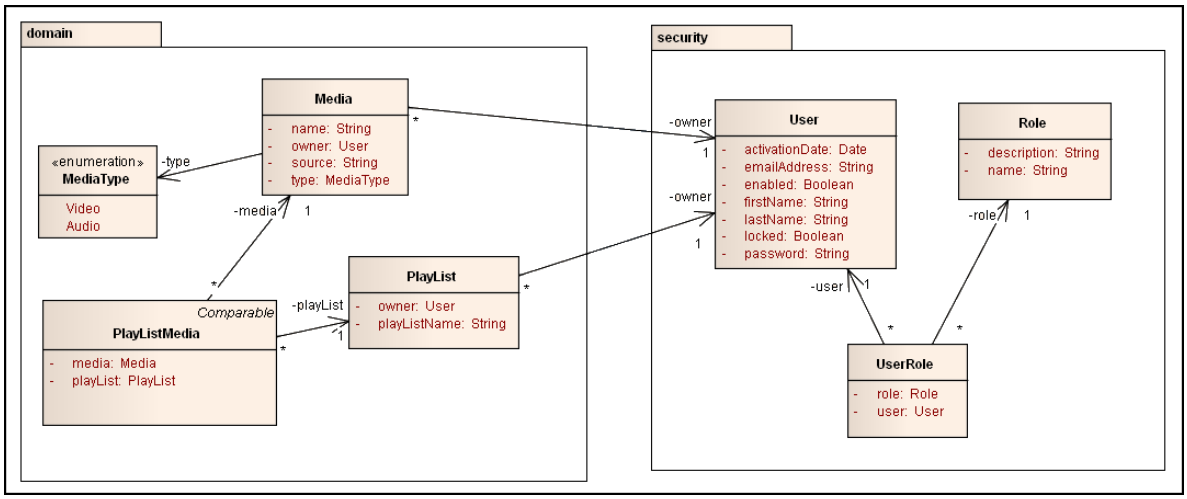


Figura 5.6. Diagrama de clases del análisis

5.4.2 Descripción de las Clases

A continuación se describirá brevemente el objetivo de cada clase identificada, así como de sus atributos

Nombre de la Clase
Media
Descripción
Clase que representa un medio o recurso de reproducción.
Responsabilidades
Representar cada uno de los medios o unidades de reproducción de la aplicación, compuestos básicamente por un nombre, y una URL donde se ubican en la web.
Atributos Propuestos
name: nombre del medio. **owner:** usuario propietario o creador del medio. **source:** URL que describe la ubicación del medio en internet. **type**: tipo de recurso (video o audio).

Nombre de la Clase
PlayList
Descripción
Clase que representa una lista de reproducción compuesta por medios de la clase Media.
Responsabilidades
Representar la agrupación de una colección de medios para su reproducción secuencial.
Atributos Propuestos
owner: usuario propietario o creador de la lista de reproducción. **playListName:** nombre que identifica la lista de reproducción.

Nombre de la Clase
PlayListMedia
Descripción
Clase de asociación entre medios y listas de reproducción.
Responsabilidades
Permitir una asociación entre medios (Media) y listas de reproducción (PlayList) evitando los problemas derivados de una cardinalidad múltiple por ambos extremos.
Atributos Propuestos
media: medio implicado en la asociación. **playList:** lista de reproducción implicada en la asociación.

Nombre de la Clase
User
Descripción
Clase que representa a los usuarios de la aplicación.
Responsabilidades
Representar a cada uno de los usuarios de la aplicación guardando sus datos, credenciales, fecha de activación y estado de bloqueo.
Atributos Propuestos
activationDate: fecha en la que la cuenta fue activada mediante el email de confirmación. **emailAdress:** dirección de email, única para cada usuario. Actúa como nombre en las credenciales de login. **enabled:** estado de habilitación de la cuenta. En el momento del registro, ésta no estará habilitada sino que se habilitará mediante un email de confirmación. **firstName:** nombre del usuario. **lastName:** apellidos del usuario. **locked:** estado de bloqueo de la cuenta. La modificación de este estado permitirá a un usuario administrador restringir el acceso a un determinado usuario. **password:** contraseña de acceso a la aplicación para el usuario de la cuenta.

Nombre de la Clase
Role
Descripción
Clase que representa un nivel de acceso para los usuarios de la aplicación
Responsabilidades

Representar un rol de la aplicación para el control de acceso a funcionalidades por parte de los usuarios
Atributos Propuestos
name: nombre del rol. **description:** descripción textual del objetivo de dicho rol.

Nombre de la Clase
UserRole
Descripción
Clase de asociación entre usuarios y roles.
Responsabilidades
Permitir una asociación entre usuarios (User) y roles (Role) evitando los problemas derivados de una cardinalidad múltiple por ambos extremos.
Atributos Propuestos
role: rol implicado en la asociación. **user:** usuario implicado en la asociación.

5.5 Análisis de Casos de Uso y Escenarios

En esta sección se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios.

Los escenarios describen las interacciones entre los usuarios y el sistema e incluyen información acerca de los objetivos, expectativas, motivaciones, acciones y reacciones que se llevan a cabo.

Los escenarios reflejan la forma en la que el sistema se comporta para definir el comportamiento deseado del *software* de manera que complemente a los requisitos funcionales antes descritos.

5.5.1 Caso de Uso 1: Registro

Registro	
Precondiciones	El usuario no está registrado
Poscondiciones	• El formulario de datos de registro ha sido completado correctamente. • El usuario queda registrado en el sistema y con un rol asignado.
Actores	Usuario anónimo.
Descripción	1. El usuario accederá a la página de registro. 2. Rellenará la información necesaria para completar el formulario. 3. Rellena el campo de recaptcha con el texto de la imagen asociada. 4. Sus datos se guardan correctamente, se envía un email de confirmación a la dirección que ha especificado, y se le notifica el éxito del registro. 5. El usuario accede a su cuenta de correo y abre el enlace presente en el mensaje de registro para proceder a la activación. 6. Se le notifica el éxito de la activación y puede proceder al proceso de login.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Los datos no han sido rellenados correctamente, o de manera completa.
	  ○ Se vuelve al paso 2 e informa al usuario de los errores cometidos en el formulario o datos que falta por rellenar.
	• **Escenario Alternativo 2**: El usuario tiene dificultades para reconocer el texto de la imagen del recaptcha.
	  ○ El usuario puede seleccionar reconocer y transcribir un mensaje de audio, que escuchará, transcribirá y podrá continuar con el registro con normalidad. Este paso sustituiría al paso 3.
	• **Escenario Alternativo 3**: La dirección de correo especificada ya

	ha sido registrada ○ Se vuelve al paso 2 informando que la dirección de correo especificada no está disponible y ha de utilizar otra. • **Escenario Alternativo 4**: La dirección de correo especificada es errónea o no existe. ○ El email de confirmación no llega a la dirección de correo del usuario, lo que impide la activación y el proceso de registro ha de reiniciarse con una dirección correcta.
Excepciones	• **La base de datos no está disponible**: No se pueden guardar el nuevo usuario en la base de datos por problemas de conexión. ○ Notificar un error asociado al problema encontrado. • **Sistema externo de envió de correos no disponible**: No se puede enviar el email de confirmación por indisponibilidad del servicio externo de envío. ○ Notificar un error asociado al problema encontrado. • **Servicio de recaptcha no disponible**: No se puede preguntar ni resolver el recaptcha. ○ Notificar un error asociado al problema encontrado.
Notas	Dependiendo del proveedor del correo del usuario, el email de confirmación podría tardar más o menos en llegar, y ser o no identificado como spam, por lo que el usuario podría tener que buscarlo en la carpeta de spam de su correo.

5.5.2 Caso de Uso 2: Recuperar contraseña

Recuperar contraseña	
Precondiciones	El usuario no está logueado, no ha iniciado sesión, pero sí tiene una cuenta en el sistema.
Poscondiciones	El formulario de recuperación y restablecimiento de contraseña han sido completados correctamente.
Actores	Usuario anónimo.
Descripción	1. El usuario accederá a la página de recuperación de password. 2. Rellenará el formulario con la dirección de correo asociada a su cuenta. 3. El usuario accede a su cuenta de correo y abre el enlace presente en el mensaje de restablecimiento de contraseña. 4. Completa el formulario donde se le pedirá escribir por duplicado su nueva contraseña. 5. Se le notifica el éxito del restablecimiento de contraseña y se puede proceder al proceso de login.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: La dirección de correo especificada es errónea o no está registrada. ○ Se notifica al usuario que la dirección no está registrada

	y se pide al usuario que la reintroduzca correctamente (paso 2). • **Escenario Alternativo 2**: Los datos no han sido rellenados correctamente, o de manera completa. ○ Se vuelve al paso 4 e informa al usuario de los errores cometidos en el formulario o datos que falta por rellenar.

Excepciones	• **La base de datos no está disponible**: No se pueden consultar la existencia de la cuenta identificada por la dirección de correo.
	  ○ Notificar un error asociado al problema encontrado.
	• **Sistema externo de envió de correos no disponible**: No se puede enviar el email de restablecimiento por indisponibilidad del servicio externo de envío.
	  ○ Notificar un error asociado al problema encontrado.

Notas	Dependiendo del proveedor del correo del usuario, el email de confirmación podría tardar más o menos en llegar, y ser o no identificado como spam, por lo que el usuario podría tener que buscarlo en la carpeta de spam de su correo.

5.5.3 Caso de Uso 3: Login

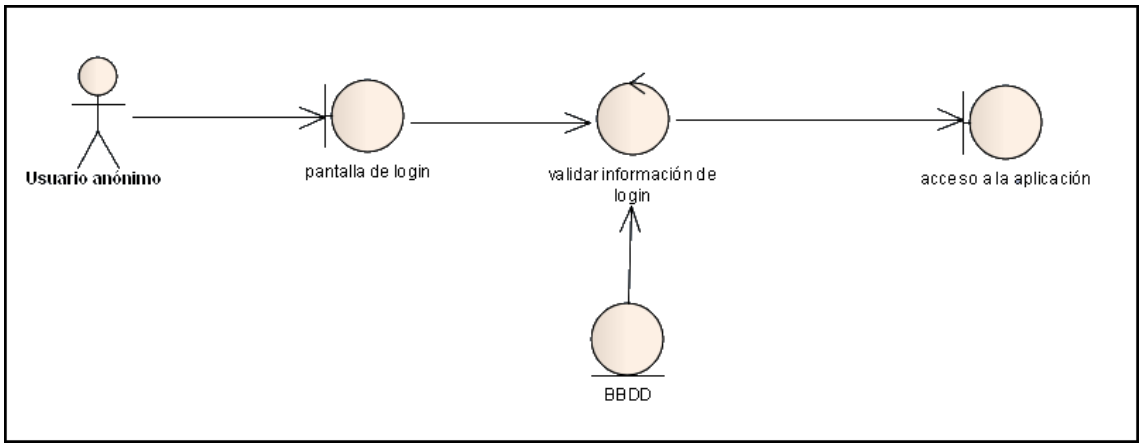


Figura 5.7. Diagrama de robustez del caso de uso Login

Login	
Precondiciones	• El usuario no está autenticado pero sí ha creado una cuenta en el sistema por lo que dispone de credenciales de usuario. • La cuenta de usuario ha de estar activada • El usuario no debe estar bloqueado
Poscondiciones	El formulario de login se ha completado correctamente.
Actores	Usuario anónimo, (se convierte en usuario registrado al finalizar)
Descripción	1. El usuario accederá a la página login.

	2. Rellenará el formulario con sus credenciales de usuario (dirección de correo y password). 3. Presiona el botón de iniciar sesión. 4. El usuario anónimo pasa a usuario registrado y accede a su listado de medios para poder empezar a utilizar la aplicación.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Las credenciales de usuario no son correctas.
	  ○ Se informa al usuario de que las credenciales introducidas no son correctas y se vuelve al paso 2
	• **Escenario Alternativo 2**: El usuario logueado es un administrador
	  ○ El usuario anónimo pasa a ser usuario administrador y es redirigido al listado de usuarios para poder ejercer sus funciones como administrador.

Excepciones	• **La base de datos no está disponible**: No se pueden consultar la existencia y adecuación de las credenciales de usuario en la base de datos por problemas de conexión. ○ Notificar un error asociado al problema encontrado.
Notas	-

5.5.4 Caso de Uso 4: Cambiar idioma

Cambiar idioma	
Precondiciones	-
Poscondiciones	El idioma ha cambiado según lo solicitado
Actores	Usuario anónimo / Usuario registrador / Usuario administrador
Descripción	1. El usuario accederá a la opción de cambio de idioma y seleccionará el que más le convenga. 2. La interfaz de la aplicación cambiará de idioma al seleccionado por el usuario, guardando en cookie la preferencia de usuario que se mantendrá durante la sesión, por lo que todas las páginas del sitio que visite a partir de ese momento se presentarán en el idioma que ha seleccionado
Variaciones (escenarios secundarios)	-
Excepciones	-
Notas	-

5.5.5 Caso de Uso 5: Descargar aplicación móvil

Descargar aplicación móvil	
Precondiciones	-
Poscondiciones	El usuario ha obtenido la aplicación móvil
Actores	Usuario anónimo / Usuario registrador / Usuario administrador
Descripción	1. El usuario accederá a la página principal de la aplicación con el navegador de su dispositivo móvil 2. Abrirá el enlace de descarga de la aplicación. 3. Descargará la aplicación. 4. Instalará la aplicación en su dispositivo móvil. 5. La aplicación está instalada y lista para configurar.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario descarga la aplicación en móvil en su pc. ○ La podrá copiar a su dispositivo móvil utilizando un medio de transferencia como un cable USB y podrá proceder con la instalación: paso 4.

Excepciones	• **El repositorio de la aplicación no está disponible**: No se puede acceder al repositorio de descarga de la aplicación porque éste no está disponible
	  ○ Notificar un error asociado al problema encontrado.
	• **El dispositivo móvill no es compatible con la aplicación**: El dispositivo no tiene la versión de Android necesaria para su instalación.
	  ○ Notificar un error asociado al problema encontrado.

Notas	-

5.5.6 Caso de Uso 6: Cambiar tema

Cambiar tema	
Precondiciones	-
Poscondiciones	El tema o skin de la aplicación ha cambiado según lo solicitado
Actores	Usuario anónimo / Usuario registrador / Usuario administrador
Descripción	1. El usuario accederá a la opción de cambio de tema y seleccionará el que más le convenga. 2. La interfaz de la aplicación cambiará al tema seleccionado por el usuario, guardando en cookie la preferencia de usuario que se mantendrá durante la sesión, por lo que todas las páginas del sitio que visite a partir de ese momento se presentarán con el tema que ha seleccionado
Variaciones (escenarios	-

secundarios)	
Excepciones	-
Notas	-

5.5.7 Caso de Uso 7: Gestionar medios

Gestionar medios	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario accederá a la página ver medios. 2. Utilizando el formulario de sugerencia busca el nombre del medio en que está interesado. 3. De entre los que se le presentan como resultados selecciona uno, cuyos datos son llevados automáticamente al formulario de inserción. 4. Hace clic en el botón guardar. 5. El nuevo medio se añade al final de la lista de medios.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario utiliza la página de añadir nuevo medio.
	  ○ El procedimiento es análogo al ya descrito, con la diferencia de que cuando el medio es añadido, se abre automáticamente la página del nuevo medio para visualizarlo.
	• **Escenario Alternativo 2**: El usuario introduce directamente los valores en el formulario de inserción.
	  ○ Se continúa al proceso general desde el paso 4.
	• **Escenario Alternativo 3**: El usuario quiere editar uno o más medios.
	  ○ El usuario hace clic en el botón de edición correspondiente al medio que quiere editar, o selecciona varios de ellos con en su correspondiente columna de selección y hace clic en el botón de editar superior a la tabla de medios.
	  ○ Edita los campos como crea conveniente
	  ○ Hace clic en el botón verde para guardar cambios que aparecerá junto al de editar.
	  ○ Los cambios se guardan y reflejan en la tabla
	• **Escenario Alternativo 4**: El usuario solicita eliminar uno o varios medios
	  ○ El usuario hace clic en el botón de borrar correspondiente al medio que quiere eliminar, o selecciona varios de ellos con en su correspondiente columna de selección y hace clic en el botón de borrar superior a la tabla de medios.
	  ○ Se pide confirmación de que desea eliminar el medio o medios seleccionados

	○ Se informa al usuario del éxito de la operación. • **Escenario Alternativo 5**: El usuario quiere revisar los medios disponibles. ○ Puede cambiar el número de medios mostrados, filtrar los resultados, o navegar entre las distintas páginas de resultados según el número de medios disponibles. ○ También puede buscar medios de un tipo determinado utilizando la opción de menú de buscar por tipo.
Excepciones	• **La base de datos no está disponible**: No se pueden acceder la base de datos por problemas de conexión. No se puede obtener información de los medios ni modificarlos ○ Notificar un error asociado al problema encontrado.
Notas	-

5.5.8 Caso de Uso 8: Gestionar listas de reproducción

Gestionar listas de reproducción	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario accederá a la página de ver listas de reproducción. 2. Introduce el nombre para una nueva lista. 3. Hace clic en el botón guardar. 4. La nueva lista de reproducción se añade al final de la tabla de listas.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario utiliza la página de añadir nueva lista. ○ El procedimiento es análogo al ya descrito, con la diferencia de que cuando el medio es añadido, se abre automáticamente la página la nueva lista para visualizarla y poder añadirle medios. • **Escenario Alternativo 2**: El usuario quiere editar una o más listas. ○ El usuario hace clic en el botón de edición correspondiente la lista que quiere editar, o selecciona varias de ellas con en su correspondiente columna de selección y hace clic en el botón de editar superior a la tabla. ○ Edita los campos como crea conveniente. ○ Hace clic en el botón verde para guardar cambios que aparecerá junto al de editar. ○ Los cambios se guardan y reflejan en la tabla.

	• **Escenario Alternativo 2**: El usuario solicita eliminar una o varias listas de reproducción.
	  ○ El usuario hace clic en el botón de borrar correspondiente a la lista que quiere eliminar, o selecciona varias de ellas con en su correspondiente columna de selección y hace clic en el botón de borrar superior a la tabla.
	  ○ Se pide confirmación de que desea eliminar.
	  ○ Se informa al usuario del éxito de la operación.
	• **Escenario Alternativo 3**: El usuario quiere revisar las listas disponibles.
	  ○ Puede cambiar el número listas mostradas, filtrar los resultados, o navegar entre las distintas páginas de resultados según el número de listas disponibles.

Excepciones	• **La base de datos no está disponible**: No se pueden acceder la base de datos por problemas de conexión. No se puede obtener información de las listas ni modificarlas ○ Notificar un error asociado al problema encontrado.
Notas	-

5.5.9 Cao de Uso 9: Gestionar elementos de una lista de reproducción.

Gestionar elementos de una lista de reproducción	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario accederá a la página de una lista de reproducción concreta. 2. Seleccionará un medio para añadir a la lista. 3. Hace clic en el botón añadir. 4. La el nuevo elemento de la lista de reproducción se añade al final de ella.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario quiere editar una o más elementos de la lista.
	  ○ El usuario hace clic en el botón de edición correspondiente del elemento que quiere editar, o selecciona varios con en su correspondiente columna de selección y hace clic en el botón de editar superior a la tabla.
	  ○ Edita los campos como crea conveniente.
	  ○ Hace clic en el botón verde para guardar cambios que aparecerá junto al de editar.

	○ Los cambios se guardan y reflejan en la tabla. • **Escenario Alternativo 2**: El usuario solicita eliminar uno o varios elementos de la lista. ○ El usuario hace clic en el botón de borrar correspondiente al elemento que quiere eliminar, o selecciona varios con en su correspondiente columna de selección y hace clic en el botón de borrar superior a la tabla. ○ Se pide confirmación de que desea eliminar. ○ Se informa al usuario del éxito de la operación.
Excepciones	• **La base de datos no está disponible**: No se pueden acceder la base de datos por problemas de conexión. No se puede obtener información de la lista ni modificar sus elementos. ○ Notificar un error asociado al problema encontrado.
Notas	-

5.5.10 Caso de Uso 10: Reproducir

Reproducir	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario accederá a la página de un medio o lista de reproducción concreta. 2. Inicia la reproducción con el botón “play”. 3. Utiliza los controles de reproducción para controlar la reproducción según sus necesidades.
Variaciones (escenarios secundarios)	-
Excepciones	• **La base de datos no está disponible**: No se pueden acceder a la información del medio o lista de reproducción en la base de datos. ○ Notificar un error asociado al problema encontrado. • **Recurso externo no disponible**: No se puede acceder al recurso remoto. ○ Notificar un error asociado al problema encontrado. • **Otros errores de reproducción**: Erres de formato o relacionados con el plugin de reproducción web utilizado. ○ Notificar un error asociado al problema encontrado.
Notas	Según el origen, tamaño del video o audio, velocidad de conexión u otros parámetros la reproducción puede demorar más o menos en cargar.

5.5.11 Caso de Uso 11: Cambiar contraseña

Cambiar contraseña	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	La contraseña del usuario ha cambiado correctamente.
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario accederá a la opción de menú de cambiar password. 2. Rellenará la información necesaria para completar el formulario: contraseña actual y contraseña nueva por duplicado, para evitar errores fortuitos de escritura. 3. Si los datos con correctos se cambia la contraseña del usuario y se notifica el éxito de la operación. 4.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Los datos no han sido rellenados correctamente, o de manera completa. ○ Se vuelve al paso 2 e informa al usuario de los errores cometidos en el formulario o datos que falta por rellenar.
Excepciones	• **La base de datos no está disponible**: No se pueden acceder a la base de datos por problemas de conexión. ○ Notificar un error asociado al problema encontrado.
Notas	El usuario administrador por defecto presente en la aplicación no está guardado en la base de datos por lo que su contraseña no puede cambiarse por este procedimiento. Cualquier otro administrador creado y guardado en base de datos sí puede.

5.5.12 Caso de Uso 12: Cerrar sesión

Cerrar sesión	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	La sesión de usuario ha finalizado correctamente
Actores	Usuario registrado / Usuario administrador. (se convierte en usuario anónimo al finalizar)
Descripción	1. El usuario selecciona la opción de menú de cerrar sesión 2. La sesión finaliza y es devuelto a la página principal de la aplicación como usuario anónimo.
Variaciones	-

(escenarios secundarios)	
Excepciones	-
Notas	-

5.5.13 Caso de Uso 13: Consultar ayuda

Consultar ayuda	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario accederá a la página de ayuda de la aplicación
Variaciones (escenarios secundarios)	-
Excepciones	-
Notas	-

5.5.14 Caso de Uso 14: Gestionar usuarios

Gestionar usuarios	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario administrador.
Descripción	1. El usuario administrador accederá al listado de usuarios donde podrá ver sus datos, y estados de activación y bloqueo. 2. Seleccionará un usuario y hará clic en el botón de editar 3. Editará los datos que crea convenientes en el formulario de edición 4. Guardará los cambios, lo cual supondrá la vuelta al listado de usuarios con los datos actualizados.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El administrador desea añadir un nuevo usuario manualmente. ○ Accede a la opción de menú para crear un nuevo usuario. ○ Rellena el formulario con los datos solicitados, activándolo y manteniéndolo desbloqueado si desea que la cuenta pase a estar disponible de inmediato. ○ Hará clic en guardar ○ Si todo está correcto se creará y guardará el nuevo usuario. Si no, se volverá al formulario donde se señalarán las incorrecciones a corregir para poder completar la operación.

	• **Escenario Alternativo 2**: El administrador quiere eliminar un usuario.
	  ○ Selecciona el usuario o usuarios a eliminar y hace clic en el botón borrar.
	  ○ Se pide confirmación del borrado.
	  ○ Se notifica al usuario el éxito de la operación.

Excepciones	• **La base de datos no está disponible**: No se pueden guardar las modificaciones de usuarios en la base de datos por problemas de conexión. ○ Notificar un error asociado al problema encontrado.
Notas	No se pueden modificar los datos del administrador por defecto puesto que no es un usuario guardado en la base de datos.

5.5.15 Caso de Uso 15: Gestionar roles

Gestionar roles	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario administrador.
Descripción	1. El usuario administrador accederá al listado de roles de usuario donde podrá ver los roles asociados a cada usuario. 2. Seleccionará un usuario y hará clic en el botón de editar 3. Editará los datos que crea convenientes en el formulario de edición asignando al usuario el rol que desee. 4. Guardará los cambios, lo cual supondrá la vuelta al listado de roles de usuario con los datos actualizados.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El administrador quiere eliminar un rol de un usuario.
	  ○ Selecciona el rol o roles a eliminar y hace clic en el botón borrar.
	  ○ Se pide confirmación del borrado.
	  ○ Se notifica al el éxito de la operación.

Excepciones	• **La base de datos no está disponible**: No se pueden guardar los cambios en la base de datos por problemas de conexión. ○ Notificar un error asociado al problema encontrado.
Notas	No se puede modificar el rol de usuario del administrador por defecto ya que no está guardado en la base de datos.

5.5.16 Caso de Uso 16: Gestionar asignaciones medio-lista

Registro	
Precondiciones	El usuario está registrado y ha iniciado sesión.
Poscondiciones	-
Actores	Usuario administrador.
Descripción	1. El usuario administrador accederá al listado de asignaciones medio-lista donde podrá ver qué medios pertenecen a cada lista de reproducción así como el usuario propietario de los mismos. 2. Seleccionará un elemento o conjunto de ellos y hará clic en el botón de editar 3. Editará los datos que crea convenientes y hará clic en el botón guardar. 4. Guardará los cambios que se reflejarán en la tabla.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El administrador quiere eliminar un elemento ○ Selecciona un elemento o elementos a eliminar y hace clic en el botón borrar. ○ Se pide confirmación del borrado. ○ Se notifica al el éxito de la operación.
Excepciones	• **La base de datos no está disponible**: No se pueden guardarlos cambios en la base de datos por problemas de conexión. ○ Notificar un error asociado al problema encontrado.
Notas	-

5.5.17 Caso de Uso 17: Configurar preferencias de control remoto

Configurar preferencias de control remoto	
Precondiciones	• El usuario tiene la aplicación móvil descargada e instalada en su dispositivo móvil • El usuario tiene una cuenta creada en el sistema.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario abrirá la aplicación móvil de control remoto en su dispositivo móvil. 2. Accederá al menú y seleccionará la opción de configuración. 3. Se le mostrará la configuración actual, con sus datos e credenciales y dirección del servidor al que se enviarán las

	órdenes de control remoto 4. Seleccionando el campo que desea editar podrá modificar y guardar su contenido con los datos correspondientes a su cuenta. 5. Una vez configuradas sus credenciales y dirección del servidor la aplicación quedará preparada para actuar como control remoto de los reproductores web abiertos con su cuenta en la página web de la aplicación.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Los datos no han sido rellenados correctamente, o de manera completa. ○ Los datos se guardarán pero las órdenes remotas no se podrán transmitir correctamente. ○ Se mostrará el error correspondiente.
Excepciones	-
Notas	-

5.5.18 Caso de Uso 18: Controlar reproducción remota

Controlar reproducción remota	
Precondiciones	• El usuario está registrado y ha iniciado sesión en la aplicación web. • El usuario tiene instalada y configurada correctamente la aplicación en su dispositivo móvil.
Poscondiciones	-
Actores	Usuario registrado / Usuario administrador.
Descripción	1. El usuario abrirá la aplicación móvil de control remoto en su dispositivo móvil. 2. El usuario abre un medio o lista de reproducción para su visualización. 3. Inicia la reproducción con el botón “play” en su aplicación móvil. 4. Utiliza los controles de reproducción para controlar la reproducción según sus necesidades: volumen, progreso, pausa, parada, avance al siguiente medio, retroceso, o cambio a pantalla completa. 5. La aplicación móvil reflejará en la pantalla el último mensaje que ha recibido correctamente el servidor. 6. La respuesta a las órdenes se verá reflejada en el reproductor abierto en el navegador web del ordenador.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El usuario tiene abiertos varios reproductores web en con su cuenta en distintos ordenadores y ubicaciones. ○ Las órdenes remotas se transmitirán a todos ellos

	sincronizándolos.
	• **Escenario Alternativo 2**: Los datos no han sido rellenados correctamente, o de manera completa.
	  ○ Los datos se guardarán pero las órdenes remotas no se podrán transmitir correctamente.
	  ○ Se mostrará el error correspondiente.

Excepciones	• **La base de datos no está disponible**: No se puede comprobar que el usuario que envía las órdenes remotas existe en la base de datos por problemas de conexión.
	  ○ Notificar un error asociado al problema encontrado.
	• **Fallo de conexión del dispositivo móvil**: No se puede enviar el las ordenes remotas porque el dispositivo móvil se encuentra sin conexión.
	  ○ Notificar un error asociado al problema encontrado.

Notas	-

5.6 Análisis de Interfaces de Usuario

5.6.1 Descripción de la Interfaz

En este apartado se muestran de manera esquemática las distintas interfaces de usuario del sistema y la organización que se dará a los elementos que las componen.

La interfaz se puede descomponer en dos partes diferenciadas, por un lado la interfaz de la aplicación web que se visualizará dentro de un navegador web, y por otra parte la interfaz de la aplicación Android de control remoto que se visualizará como una aplicación nativa en el dispositivo móvil.

5.6.1.1 Interfaz Web

La pantalla principal de la aplicación puede ser un buen ejemplo para ilustrar la organización general de elementos en la aplicación web:

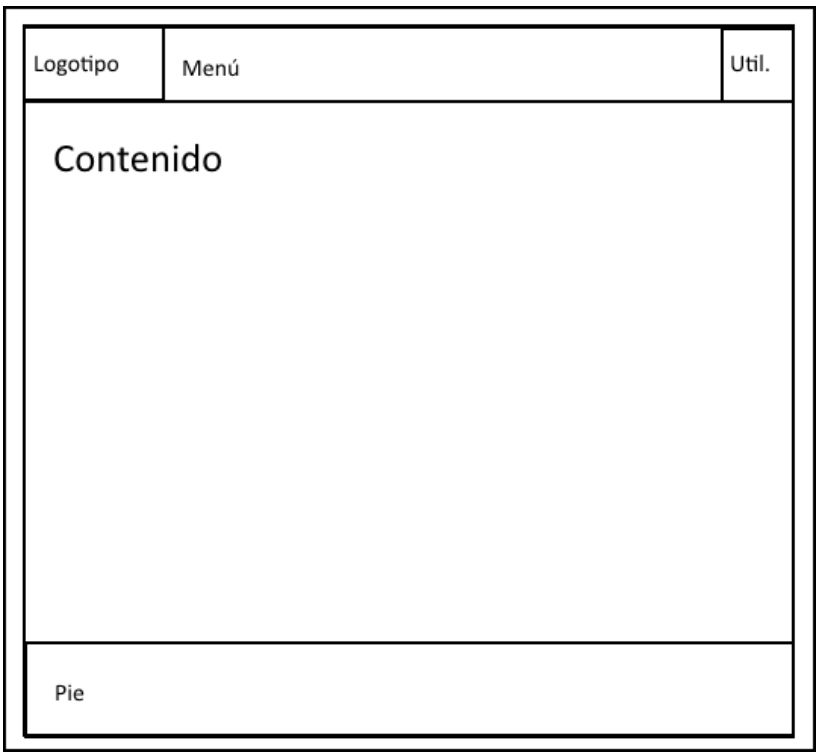


Figura 5.8. Boceto de la pantalla principal de la aplicación

Nos encontramos ante una organización bastante típica de los elementos de una aplicación web, con una cabecera donde se situará el menú, precedido por el logotipo de la aplicación a la derecha que dará siempre acceso a la página principal, y sucedido de una sección de utilidades en la esquina superior izquierda, que agrupará opciones de configuración.

Tras esta cabecera se encontrará el contenido de la página, de carácter informativo por ejemplo en esta página principal, pero con otras organizaciones internas en el caso de otras páginas de la aplicación.

Para concluir con la página se presenta un pie, que incluirá dirección de contacto, fecha de última versión, e información sobre el framework utilizado.

Las opciones de menú desplegable variarán en función del estado de autenticación y rol de usuario, teniendo solo opción a iniciar sesión, si se trata de un usuario anónimo, opciones de manejo de medios y listas para un usuario registrado, y todo el resto de opciones de administración de usuarios , roles, asignaciones medio-lista si se trata de un usuario administrador.

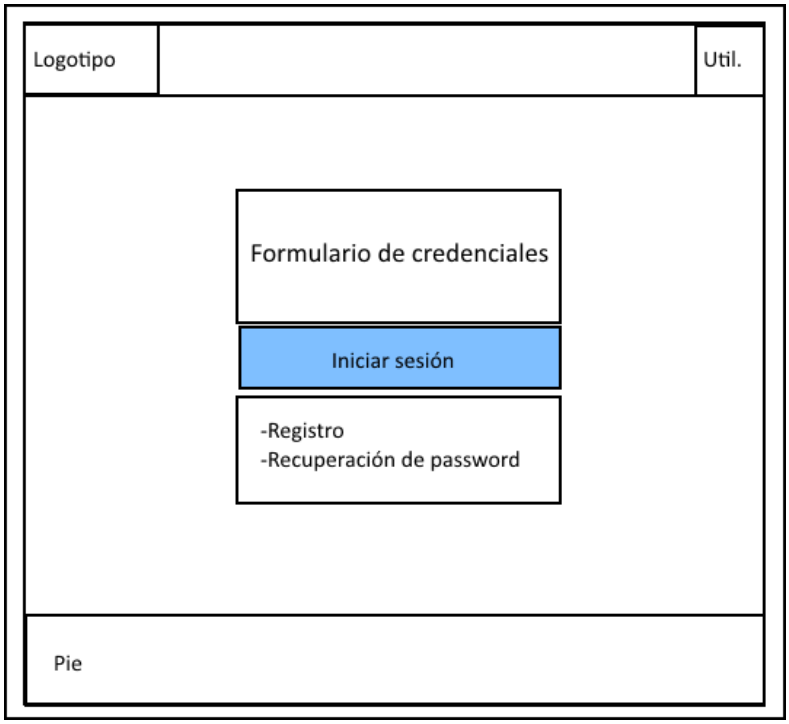


Figura 5.9. Boceto de la página de login de la aplicación

Como vemos en el esquema anterior correspondiente a la pantalla de inicio de sesión, se mantiene el esquema básico de la pantalla principal que hemos vistos anteriormente, pero el área de contenido se compone en esta ocasión de un formulario en el que se piden las credenciales, y una pequeña sesión subyacente a este con opciones adicionales para el registro de nuevos usuario o el restablecimiento de password para usuarios que hayan olvidado su contraseña.

El menú principal en este caso carece de opciones puesto que el inicio de sesión es la única acción disponible, y ya se encuentra como contenido de la página.

Una vez iniciada sesión, un usuario será enviado a la página de listado de medios, donde podrá revisar los medios que ya ha añadido o añadir nuevos, y cuya interfaz se organiza de la siguiente manera:

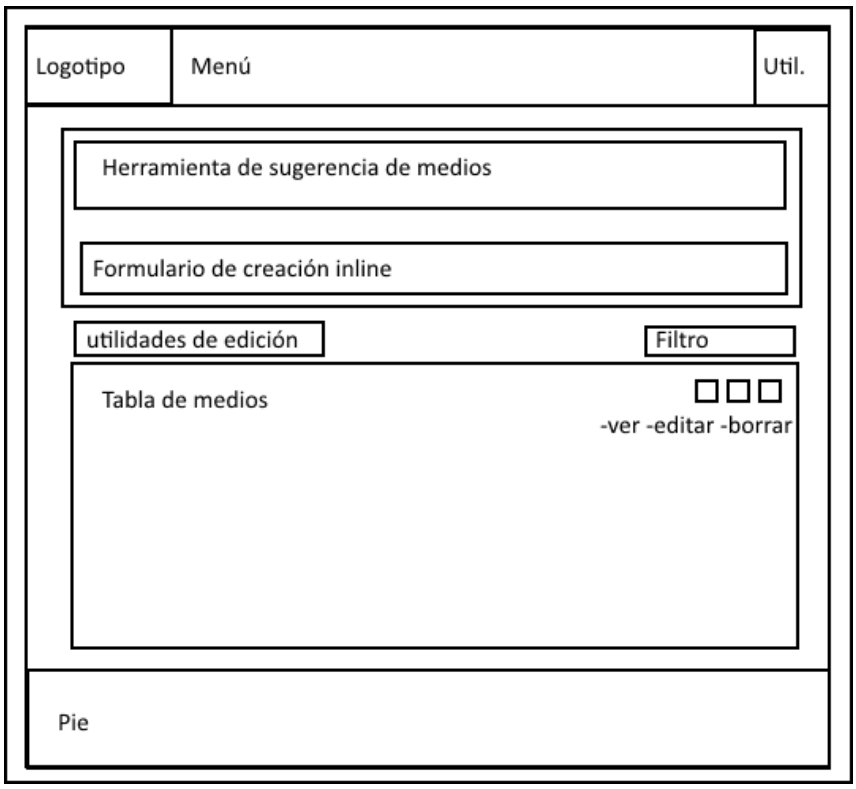


Figura 5.10 Boceto de la página de creación de medios

La organización general se conserva como siempre en esta página de creación de medios, pero en la sección de contenido nos encontramos primero con un formulario para la creación directa de nuevos medios, compuesto a su vez por un campo de sugerencia, que podrá rellenar automáticamente todos los campos del formulario de creación.

Al hacer una búsqueda en esta herramienta de sugerencia, se mostrará un overlay o pantalla superpuesta con los resultados de la búsqueda entre los que el usuario seleccionará el que prefiera y sus datos rellenarán el formulario de creación.

Tras esta sección de creación nos encontramos con la tabla de medios, que mostrará todos los medios que el usuario ha creado, paginados si son demasiados para mostrar en una sola página.

A la derecha de cada fila de la tabla habrá unas opciones para visualización, edición y borrado de cada medio, selección de número de elementos a mostrar etc. A la derecha habrá una columna de selección lo cual permitirá seleccionar varias filas para edición o borrado múltiple.

Sobre esta tabla habrá unas utilidades de filtrado, edición, cabeceras para el reordenado de los elementos.

Cuando el usuario seleccione visualizar un medio accederá a la página de visualización estructurada de la siguiente manera:

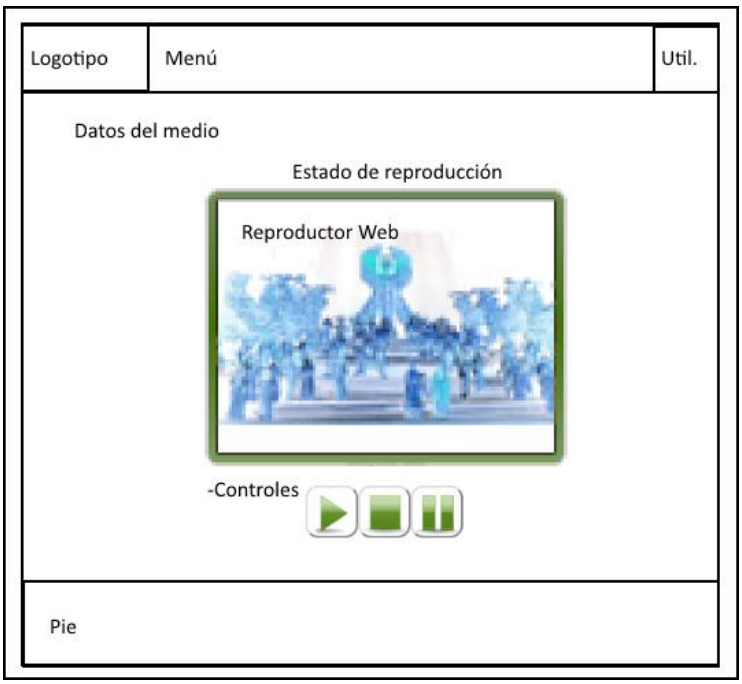


Figura 5.11 Boceto de la página de visualización de medios

La organización de la interfaz para el listado de listas de reproducción es muy similar a la de listado de medios, salvo por la ausencia de las herramientas de sugerencia.

Cuando se selecciona una lista para reproducir, se muestra en una página como la siguiente.

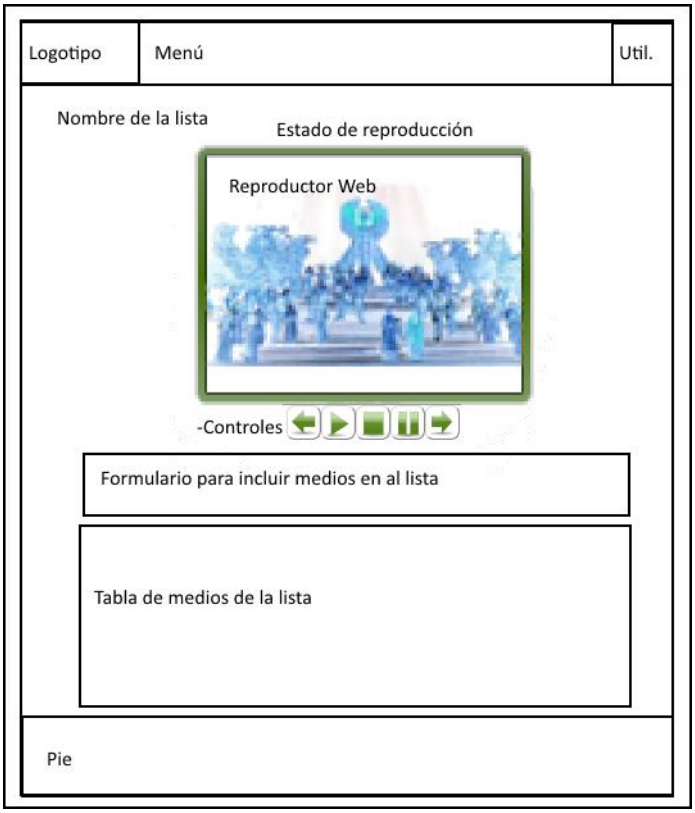


Figura 5.12 Boceto de la página de visualización de listas de reproducción

En esta página se puede además de reproducir la lista concreta, añadirle nuevos elementos, de entre los medios ya creados, modificarlos y/o eliminar su inclusión en la lista.

La creación y edición de todas las entidades de la aplicación, tanto medios, como listas de reproducción para un usuario normal, como la creación de nuevos usuarios, etc. para usuarios administradores se pueden hacer desde las páginas de creación que responden al siguiente esquema:

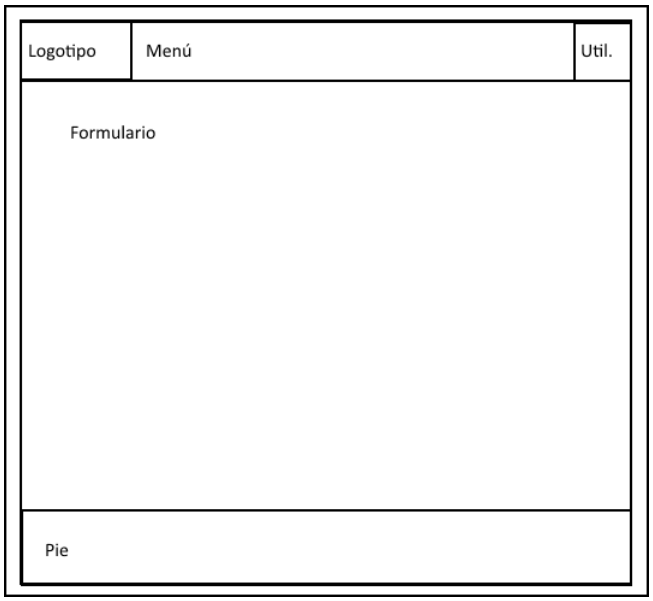


Figura 5.12 Boceto páginas de creación y edición

Las páginas de registro, recuperación de contraseña, cambio de password, etc. también se corresponden con este esquema.

Los otros listados no detallados previamente se corresponden al siguiente esquema:

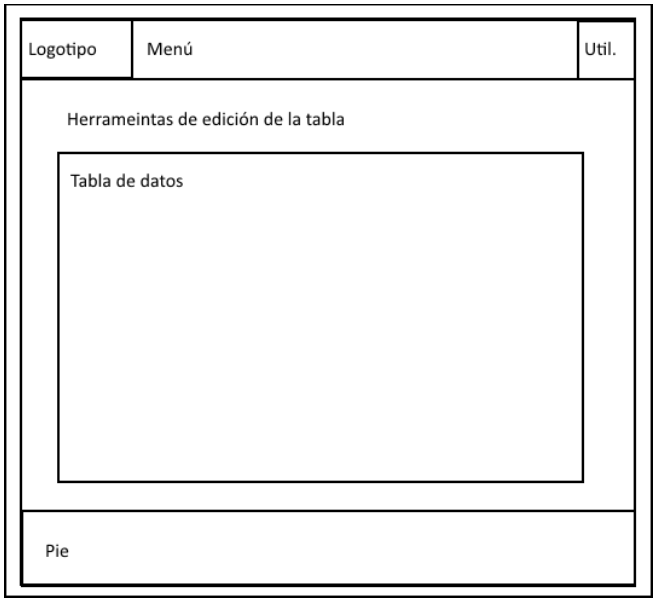


Figura 5.13 Boceto de la páginas de tablas de listado de elementos

5.6.1.2 Interfaz de la aplicación móvil

La aplicación móvil cuenta con una sencilla interfaz compuesta por solo dos pantallas, una de las cuales acumula todos los controles de reproducción y se estructura de la siguiente manera:

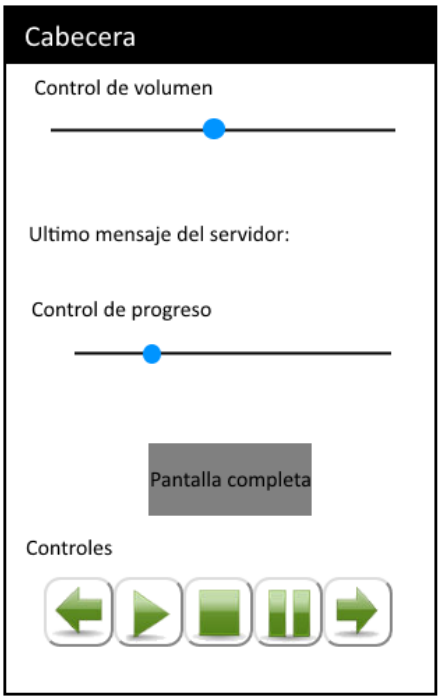


Figura 5.13 Boceto de la pantalla principal del control remoto móvil

La segunda es una pantalla de configuración para poder establecer correctamente los datos de conexión al servidor.

Figura 5.14 Boceto de la pantalla de configuración del control remoto móvil

5.6.2 Descripción del Comportamiento de la Interfaz

En este apartado debemos especificar cosas como los convenios que vamos a crear para validar la entrada de datos de la aplicación, los mensajes de error que mostraremos y el tipo de ayuda que vamos a proporcionar al usuario.

La selección de componentes que se hace para la recogida de los datos que debe introducir el usuario se ha hecho pensando en minimizar las posibilidades de equivocación del mismo, pues ofrecen un número limitado de opciones que son válidas.

No obstante algunos formularios requieren valores determinados, como en el caso de confirmación de contraseña, que debe coincidir con la contraseña introducida previamente, o nombres con una longitud máxima o mínima determinada, formato de una dirección de correo, o simplemente campos que no pueden dejarse en blanco. Por ellos cuando el usuario intente enviar un formulario cuyos valores no sean los adecuados, se le notificará con mensajes de error cercanos al campo del formulario que deba corregir o completar.

Para ilustrar ese tipo de notificaciones se presenta el siguiente ejemplo:

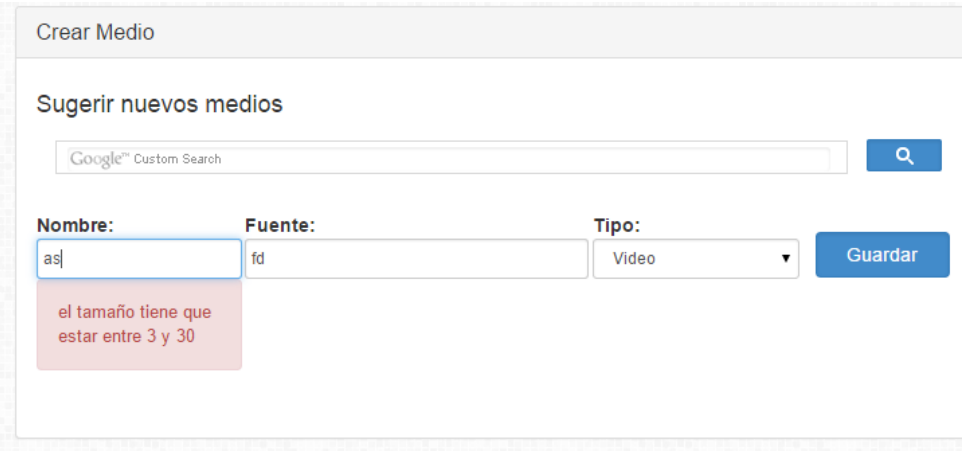


Figura 5.15 Ejemplo de notificación de error en un formulario

Otro tipo de notificaciones pop-up son ofrecidas si se producen errores de conexión con la base de datos u otro tipo.

La aplicación también cuenta con páginas de error específicas cuando se solicitan recursos no existentes, se trata de acceder a partes de la aplicación de administración para las que no se dispone de privilegios suficientes, fallos de datos o errores desconocidos.

Figura 5.16 Ejemplo de notificación de error al solicitar un recurso no existente

Los usuarios registrados también cuentan con una opción de ayuda en el menú en el apartado de utilidades con información útil acerca de problemas de configuración.

En el caso de la aplicación móvil, se presenta un campo con información sobre la última orden recibida y transmitida por el servidor, que informa de un posible error de conexión si este ha tenido lugar.

5.6.3 Diagrama de Navegabilidad

En esta sección incluiremos diagramas que muestren la navegación que habrá entre las pantallas del programa y su relación con las computaciones que tienen lugar en las mismas.

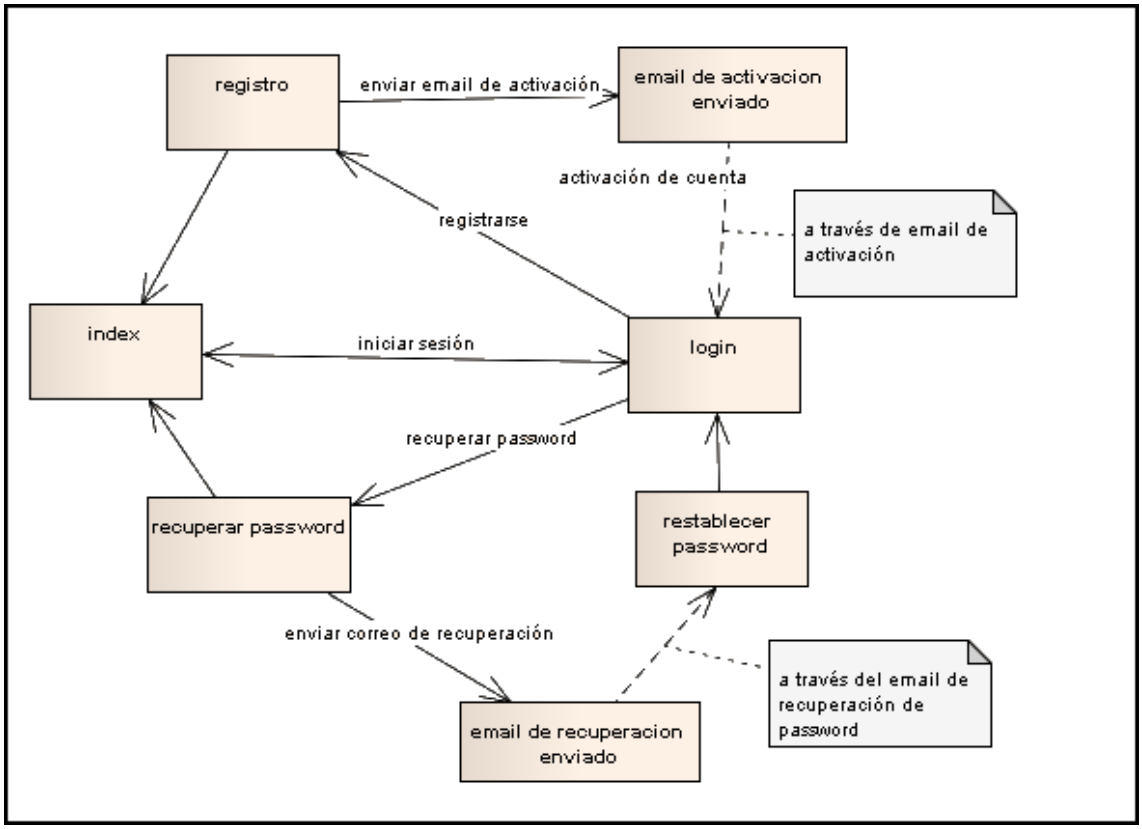


Figura 5.17 Diagrama de navegabilidad para usuario anónimo

Como vemos en el diagrama anterior el usuario accede inicialmente por el index o página principal, y desde ahí va hacia la página de login para iniciar sesión. Si no tiene una cuenta puede crear una, y posteriormente activarla mediante el email de activación, tras lo cual llegará de nuevo a la página de login, pero esta vez con una cuenta disponible para iniciar sesión.

Si el usuario ha olvidado la password puede solicitar recuperarla, lo cual hace que se envíe un email que dirige al usuario a un formulario de restablecimiento de una nueva contraseña, y tras ello al login para poder iniciar sesión con ella.

Una vez iniciada sesión el usuario accede al listado de medios, si se trata de un usuario normal, o al listado de usuarios si se trata de un usuario administrador.

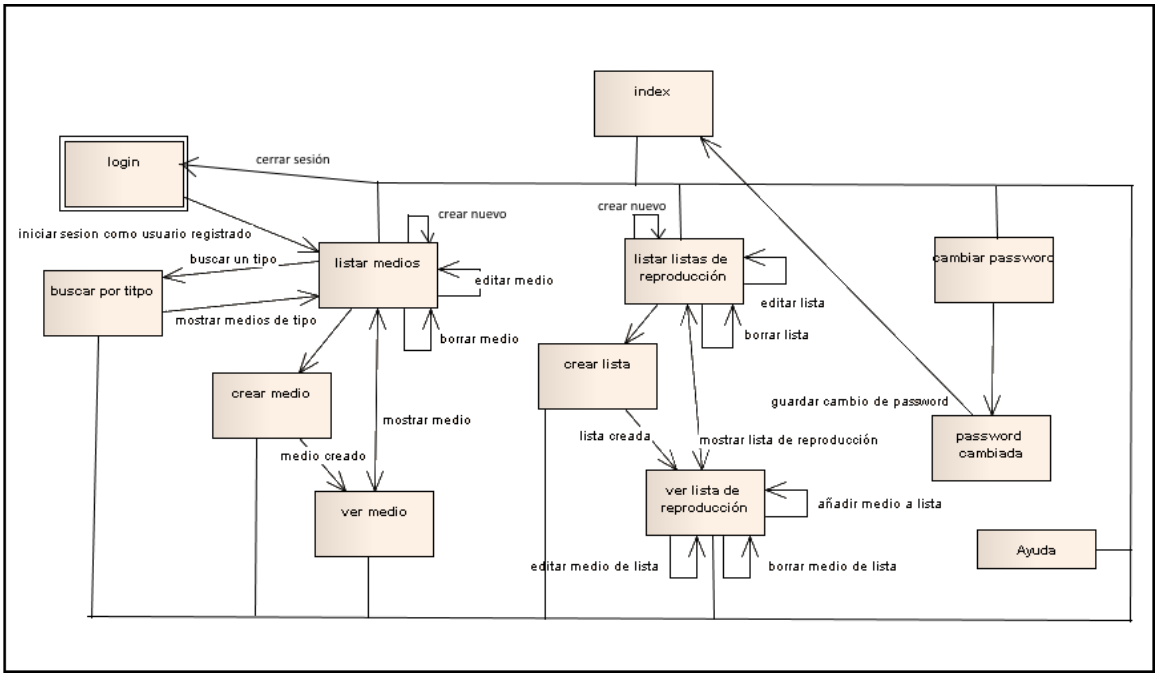


Figura 5.18 Diagrama de navegabilidad para usuario registrado

Como vemos en el diagrama anterior el usuario registrado accede al listado de medios desde el que puede manejas su creación, edición borrado, acceder a visualización, etc., así como manejar las listas de reproducción.

Todas las páginas de listado, además de la de búsqueda por tipo y cambio de password son accesibles desde el menú principal, y la página principal es accesible desde todas las páginas.

Las páginas de visualización de medios y listas son accesibles desde sus listados correspondientes.

Al cerrar sesión desde cualquier página el usuario es devuelto a la página de login como usuario anónimo.

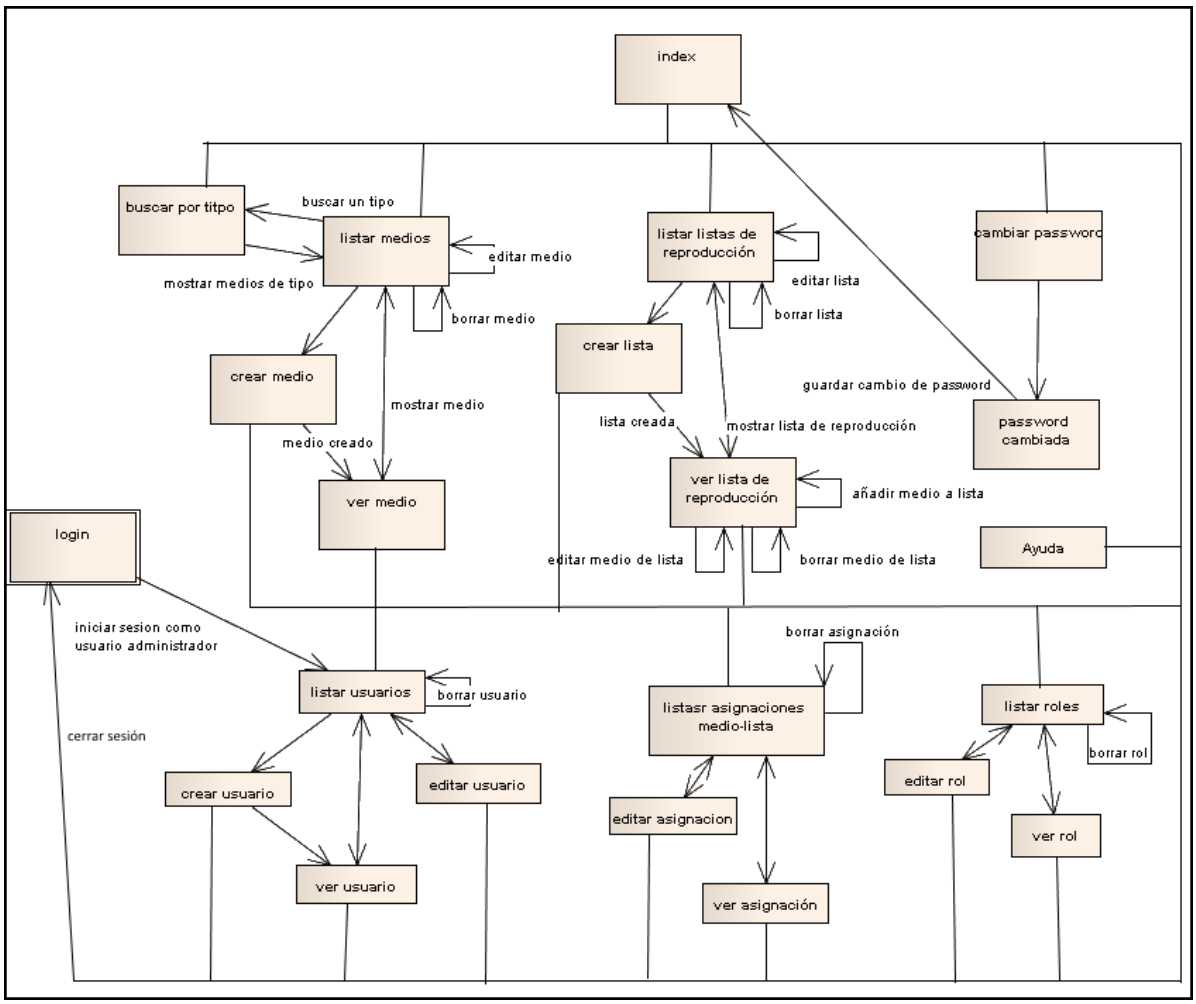


Figura 5.19 Diagrama de navegabilidad para usuario administrador

Como se puede ver en el diagrama anterior los usuarios administradores son dirigidos al listado de usuarios tras su proceso de login, y disponen además de todas las opciones de usuarios standard, de una navegabilidad extra como administradores pudiendo acceder a las páginas de administración de usuarios, asignaciones de lista y roles.

Al igual que los otros usuarios, al cerrar sesión son devueltos a la página de login como usuarios anónimos.

La navegabilidad en la aplicación de control es muy sencilla constando solo de dos pantallas, una principal de control, y otra de configuración, como se ve en el siguiente diagrama.

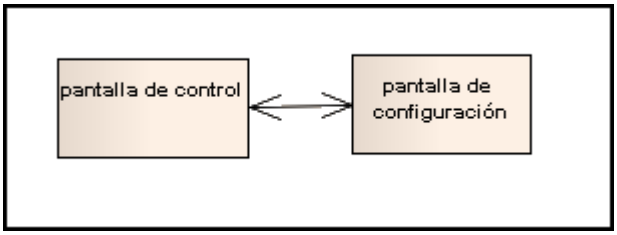


Figura 5.20 Diagrama de navegabilidad de aplicación de control remoto

5.7 Especificación del Plan de Pruebas

En esta sección crearemos y diseñaremos el plan de pruebas de la aplicación y sus funciones, así como todos los mecanismos que utilizaremos para detectar errores y corregirlos ya en la fase de implementación.

Las pruebas contemplarán aspectos tanto de funcionalidad de la aplicación como de aspectos de los usuarios o clientes de la misma.

Como uno de los objetivos de las pruebas es asegurar que se cumplan las funciones del programa, organizaremos las pruebas a realizar en función de los casos de uso.

Caso de Uso 1: Registro	
Prueba	**Resultado Esperado**
Rellenar datos correctamente	El sistema los guarda correctamente, se notifica el éxito de la operación y se envía el email de activación.
Prueba	**Resultado Esperado**
Introducir un email no válido	El sistema notifica que la dirección introducida no es correcta y solicita introducirla de nuevo.
Prueba	**Resultado Esperado**
Introducir un email ya registrado	El sistema notifica que la dirección introducida no está disponible y solicita introducirla de nuevo.
Prueba	**Resultado Esperado**
Introducir una contraseña distinta en el campo el confirmación de contraseña	El sistema notifica que las contraseñas no coinciden.
Prueba	**Resultado Esperado**
Dejar campos obligatorios sin rellenar	El sistema notifica que han quedado campos en blanco y deben rellenarse.
Prueba	**Resultado Esperado**
Transcribir el recaptcha incorrectamente	El sistema notifica el error y carga un nuevo recaptcha.

Caso de Uso 2: Recuperar contraseña	
Prueba	**Resultado Esperado**
Introducir correctamente la dirección del correo de recuperación	El sistema envía correctamente el email de restablecimiento de contraseña.
Prueba	**Resultado Esperado**
Introducir una dirección incorrecta	El sistema advierte de que no se ha encontrado ninguna cuenta asociada a dicha dirección de correo.
Prueba	**Resultado Esperado**
En el formulario de restablecimiento introducir una contraseña distinta en el campo el confirmación de contraseña	El sistema notifica que las contraseñas no coinciden.

Prueba	**Resultado Esperado**
En el formulario de restablecimiento introducir correctamente los datos	La password es restablecida correctamente.

Caso de Uso 3: Login	
Prueba	**Resultado Esperado**
Introducir las credenciales correctas como usuario standard	El sistema redirige al usuario al listado de medios.
Prueba	**Resultado Esperado**
Introducir las credenciales correctas como usuario administrador	El sistema redirige al administrador al listado de usuarios.
Prueba	**Resultado Esperado**
Introducir unas credenciales incorrectas	Se notifica que las credenciales no son válidas.
Prueba	**Resultado Esperado**
Introducir unas credenciales correctas con un usuario bloqueado o inactivo	Se notifica el estado de bloqueo o falta de activación y se deniega el acceso.

Caso de Uso 4: Cambiar idioma	
Prueba	**Resultado Esperado**
Solicitar un cambio de idioma	El idioma de la interfaz cambia y se mantiene correctamente en las siguientes peticiones.

Caso de Uso 5: Descargar aplicación móvil	
Prueba	**Resultado Esperado**
Acceder al enlace de descarga de la aplicación	Funciona el enlace y se puede descargar la aplicación correctamente.

Caso de Uso 6: Cambiar tema	
Prueba	**Resultado Esperado**
Solicitar un cambio de tema	El tema de la interfaz cambia y se mantiene correctamente en las siguientes peticiones.

Caso de Uso 7: Gestionar medios	
Prueba	**Resultado Esperado**
Crear un nuevo medio introduciendo nombre y ubicación, desde la página de listado.	El medio se crea y añade correctamente a la lista.
Prueba	**Resultado Esperado**
Crear un nuevo medio utilizando la página de creación	El medio se crea y a continuación se muestra.

Prueba	**Resultado Esperado**
Crear un nuevo medio utilizando la herramienta de sugerencia	El formulario se autocompleta correctamente y el medio se añade de manera correcta.
Prueba	**Resultado Esperado**
Editar un medio	Los cambios se guardan correctamente.
Prueba	**Resultado Esperado**
Borrar uno o más medios	Los medios se borran correctamente.
Prueba	**Resultado Esperado**
Cancelar la edición o borrado	La operación se cancela correctamente.
Prueba	**Resultado Esperado**
Buscar medios por tipo	Se muestra el listado de medios sólo del tipo especificado.
Prueba	**Resultado Esperado**
Filtrar medios	Se muestra el listado de medios pero sólo aquellos que contienen la cadena de texto especificada en el filtro en alguno de sus campos.
Prueba	**Resultado Esperado**
Cambiar número de registros a mostrar	Se cambia correctamente el número máximo de medios que se muestran en cada página.

Caso de Uso 8: Gestionar listas de reproducción	
Prueba	**Resultado Esperado**
Crear una nueva lista de reproducción desde la página de listado	Se crea y se añade correctamente a la tabla de listas.
Prueba	**Resultado Esperado**
Crear una nueva lista utilizando la página de creación	La lista se crea y a continuación se muestra.
Prueba	**Resultado Esperado**
Editar lista	Los cambios se guardan correctamente.
Prueba	**Resultado Esperado**
Borrar una o más listas	Se borran correctamente.
Prueba	**Resultado Esperado**
Cancelar la edición o borrado	La operación se cancela correctamente.
Prueba	**Resultado Esperado**
Filtrar listas	Se muestra el listado de elementos que contienen la cadena de texto especificada en el filtro en alguno de sus campos.
Prueba	**Resultado Esperado**
Cambiar número de registros a mostrar	Se cambia correctamente el número máximo de listas que se muestran en cada página.

Caso de Uso 9: Gestionar elementos de una lista de reproducción	
Prueba	**Resultado Esperado**
Añadir un medio a la lista de reproducción	El medio se añade correctamente.
Prueba	**Resultado Esperado**
Editar un medio de la lista	Las modificaciones se guardan correctamente.
Prueba	**Resultado Esperado**

Eliminar un medio de la lista	El medio se elimina de la lista.
Prueba	**Resultado Esperado**
Cancelar la edición o borrado	La operación se cancela correctamente.

Caso de Uso 10: Reproducir	
Prueba	**Resultado Esperado**
Reproducir un medio	El medio se reproduce correctamente.
Prueba	**Resultado Esperado**
Reproducir una lista	Se reproduce correctamente.
Prueba	**Resultado Esperado**
Reproducir un medio con una fuente errónea o no disponible	El medio no se reproduce y se notifica el error.
Prueba	**Resultado Esperado**
Reproducir una lista vacía	No se muestra el reproductor y solo permite añadir medios a la lista.
Prueba	**Resultado Esperado**
Avanzar o retroceder de medio en una lista	Se cambia de medio correctamente.
Prueba	**Resultado Esperado**
Utilizar varios reproductores simultáneamente con la misma cuenta	La reproducción se sincroniza compartiendo las órdenes que reciben y envían .
Prueba	**Resultado Esperado**
Parar reproducción	La reproducción se para.
Prueba	**Resultado Esperado**
Pausar reproducción	La reproducción se pausa.

Caso de Uso 11: Cambiar contraseña	
Prueba	**Resultado Esperado**
Rellenar el formulario correctamente	Se cambia la contraseña y se notifica el éxito de la operación.
Prueba	**Resultado Esperado**
Introducir incorrectamente la contraseña actual	Se notifica el error y se pide introducirla correctamente.
Prueba	**Resultado Esperado**
Introducir nuevas contraseñas que no coincidan	Se notifica el error y se pide que coincidan.

Caso de Uso 12: Cerrar sesión	
Prueba	**Resultado Esperado**
Cerrar sesión	La sesión finaliza correctamente, se redirige a la página de login como usuario anónimo y el menú cambia con las opciones disponibles en ese momento.

Caso de Uso 13: Consultar ayuda	
Prueba	**Resultado Esperado**
Consultar ayuda	La ayuda se muestra correctamente.

Caso de Uso 14: Gestionar usuarios	
Prueba	**Resultado Esperado**
Crear un nuevo usuario correctamente	El usuario es creado y guardado en la base de datos.
Prueba	**Resultado Esperado**
Introducir errores u omisiones en la creación de un nuevo usuario	El sistema notifica los errores y requiere su corrección para poder guardar el usuario.
Prueba	**Resultado Esperado**
Editar los datos de un usuario	Los cambios se guardan correctamente.
Prueba	**Resultado Esperado**
Eliminar un usuario	El usuario se elimina correctamente.
Prueba	**Resultado Esperado**
Bloquear usuario	El usuario queda bloqueado y se impide su inicio de sesión sistema.

Caso de Uso 15: Gestionar roles	
Prueba	**Resultado Esperado**
Asignar un rol a un usuario	El usuario adquiere el nuevo rol.
Prueba	**Resultado Esperado**
Eliminar un rol de un usuario	El usuario pierde el rol.

Caso de Uso 16: Gestionar asignaciones medio-lista	
Prueba	**Resultado Esperado**
Editar una asignación	Los cambios se guardan correctamente.
Prueba	**Resultado Esperado**
Eliminar una asignación	La asignación se elimina correctamente.

Caso de Uso 17: Configurar preferencias de control remoto	
Prueba	**Resultado Esperado**
Introducir datos correctamente	La configuración se guarda correctamente y permite la transmisión de órdenes.
Prueba	**Resultado Esperado**
Rellenar datos de forma incorrecta	Las órdenes no se transmiten.

Caso de Uso 18: Controlar reproducción remota	
Prueba	**Resultado Esperado**
Controlar un reproductor remotamente	Las órdenes se transmiten correctamente y el estado del reproductor cambia según ellas, cambio de medio, estado de reproducción cambio de volumen, cambio a pantalla completa.

Prueba	**Resultado Esperado**
Controlar varios reproductores remotamente	Las órdenes de control se transmiten a todos los reproductores asociados a la cuenta de manera sincronizada.
Prueba	**Resultado Esperado**
Intentar reproducir sin conexión	Se notifica el error de falta de conexión.

Capítulo 6. Diseño del Sistema

En este apartado se detallará el diseño del sistema construido a partir del análisis realizado en el apartado anterior.

6.1 Arquitectura del Sistema

6.1.1 Diagramas de Paquetes

Los diagramas de paquetes muestran la organización lógica de la aplicación (en contraposición con la organización física, que aparecerá en los diagramas posteriores), presentando como se agrupan las clases de dicha aplicación en paquetes y la relación existente entre los mismos.

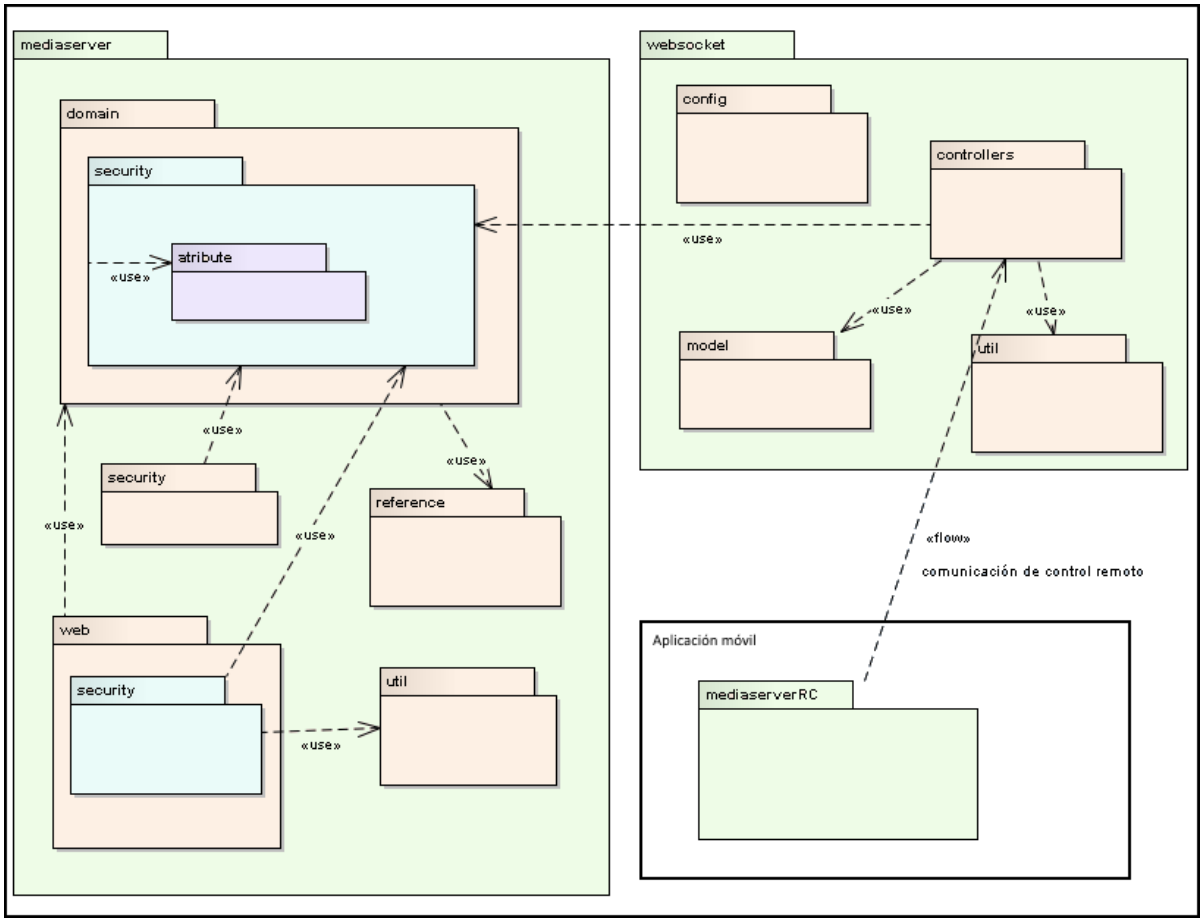


Figura 6.1 Diagrama de paquetes del sistema web

Para facilitar la apreciación, se han sombrado los paquetes de distintos colores según el nivel de profundidad.

Como se ve en el diagrama, el sistema se descompone en 3 grandes paquetes principales, los dos primeros (mediaserver y websocket) corresponden al sistema web, y el tercero

(mediaserverRC) el paquete que contiene la lógica de funcionamiento de la aplicación móvil de control remoto.

El primer paquete "mediaserver" correspondería a la aplicación web y la gestión de sus entidades, el segundo gran paquete "websocket" está dedicado a la comunicación y sincronización de reproducción, por una parte la recepción de las ordenes de control remoto y comunicación con la aplicación móvil mediante mensajes HTTP, y por otra la comunicación entre el servidor y el cliente web mediante websockets.

6.1.1.1 Paquete mediaserver.domain

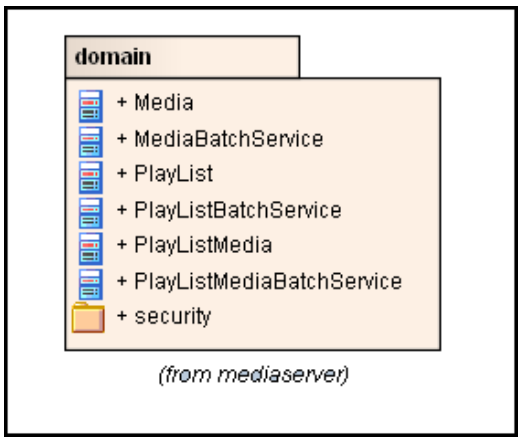


Figura 6.2 Diagrama de paquete mediaserver.domain

Este paquete contiene las entidades de dominio de la aplicación como son medios, listas de reproducción, clases de relación y clases complementarias.

6.1.1.2 Paquete mediaserver.domain.security

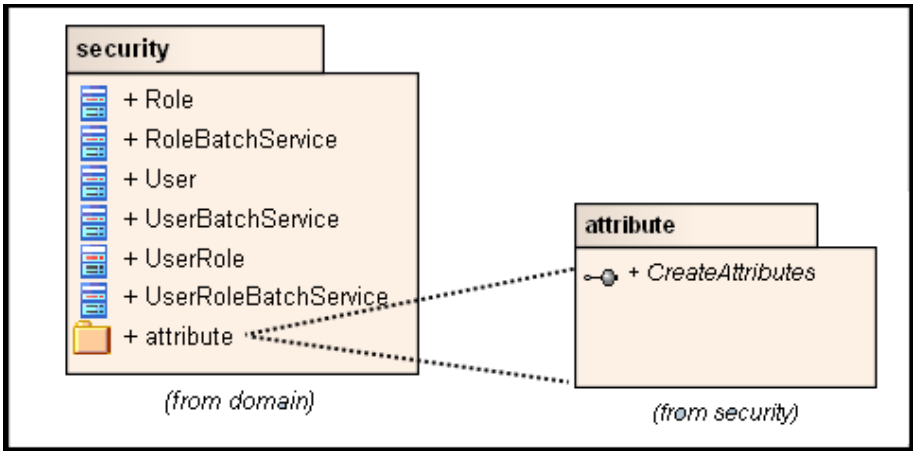


Figura 6.3 Diagrama de paquete mediaserver.domain.security

Este paquete contiene las entidades de dominio correspondientes a usuarios, roles, clases de relación, y clases complementarias

6.1.1.3 Paquete mediaserver.reference

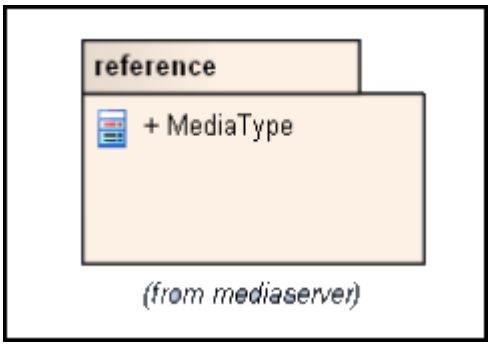


Figura 6.4 Diagrama de paquete mediaserver.reference

El paquete referencia contiene un enumerado de tipos de medios.

6.1.1.4 Paquete mediaserver.security

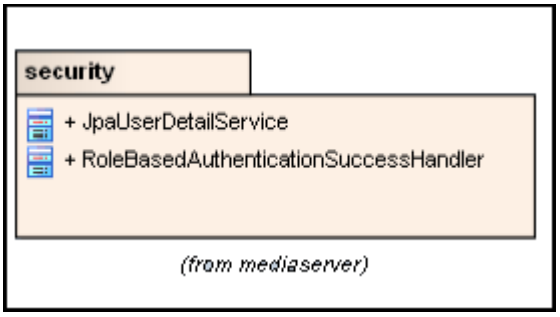


Figura 6.5 Diagrama de paquete mediaserver.security

El paquete mediaserver.security contiene clases relacionadas con la autenticación de usuarios y su acceso a páginas según el rol que posean.

6.1.1.5 Paquete mediaserver.util

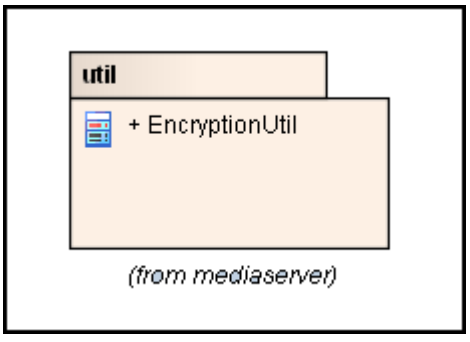


Figura 6.6 Diagrama de paquete mediaserver.util

Este paquete contiene herramientas de codificación utilizadas en el envió de emails de activación de cuentas y recuperación de passwords.

6.1.1.6 Paquete mediaserver.web

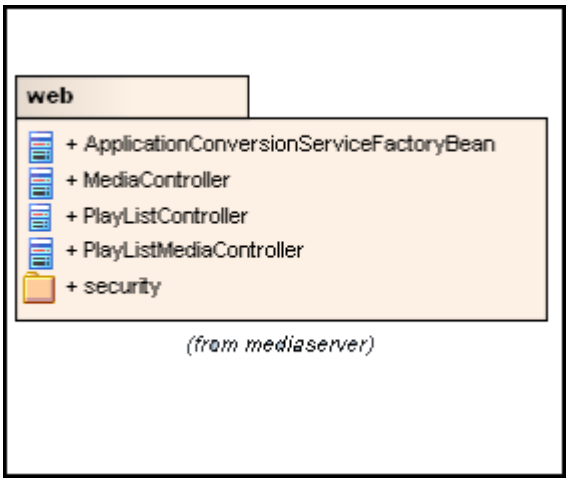


Figura 6.7 Diagrama de paquete mediaserver.web

Este paquete contiene los controladores de los modelos para la interfaz web, así como una clase de conversión de entidades a otras representaciones de las mismas, principalmente cadenas de texto.

6.1.1.7 Paquete mediaserver.web.security

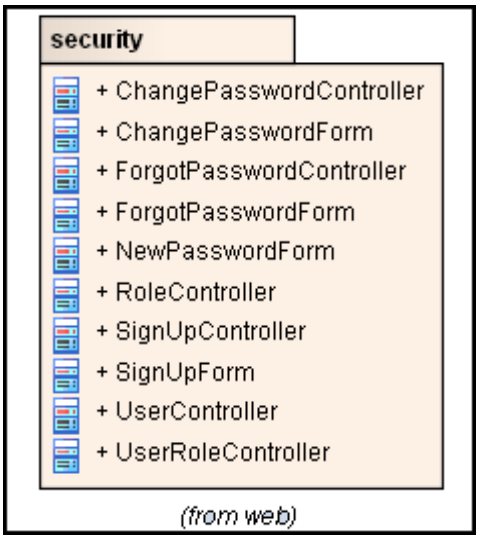


Figura 6.8 Diagrama de paquete mediaserver.domain.security

Este paquete contiene los controladores de los modelos para la interfaz web correspondientes a las entidades de seguridad, usuarios, roles etc., así como controladores adicionales para la creación de cuentas, y recuperación y cambio de passwords.

6.1.1.8 Paquete websocket.config

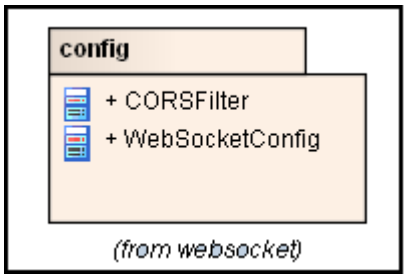


Figura 6.9 Diagrama de paquete websocket.config

Contiene clases de configuración necesarias para la utilización de los websockets.

6.1.1.9 Paquete websocket.controllers

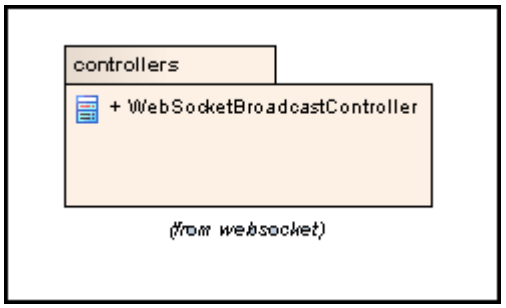


Figura 6.10 Diagrama de paquete websocket.controllers

Contiene el controlador principal de comunicación de websockets, y mensajes HTTP

6.1.1.10 Paquete websocket.model

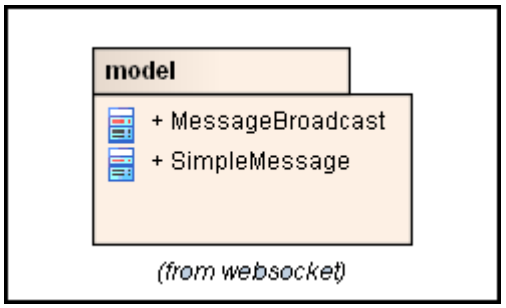


Figura 6.11 Diagrama de paquete websocket.model

Este mensaje contiene las entidades utilizadas para el envío de mensajes mediante websockets, tanto mensajes simples como mensajes de difusión mútiple.

6.1.1.11 Paquete websocket.util

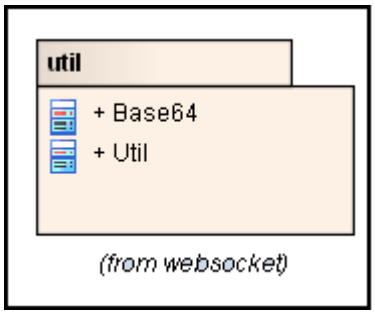


Figura 6.12 Diagrama de paquete websocket.util

Contiene herramientas de codificación utilizadas para mejorar la seguridad de las comunicaciones y obtención de fechas para el marcado de los mensajes.

6.1.1.12 Paquete mediaserverRC

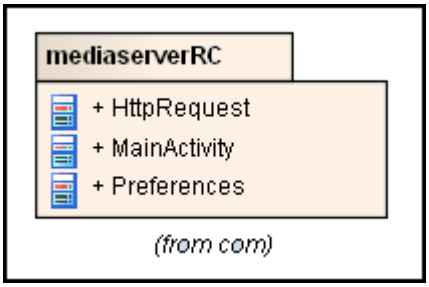


Figura 6.13 Diagrama de paquete mediaserverRC

Contiene toda la lógica de la aplicación móvil, tanto interfaz principal, como configuración y comunicación HTTP.

6.1.2 Diagramas de Despliegue

El sistema creado está compuesto por varios procesos software y máquinas que colaboran para llevar a cabo la tarea encomendada. En esta sección se trata de representar estos procesos y máquinas así como la relación existente entre ellos.

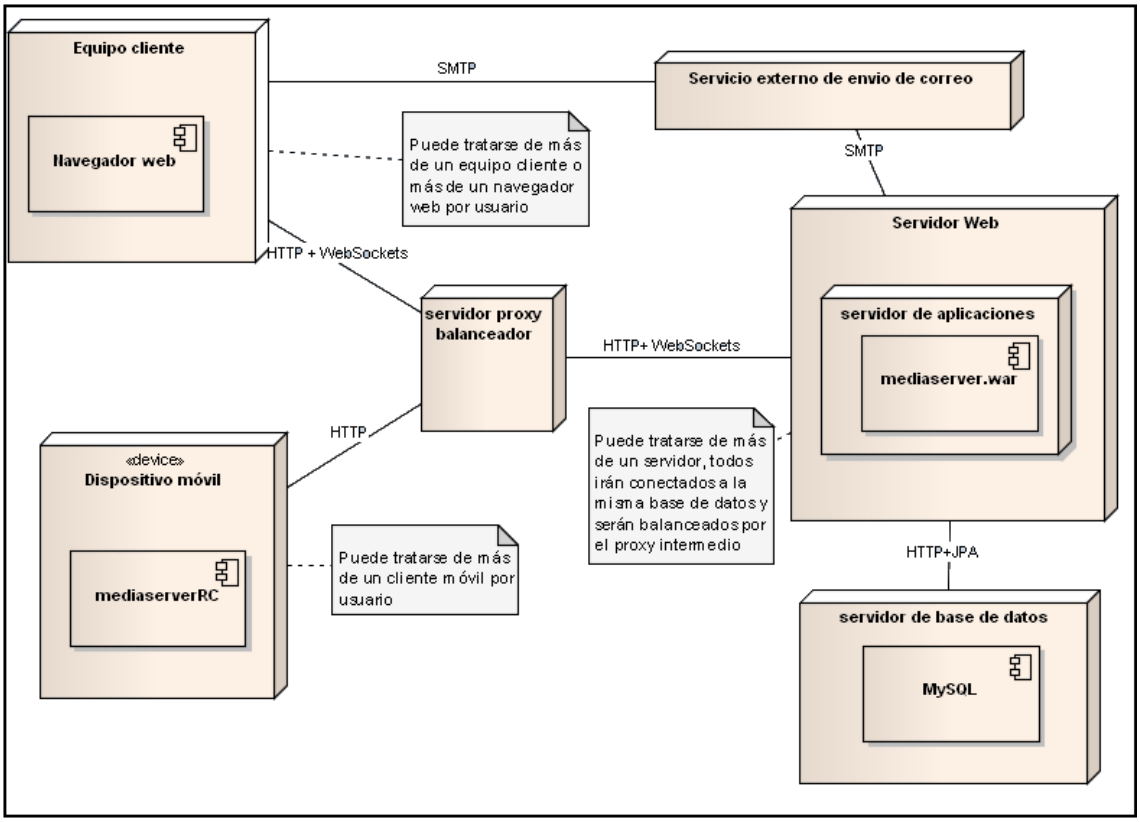


Figura 6.14 Diagrama de despliegue del sistema

El diagrama anterior representa los componentes posibles en un despliegue del sistema. No todos son estrictamente necesarios, como puede ser el caso del servidor proxy, que permite el balanceo y trabajo conjunto de varios servidores web tras él, al que de otra manera se conectarían directamente los clientes web y móviles.

6.1.2.1 Equipo cliente

Este es el equipo que utilizará el usuario para interactuar con la aplicación web. En él debe de encontrarse instalado algún navegador web (Opera, Chrome, Firefox…) con el cual comunicarse con la aplicación web por medio del protocolo HTTP, o WebSockets sobre HTTP para en el caso de la reproducción.

Desde él se podrá realizar toda la gestión de entidades del sistema.

También será aquí donde se preparen los reproductores web para realizar el control remoto desde dispositivos móviles, y donde se aprecien los resultados de este control.

6.1.2.2 Servidor Web

Este equipo responderá a las peticiones de los clientes web, y también transmitirá las ordenes remotas procedentes de los clientes móviles a sus correspondientes clientes web asociados.

En el servidor web ha de instalarse un servidor de aplicaciones, como puede ser Tomcat 7.50, que ha de ser compatible con las tecnologías utilizadas, especialmente con Datatables, y Websockets.

En este servidor de aplicaciones ha de desplegarse la aplicación web mediaserver.war.

Este nodo puede replicarse si se dispone de un servidor de proxy intermedio que se encargue de balancear las peticiones de los clientes, y ha de conectarse a una base de datos MySQL que puede estar alojada en la misma o diferente máquina.

6.1.2.3 Dispositivo móvil

Se trata del dispositivo desde el que se realizará el control remoto de la reproducción preparada en el cliente web.

Ha de disponer de un sistema operativo Android, y de la aplicación de control remoto del sistema mediaserverRC, así como una correcta configuración de credenciales y dirección del servidor para comunicarse con el servidor web, y este a su vez con el cliente web.

También es imprescindible su conexión a internet para la comunicación mediante mensajes HTTP con el servidor web.

6.1.2.4 Servidor de Base de datos

Este equipo debe tener instalada y configurada una base de datos MySQL que contiene la base de datos de la aplicación, con todas sus entidades.

Esta base de datos puede estar alojada en la misma máquina que uno de los servidores de aplicación. Pero accesible a todos ellos.

6.1.2.5 Servidor proxy balanceador

La función de este equipo es la de repartir la carga de peticiones de los clientes entre los distintos servidores que tiene tras él. Su presencia resulta transparente para los clientes que a él se conectan.

Según la configuración podrá repartir las peticiones entre los servidores web, o disponer un protocolo de respaldo si uno de ellos falla, enviando las peticiones a otro que sí esté disponible.

6.1.2.6 Servicio externo de envío de correo

La función de este servicio es enviar a los usuarios los correos que maneja la aplicación, utilizados para la activación de usuarios y recuperación de contraseñas. La utilización de un servicio de este tipo, como es el caso del gratuito mandrill.com permite tener un exhaustivo control del funcionamiento del correo, su estado de envío y recepción, disponer de un dominio de correo acorde a la aplicación, etc.

6.2 Diseño de Clases

En esta sección representaremos diagramas que muestren los paquetes y las relaciones entre ellos, así como las clases que formarán parte de la implementación final del sistema y sus relaciones.

6.2.1 Diagrama de Clases

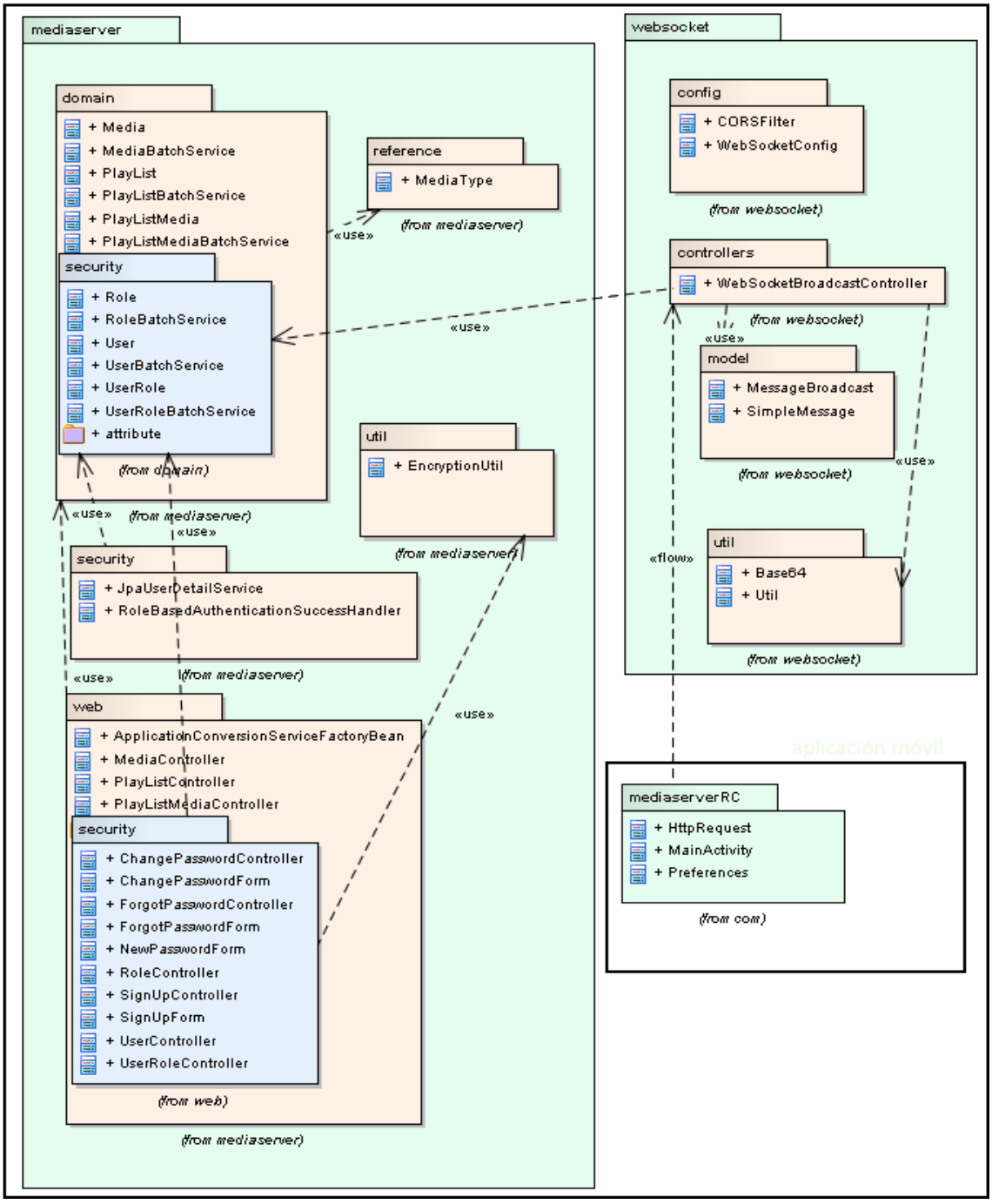


Figura 6.15 Diagrama de clases de la aplicación agrupados por paquetes

Se ha de tener en cuenta que puesto el framework Spring-Roo que se utilizará la tecnología ASpectJ, no toda la funcionalidad estará incluida en las clases java, sino que también puede estarlo en sus archivos de aspectos asociados.

A continuación se incluyen los diagramas de los distintos paquetes por separado.

6.2.1.1 Paquetes domain, security y reference

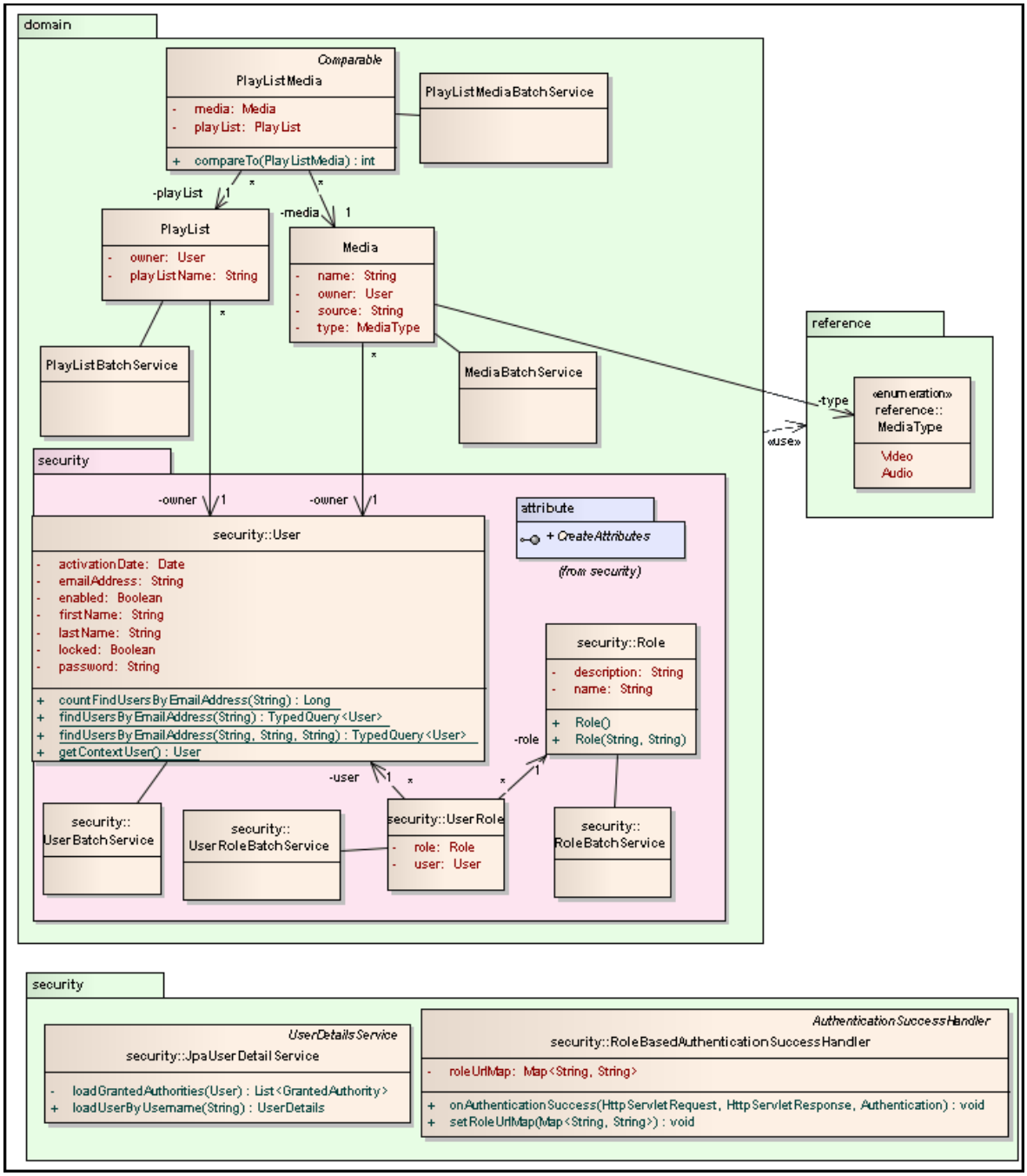


Figura 6.16 Diagrama de clases de la de los paquetes domain security y reference

Estos paquetes contienen las entidades básicas del dominio de la aplicación, por una parte los medios y listas de reproducción, y por otra los usuarios asociados a esos medios y listas, y los roles de los mismos que marcarán la capacidad de acción sobre la interfaz del sistema.

6.2.1.2 Paquete web

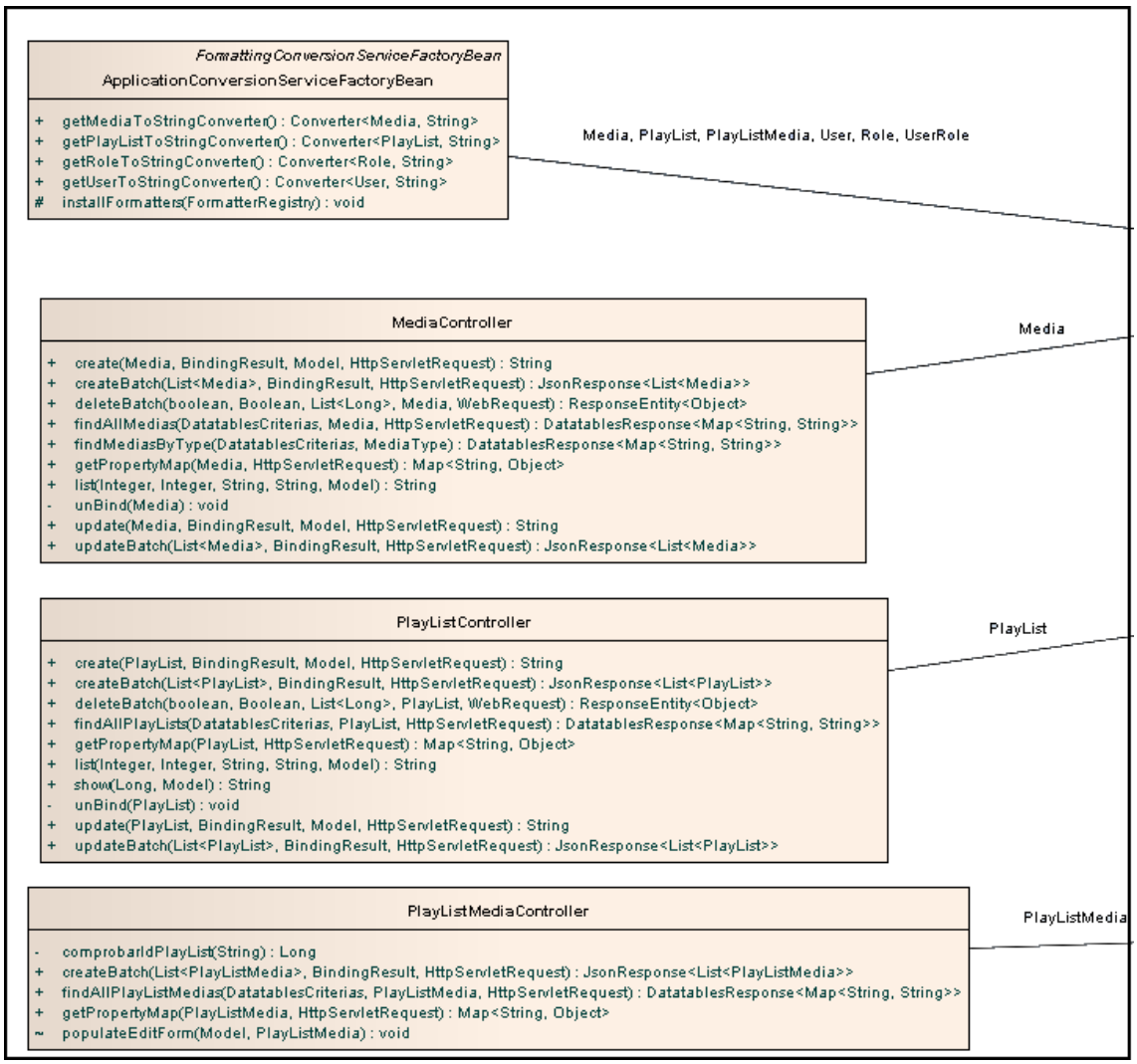


Figura 6.17 Diagrama de clases del paquete web

Las clases presentes en este paquete se corresponden a los controladores web de las entidades de la aplicación. Cada uno con la clase que va incluida en su nombre. Se encargan de comunicar la interfaz web con las entidades que maneja la aplicación.

La clase ApplicationConversionServiceFactoryBean se encarga de la conversión de las entidades a otros formatos para mostrar como por ejemplo cadenas de texto.

6.2.1.3 Paquetes mediaserver.web.security y mediaserver.util

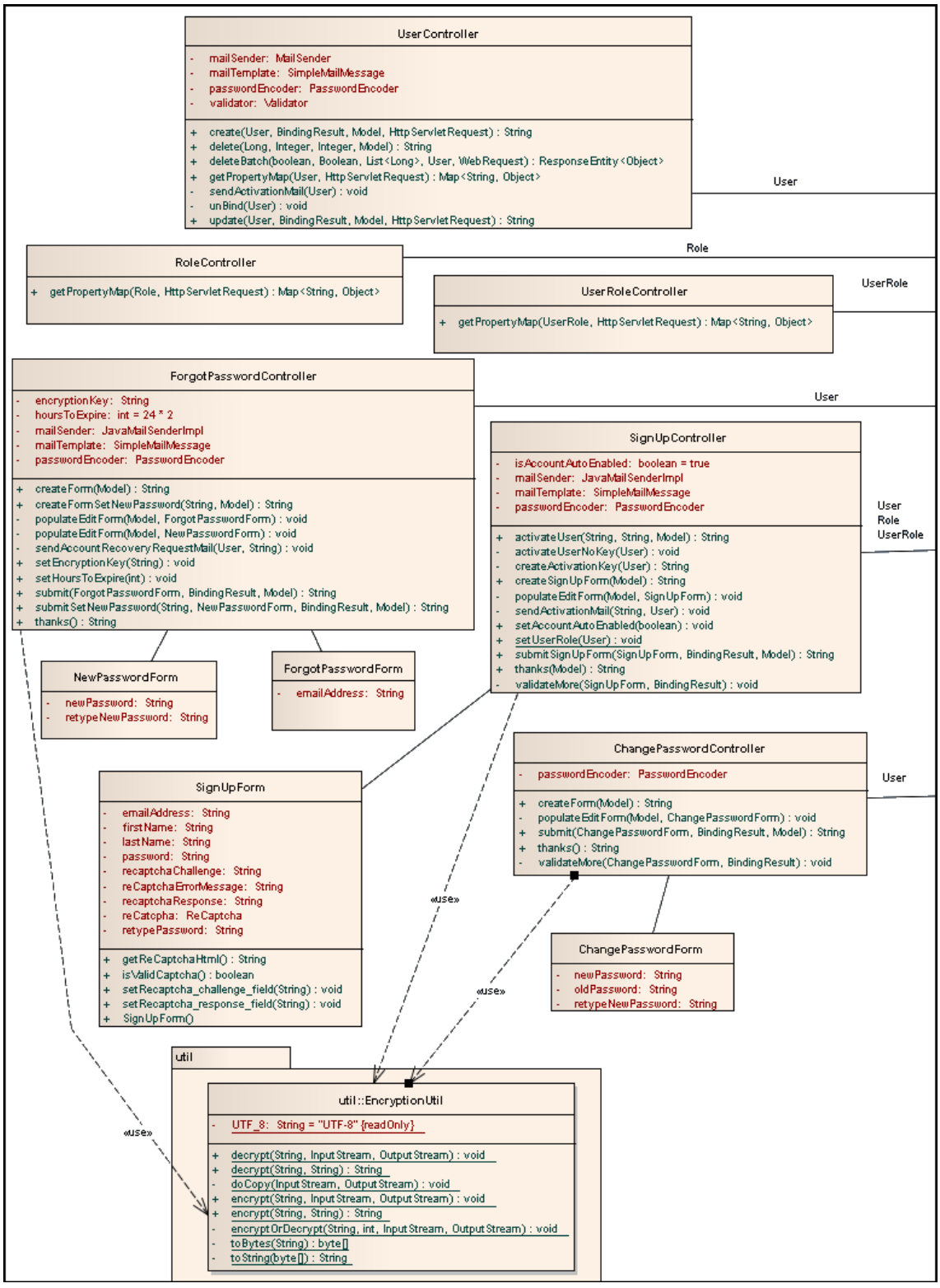


Figura 6.18 Diagrama de clases de la de los paquetes web y util

Las clases presentes en el paquete web.security se corresponden con los controladores o manejadores de las entidades de usuario de la aplicación, y sirven para manejar su gestión a través de la interfaz web.

La clase UserController se encarga del manejo directo de usuarios llevado a cabo por parte de los administradores del sistema, mientras que los controladores de SignUpController, ForgotPasswordController ChangePassowordController, son destinados a los usuarios standard, y usuarios anónimos.

También se incluye en el diagrama el paquete Util con su clase de encriptación utilizada para ofrecer mayor seguridad a la información enviada por correo con fines de activación de cuenta y recuperación e contraseñas, pues esta clase es utilizada por los controladores anteriormente citados.

6.2.1.4 Paquete websocket

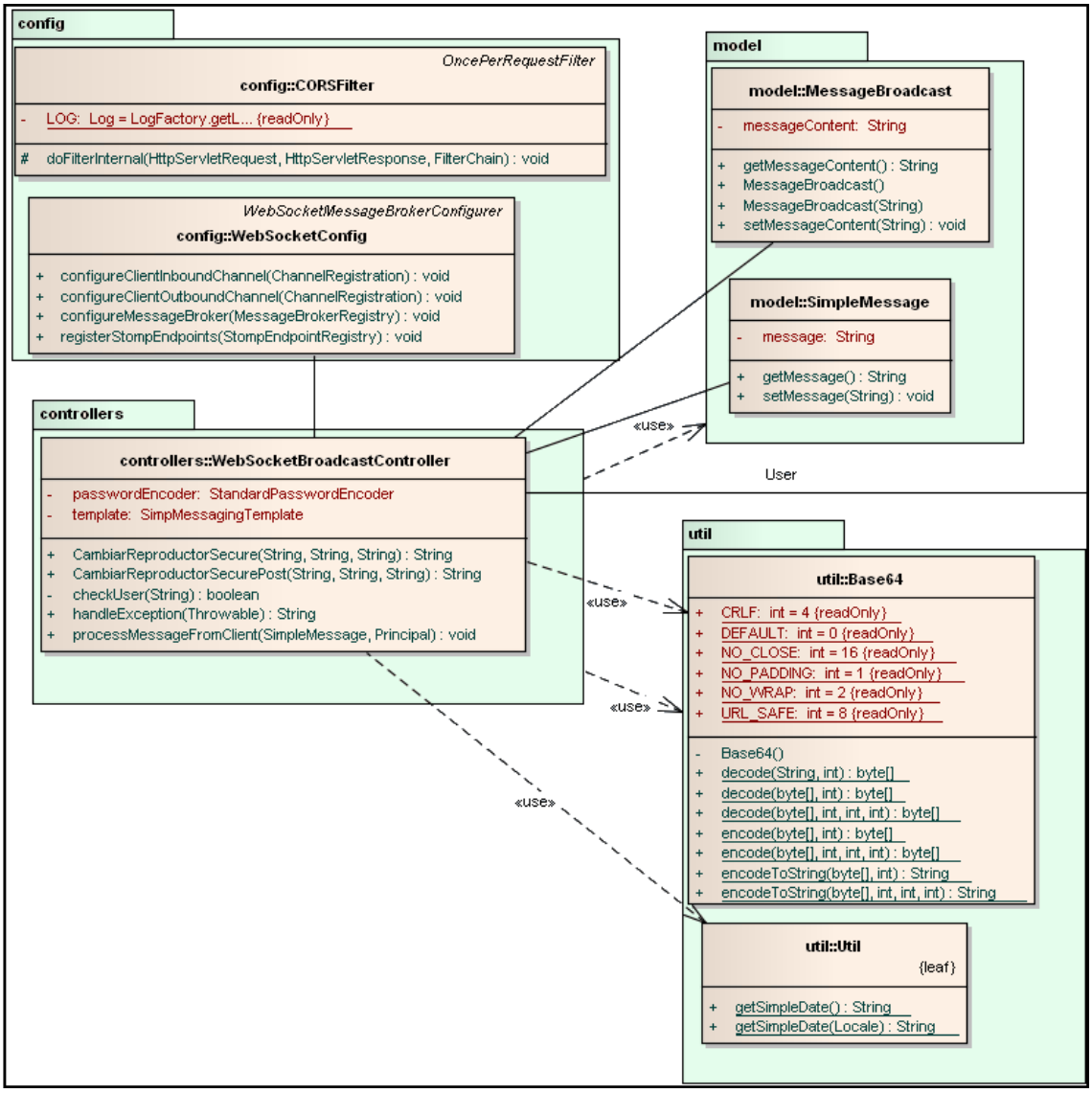


Figura 6.19 Diagrama de clases del paquete websocket

Las clases presentes en este paquete se encargan de la comunicación y sincronización de reproductores centrada en la clase WebScoketGradcastController, desde la que se envían

mensajes simples o de difusión del paquete modelo a la otra parte que maneja el envío y recepción de mensajes desde el cliente web. Esa parte está implementada en javascript.

Para marcar la fecha de los mensajes se hace uso de la case Util.

El controlador de websockets también contiene la lógica de comunicación con la aplicación móvil mediantes mensajes HTTP, y utiliza la clase Base64, procedente de la biblioteca de Android para decodificar la información enviada en los mensajes de la aplicación móvil. Se ha decidido utilizar esta clase para decodificar puesto que contiene la función complementaria a la codificación hecha por la misma clase desde la aplicación móvil.

Las clases CORSFilter y WebSocketConfig son necesarias para la configuración de la comunicación mediante websockets entre el servidor y el cliente web.

6.2.1.5 Paquete mediaserverRC

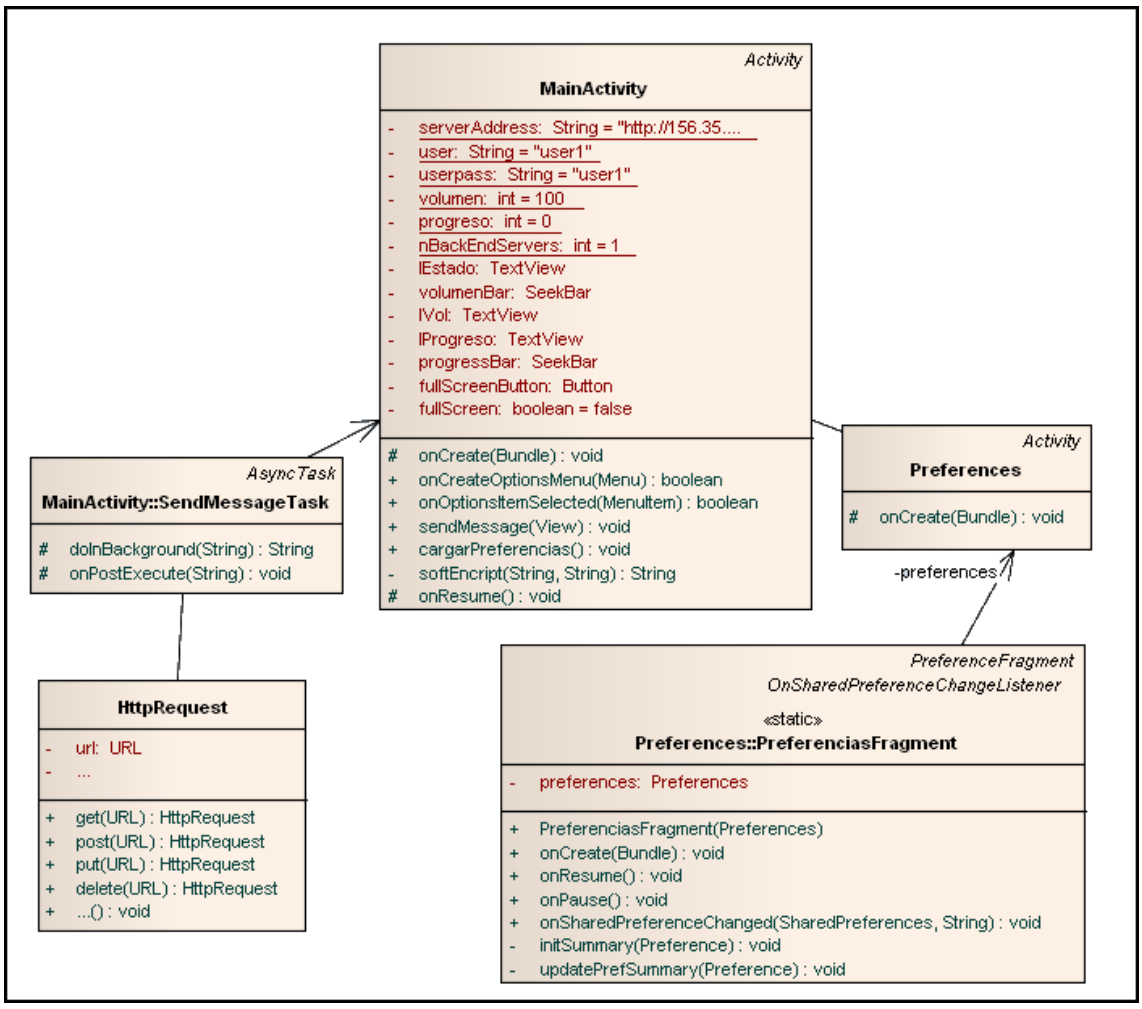


Figura 6.20 Diagrama de clases del paquete mediaserverRC

Las clases del paquete mediaserverRC implementan la lógica de la aplicación de control remoto.

La clase MainActivity, es la encargada de gestionar la interfaz de usuario y comunicarse con el servidor para transmitirle las ordenes de control a través de mensajes HTTP empleando para ello tareas asíncronas de envío de mensajes, lo cual se hace mediante la clase SendMessageTask, que a su vez utiliza la clase HttpRequest, que implementa una gran variedad de métodos y parámetros de envío de mensajes HTTP, por lo que se muestra simplificada en este diagrama.

La clase de actividad Preferences es una actividad que a su vez contiene un fragmento PreferencesFragment, que se encarga de gestionar y guardar las preferencias de configuración del usuario para luego realizar con estos datos la conexión al servidor.

6.3 Diagramas de Interacción

A continuación se muestran los diagramas de secuencia que representan las clases que intervienen en los procesos de gestión de medios y control de reproducción que constituyen los casos de uso más representativos de la aplicación.

6.3.1 Caso de Uso 7: Gestionar medios-Listado de medios

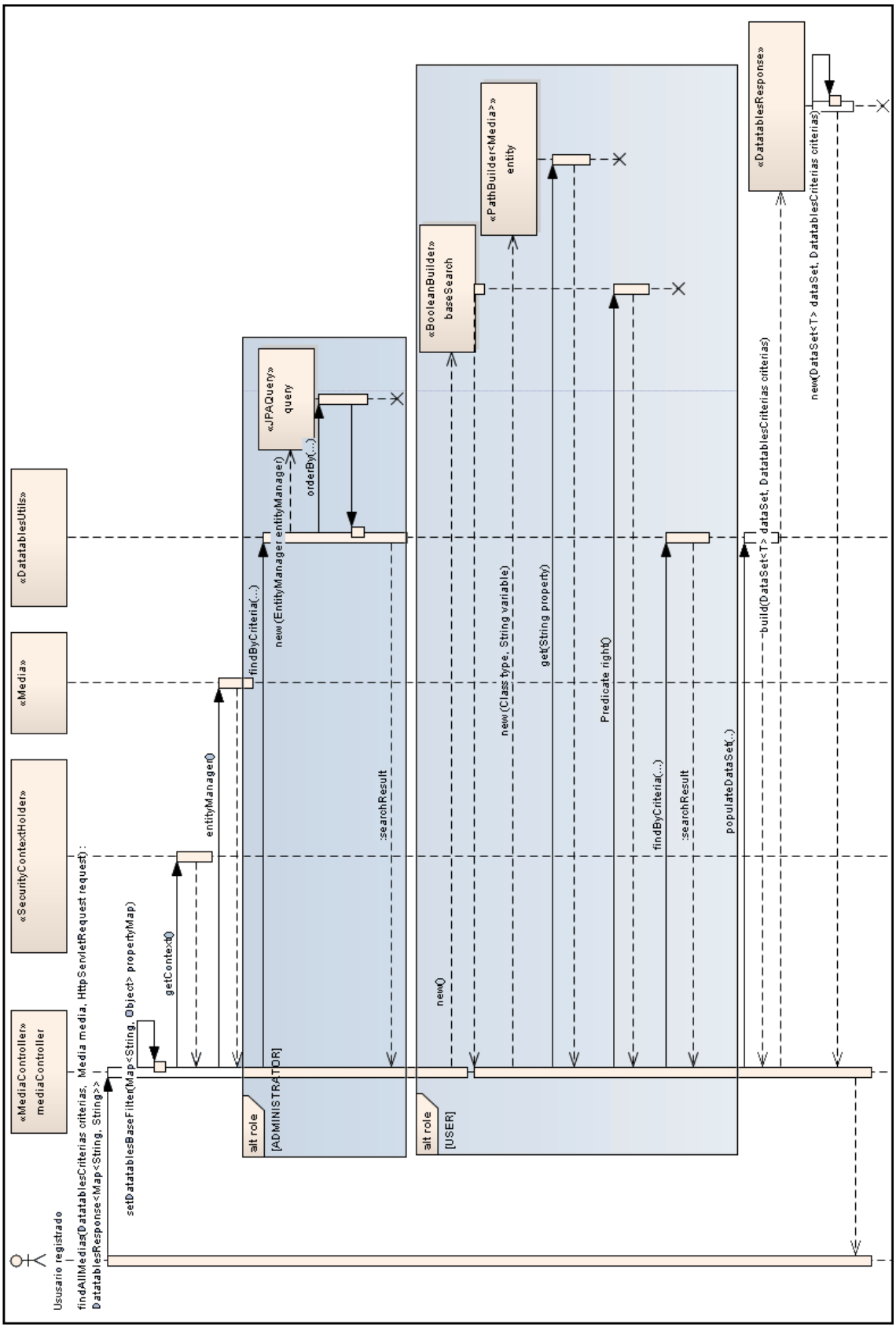


Figura 6.21 Diagrama de secuencia del proceso de listar medios

El diagrama representa el proceso por el cual un usuario solicita el listado de sus medios disponibles. Tras pasar todos los filtros establecidos de comprobación de acceso, el controlador de medios recibe la petición para mostrar la lista de medios. Lo primero que hace es obtener el rol del usuario para poder decidir si mostrar todos los medios, en caso de un usuario administrador, o solamente los medios del usuario, en caso de usuario regular.

En caso de tratarse de un usuario administrador, se recogen todos los medios, y se devuelven tras aplicarles los filtros que el usuario tenga establecidos, tales como orden, número de registros a mostrar, etc.

Si se trata de un usuario regular, a la búsqueda se le añadirá un parámetro nuevo para que los resultados sean filtrados además por el nombre del usuario propietario de los mismos.

Finalmente los datos de la respuesta serán organizados en una tabla de datos que construirá la repuesta visual para el usuario.

6.3.2 Caso de Uso 18: Controlar reproducción remota

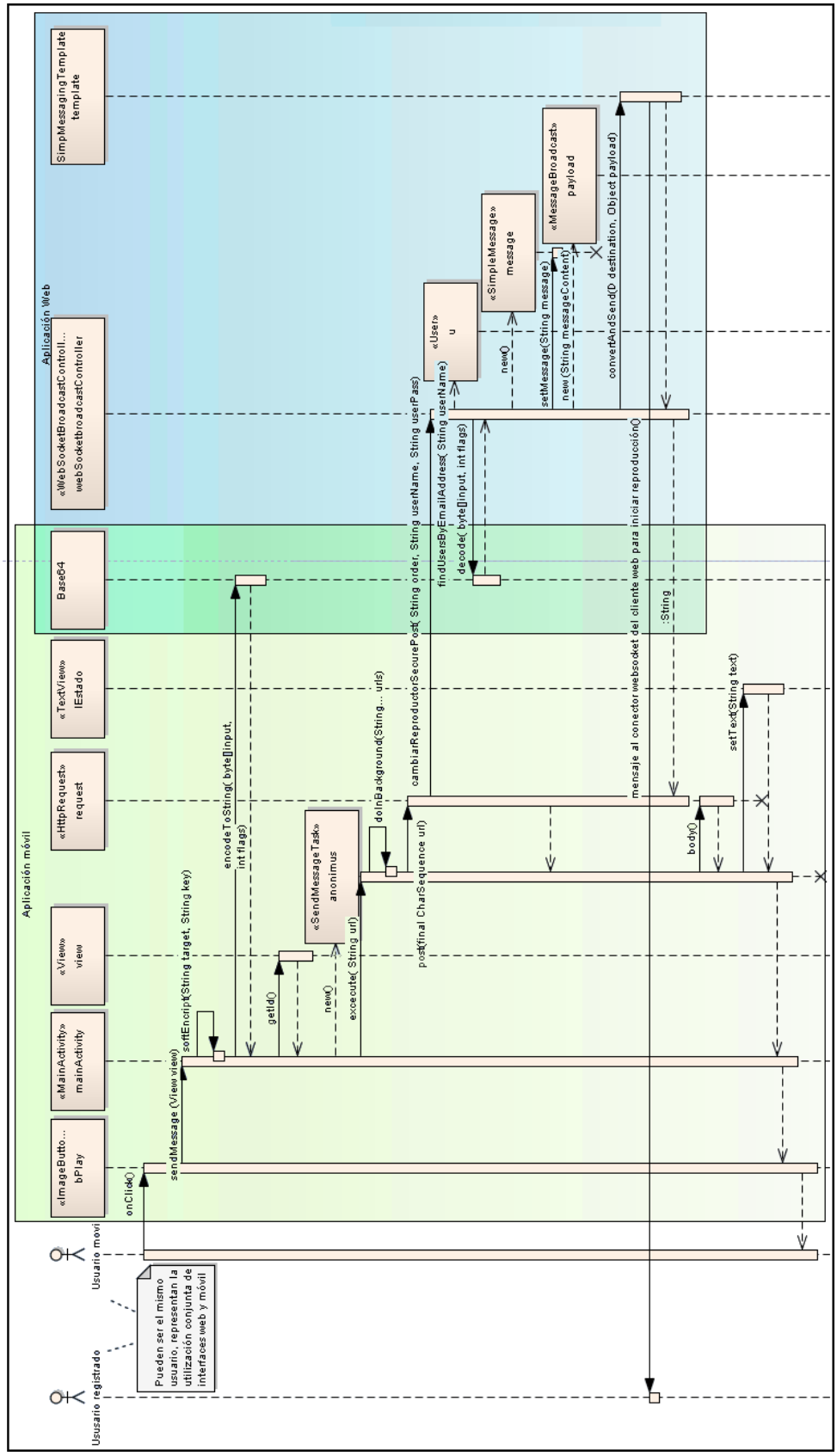


Figura 6.22 Diagrama de secuencia del proceso de listar medios

El diagrama anterior representa el proceso por el cual un usuario controla desde su dispositivo móvil, la reproducción dispuesta en su navegador como usuario registrado.

El proceso comienza cuando el usuario interactúa con el botón de *play* de la aplicación móvil, lo cual hace que la actividad principal comience el proceso de generación de un mensaje de control que será enviado al servidor para que este lo retransmita a la aplicación web.

Primeramente se codifican ciertos parámetros del mensaje haciendo uso de la clase Base64, que luego será utilizada también desde el servidor para decodificar el mensaje recibido.

A continuación, como la actividad principal debe seguir atendiendo al usuario por si se produce una nueva interacción, se lanza una actividad asíncrona de envío de mensaje, que en segundo plano hace uso de la clase HttpRequest para enviar un el mensaje post al servidor conteniendo la orden de control, así como la identificación del usuario que ha sido previamente codificada.

El servidor está preparado para recibir el mensaje de control gracias a la clase WebSocketBroadcastController, que supone el centro de comunicación de control de reproducción de la aplicación.

Tras decodificar el mensaje recibido con el método complementario al que se usó para codificarlo por parte de la aplicación móvil se comprueba que efectivamente ha sido enviado por un usuario de la aplicación y a continuación se procede a crear un mensaje de difusión con la orden de control que se envía al canal de comunicación de webSockets asociado al usuario. Además se responde a la petición de la aplicación móvil con el resultado de su orden de control si esta se ha retransmitido exitosamente, o con un error en caso contrario.

La orden por parte del servidor una vez transmitida al cliente web es recogida por el programa javascript encargado de la comunicación webSocket en el cliente, que analiza el mensaje y transmite la orden al reproductor web.

En caso de que hubiera más de un cliente web del mismo usuario dispuesto para escuchar ordenes de control, todos ellos recibirían y ejecutarían las órdenes produciéndose un control sincronizado de todos los reproductores.

6.4 Diseño de la Base de Datos

La aplicación desarrollada precisa de una base de datos en la que almacenar los datos tanto de los propios usuarios, como de los recursos asociados a los mismos.

Se ha decidido utilizar es el MySQL como sistema de gestión de base de datos; el cual es un sistema relacional, multihilo y multiusuario con más de seis millones de instalaciones desarrollado en su mayor parte en ANSI C.

6.4.1 Descripción del SGBD Usado

Desde abril de 2009 MySQL pertenece a Oracle Corporation que lo distribuye como software libre en un esquema de licenciamiento dual. Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso.

De entre las ventajas encontradas para decidirse por este SGBD podemos señalar las siguientes:

- Licencia GPL.
- Soporte de gran variedad de Sistemas Operativos.
- Popularidad, aceptación y soporte.
- Existencia de infinidad de librerías y otras herramientas para su uso en múltiples lenguajes de programación.
- Bajo consumo de recursos del sistema.
- Facilidad de configuración e instalación.
- Velocidad y rendimiento.

6.4.2 Integración del SGBD en Nuestro Sistema

Para llevar a cabo la integración de la base de datos en el sistema se ha utilizado JPA (Java Persistence API) e HIBERNATE, herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación.

Al utilizar el framework de desarrollo rápido gv-NIX /Spring Roo, la integración con la base de datos precisa de una configuración inicial durante el proceso de creación de la aplicación mediante comandos.

La base de datos ha de estar construida y disponible.

Según la descripción que mediante comandos se va dando del modelo de dominio, se van generando tanto las clases básicas necesarias, como las tablas en la base de datos y los archivos aspectJ asociados a las clases para la recuperación y guardado de las entidades y

atributos de estas clases en la base de datos. También se crean según se soliciten métodos de búsqueda según atributos específicos.

El marcado en las clases java y ficheros aspectJ de la relación con la base de datos se realiza mediante anotaciones.

6.4.3 Diagrama E-R

El esquema de base de datos creado por el framework tras la descripción de las entidades, enumerados y sus relaciones es el siguiente:

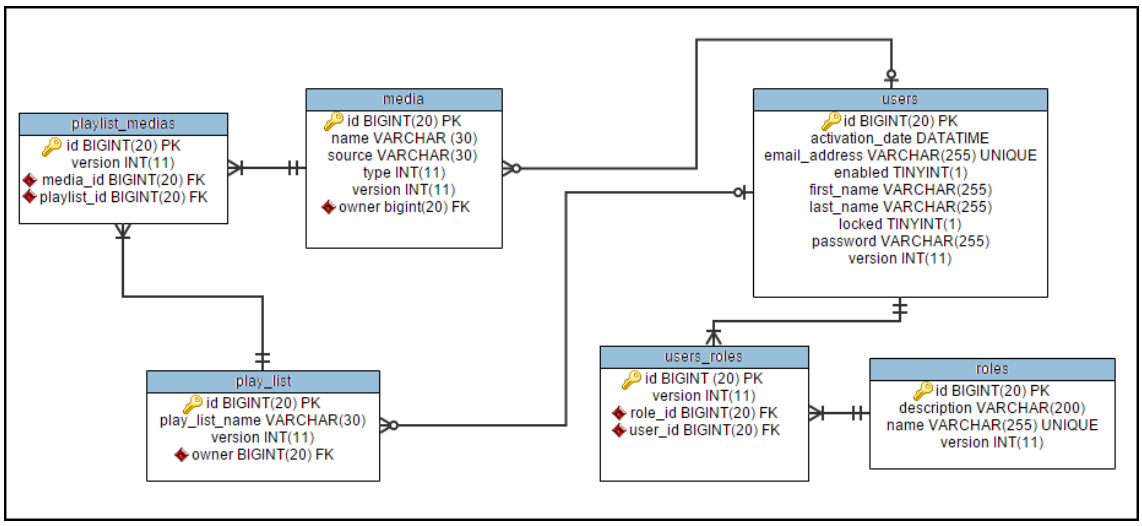


Figura 6.23 Diagrama Entidad-Relación de la base de datos de la aplicación

Como vemos, el diagrama se corresponde con los otros modelos ya presentados para representar las entidades de la aplicación, con cuatro entidades principales: media, play_list users y roles, y otras dos entidades para evitar asociaciones de muchos a muchos: playlist_medias y users_roles.

Todas las tablas tiene un atributo id que funciona como clave primaria que se va autoincrementando a medida que se incorporan nuevos elementos. La clave primaria está marcada con las siglas PK en el diagrama y con el icono de una llave. Además las columnas que funcionan como enlace con otras, claves externas están marcadas con las siglas FK y un icono rojo.

Los usuarios han de tener una dirección de email distinta a todos los demás por lo cual esa columna ha de tener un valor único en toda la tabla.

Las tablas también cuentan con un atributo versión, cuya utilidad es evitar conflictos a la hora de actualizar los valores de las filas.

6.5 Diseño de la Interfaz

En esta sección mostramos la interfaz definitiva de la aplicación y las diferentes partes de las que consta.

Se presentarán por separado las interfaces web, y móvil.

6.5.1 Interfaz web

En esta sección se muestran algunas capturas de pantalla del aspecto que tendrá la interfaz web, y qué es lo que con ella se puede hacer en cada una de las páginas.

6.5.1.1 Interfaz pública

A esta parte de la interfaz puede acceder cualquier usuario sin necesidad de haber superado el proceso de login.

Figura 6.24 Página principal de la aplicación

Esta será la página principal de la aplicación. Su estructura es la que se mantendrá como norma general en todo el sitio, con un menú en la parte superior, el contenido en el centro y un pie en la parte inferior con información de contacto, última revisión, y cambio de tema del sitio.

En el menú de configuración en este punto solo se puede seleccionar el idioma del sitio.

el menú principal, como usuario anónimo solo ofrece la opción de iniciar sesión dirigiendo al usuario a la página de login.

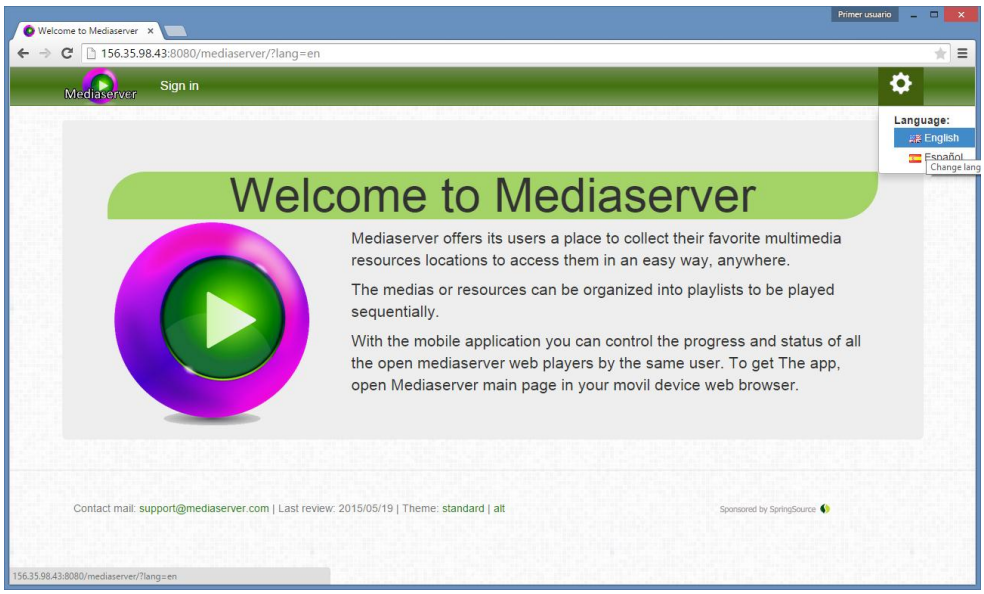


Figura 6.25 Página principal de la aplicación tras solicitar un cambio de idioma

Al solicitar un cambio de idioma, la página se recarga con los textos adaptados al idioma seleccionado y esta configuración se conserva de ahí en adelante mostrando todas las páginas en el idioma elegido. El usuario puede cambiar el idioma de nuevo en cualquier momento.

Figura 6.26 Página principal de la aplicación tras solicitar un cambio de tema

Al solicitar un cambio de tema, la página se recarga cambiando el esquema de colores básico por el alternativo para facilitar que el usuario aprecie los contenidos si padece algún defecto visual que le haga distinguir mejor los mismos con el nuevo esquema de colores, o simplemente le gusta más.

Se ha preferido mantener el cambio de tema en el pie de la página para no sobrecargar el menú de configuración, y por tener cierta relación con la accesibilidad en su cambio de colores.

La elección de tema se conserva para toda la navegación sucesiva, aunque el usuario puede cambiarla de nuevo en cualquier momento.

Figura 6.27 Página principal de la aplicación vista en un navegador móvil (pantalla estrecha)

Al acceder al sitio de la aplicación con el navegador de un dispositivo móvil, como por ejemplo un teléfono, se le recuerda al usuario la posibilidad de obtener la aplicación Android de control remoto simplemente haciendo clic en el llamativo cartel.

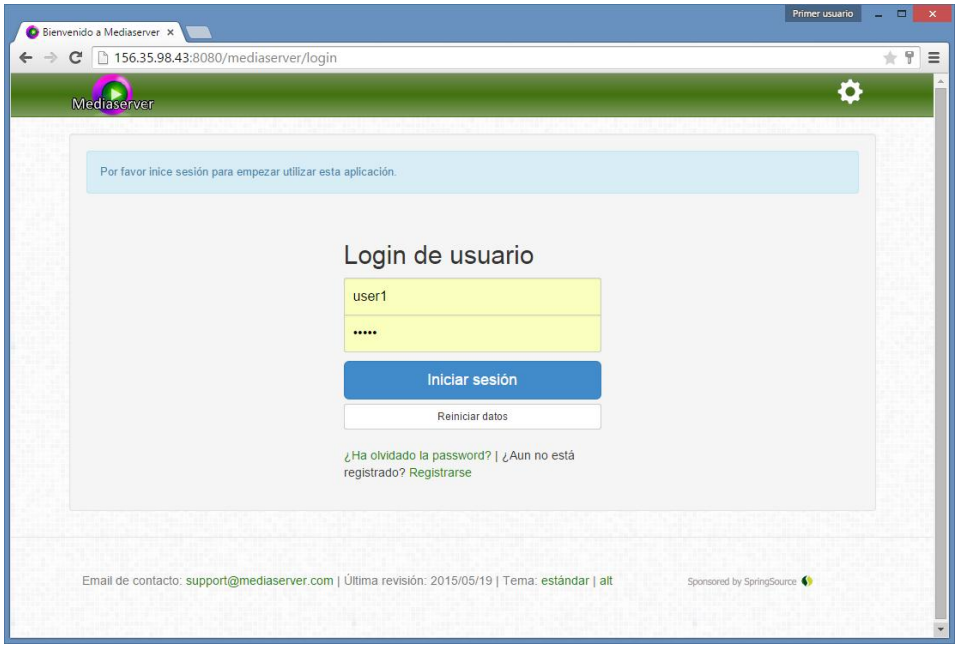


Figura 6.28 Página de inicio de sesión

Antes de empezar a utilizar las funciones de la aplicación el usuario ha de iniciar sesión en la pantalla de login a la que se accede a través del menú de la página principal.

Si el usuario no dispone de una cuenta, se le ofrece un enlace al formulario de registro de nuevos usuarios.

Si dispone de cuenta pero ha olvidado su contraseña también dispone de una opción para reestablecerla previa confirmación mediante su correo.

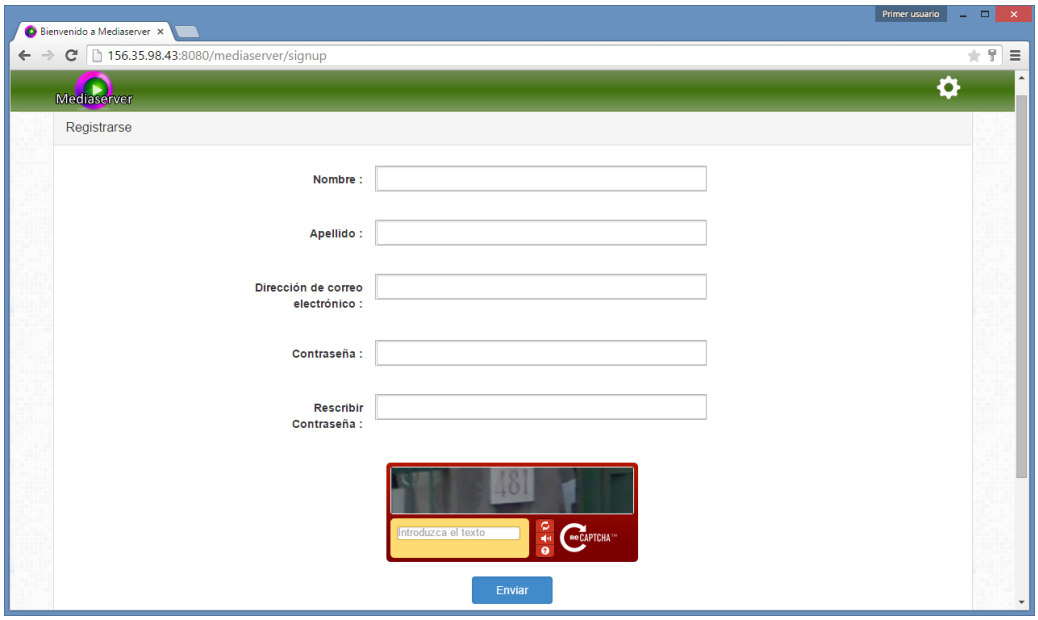


Figura 6.29 Página de registro

Para registrarse como nuevo usuario es necesario completar el formulario de la imagen anterior con los datos del usuario: nombre apellido, dirección de correo electrónico (que funcionará como nombre de usuario y debe ser única en el sistema) y contraseña por duplicado para evitar errores fortuitos

También ha de rellenarse un recaptcha para evitar creaciones automáticas de cuentas en el sistema.

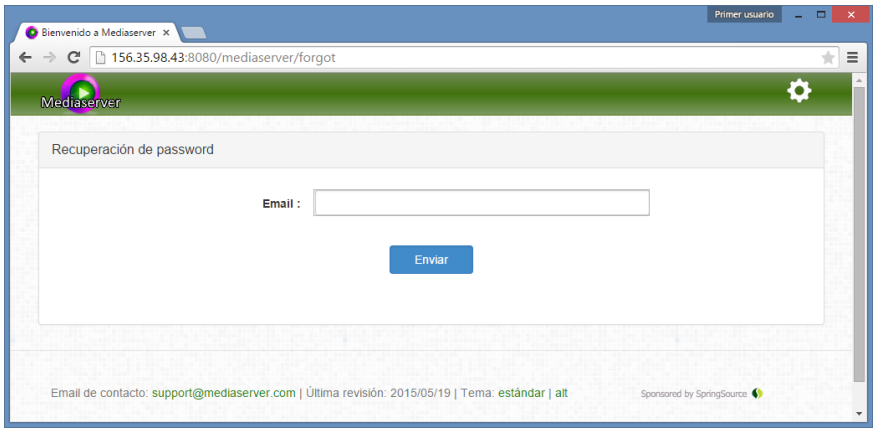


Figura 6.30 Página de recuperación de contraseña

Para recuperar la contraseña ha de introducirse la dirección de email del usuario para poder enviar el email con el enlace para el restablecimiento.

6.5.1.2 Interfaz de usuario registrado

Una vez el usuario haya iniciado sesión introduciendo sus credenciales en la página de login, será enviado al listado de medios.

El menú ha cambiado con nuevas opciones, para la gestión y visualización de medios y listas de reproducción, ayuda, cambio de password y cierre de sesión.

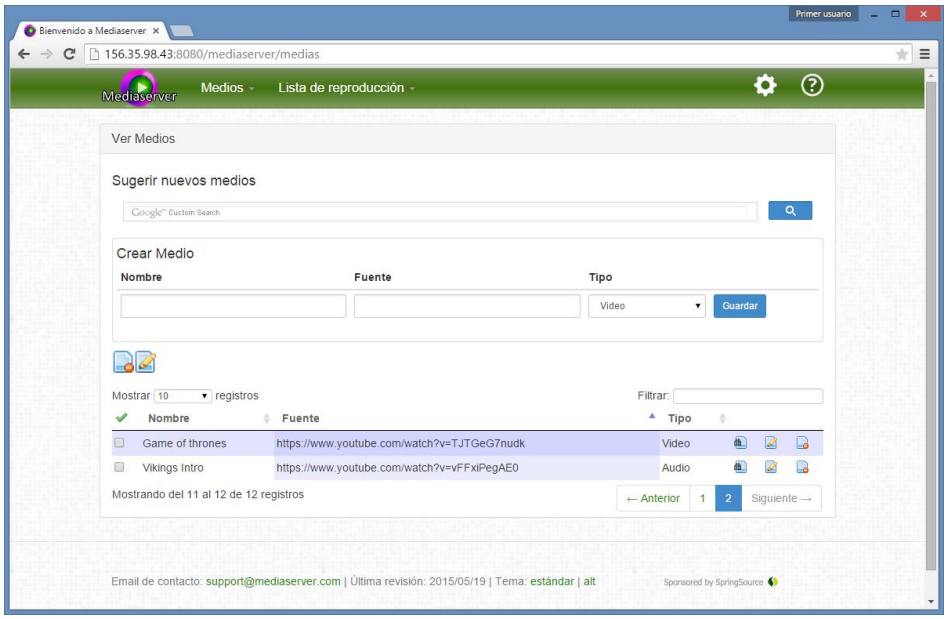


Figura 6.31 Página de listado de medios

En esta página, el usuario podrá revisar los medios que ya ha guardado, editarlos, borrarlos o crear nuevos medios.

Para ello solamente ha de introducir un nombre, fuente y tipo de medio y hacer clic en el botón de guardar.

Utilizando el formulario de sugerencia de medios, podrá buscar recursos para guardar con simplemente introducir su nombre.

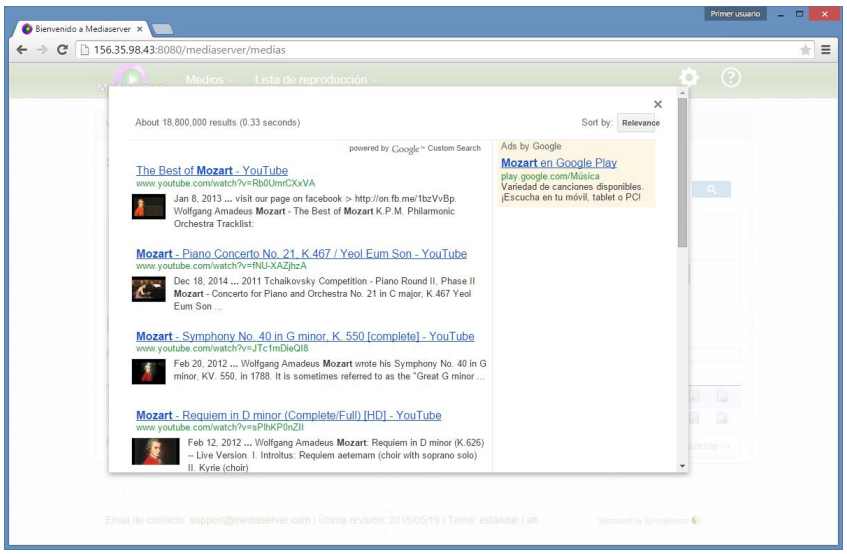


Figura 6.32 Página de creación de listado de medios con overlay de sugerencia activo

Al hace una búsqueda con la herramienta de sugerencia, aparecerán los resultados en una página superpuesta para que el usuario pueda elegir cuál de ellos prefiere, y al hacerlo, los datos del medio se rellenarán automáticamente en el formulario para añadir el nuevo medio a la colección.

Desde el listado también se puede acceder a la reproducción de cada medio concreto con su enlace correspondiente en la parte de la derecha.

La relación de elementos se puede reordenar según las distintas columnas, se pueden filtrar los medios mostrados, cambiar el número máximo de medios por página, cambiar de página, etc.

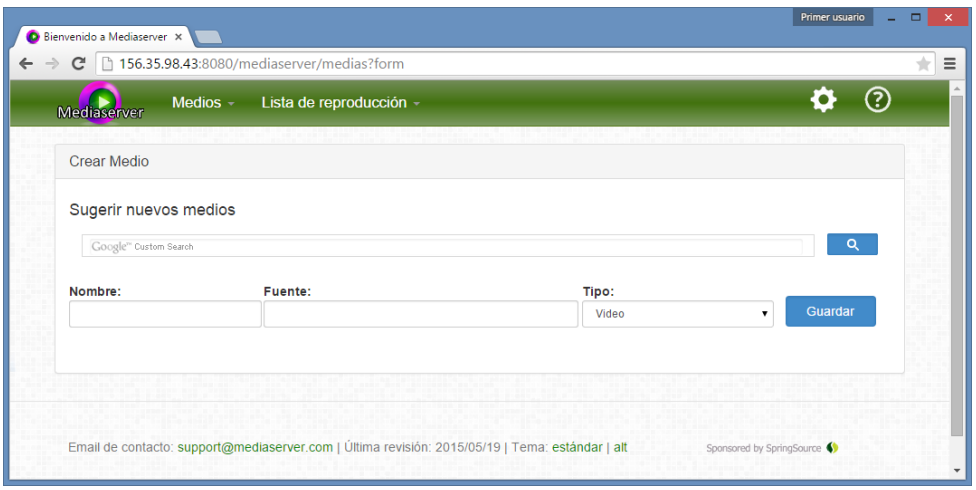


Figura 6.33 Página de creación de nuevo medio

El formulario para crear un nuevo medio accesible desde el menú es muy similar al ya explicado en el listado

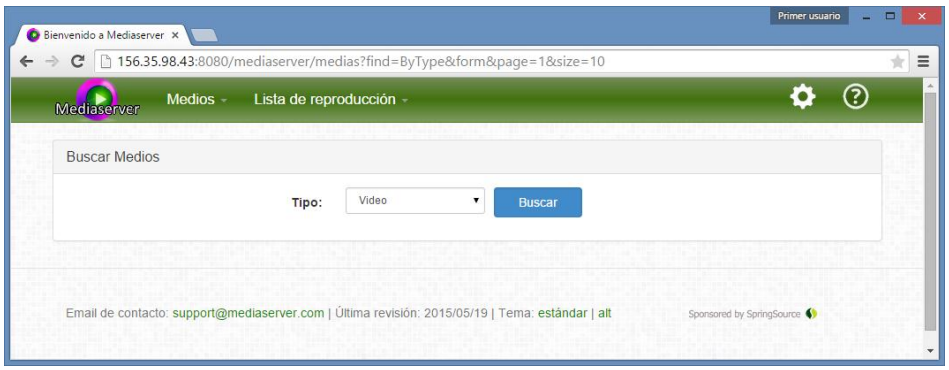


Figura 6.34 Página de búsqueda por tipo

La opción de menú de buscar medios por tipo ofrece un formulario para seleccionar el tipo que queremos buscar, entre video o audio, y al hacer clic en buscar, muestra un listado de medios del usuario filtrado según el tipo elegido.

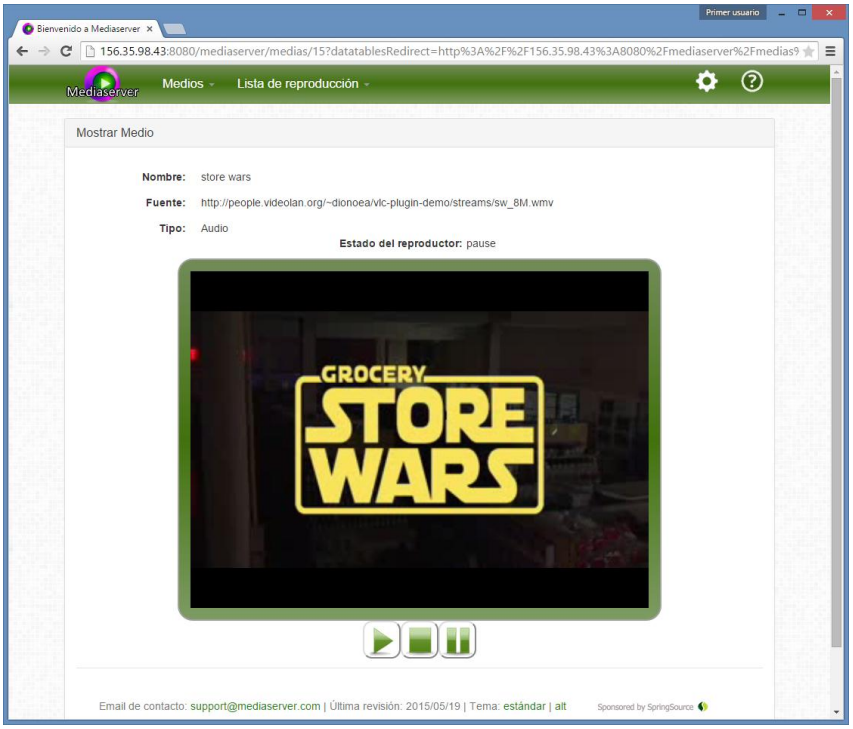


Figura 6.35 Página de ver medio

Al acceder a la visualización de un medio desde el listado accedemos a la página de reproducción. En ella además de la información del medio concreto disponemos de un reproductor con controles para visualizadlo e información sobre el estado de reproducción.

En esta página se reflejarán las órdenes recibidas del control remoto móvil.

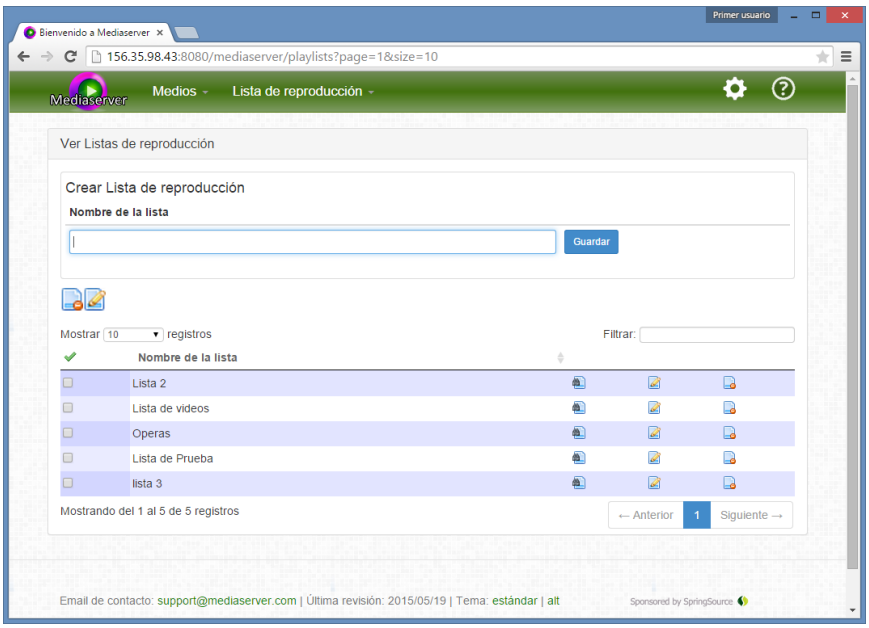


Figura 6.36 Página listado de listas de reproducción

En esta página se puede revisar y editar las listas de reproducción del usuario así como incorporar otras nuevas con solo especificar su nombre.

Además podremos acceder a la visualización de cada lista con el enlace correspondiente.

Como en el caso del listado de medios, la tabla de listas de reproducción puede filtrarse, reordenarse, y cambiar el número máximo de registros mostrados.

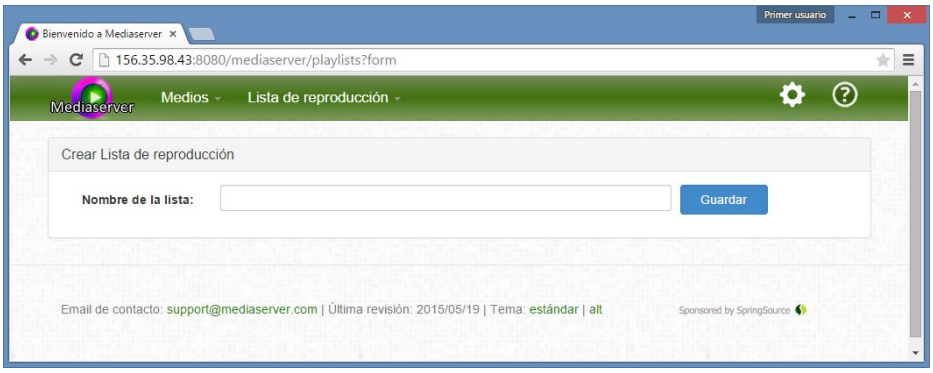


Figura 6.37 Página de crear lista de reproducción

El formulario de creación de nuevas listas, del todo similar al ya descrito en el listado.

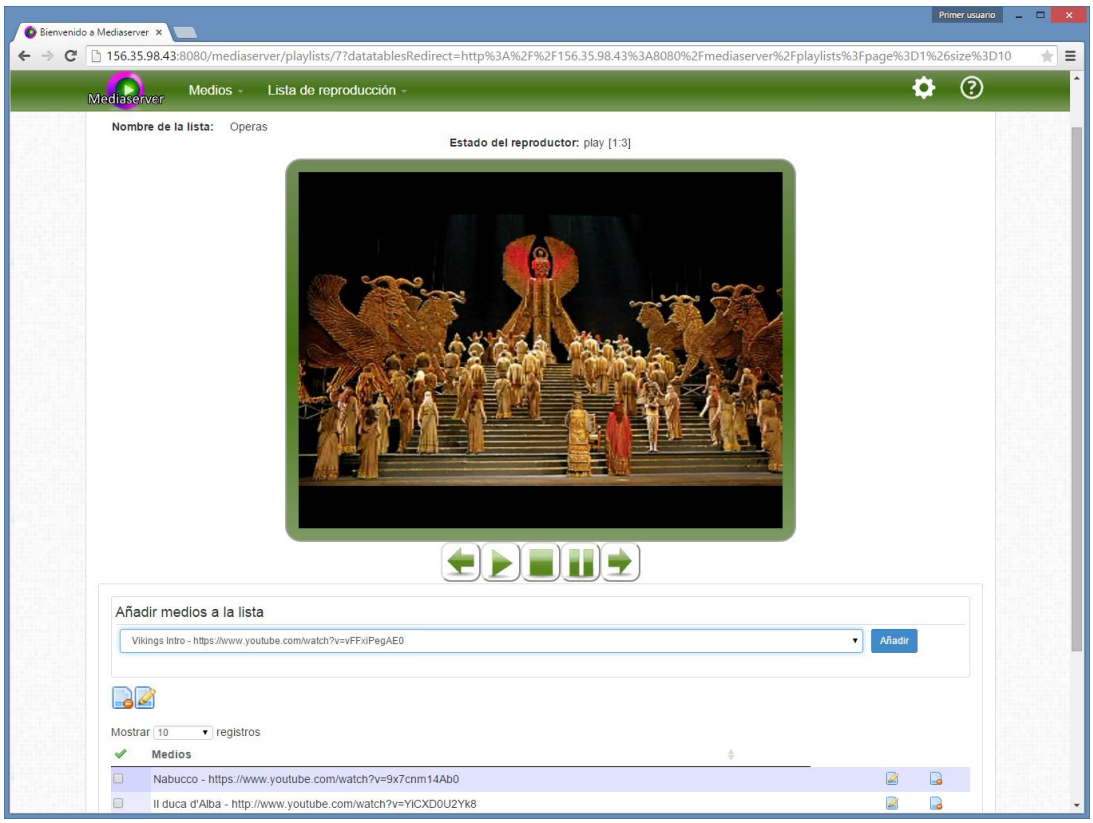


Figura 6.38 Página de ver lista de reproducción

Al acceder a la visualización de una lista de reproducción disponemos además de un reproductor y controles para la reproducción, de un formulario para añadir y editar qué medios forman parte de esa lista.

Esta página de reproducción también es controlable desde la aplicación móvil.

Además del estado de reproducción se informa del número de medio reproducido actualmente y el número total de medios de la lista.

Figura 6.39 Página de ayuda

En la ayuda de la aplicación se trata de resolver las dudas que el usuario pudiera tener para conseguir que el usuario reproduzca correctamente los videos en su ordenador, habilitando y desbloqueando el plugin de reproducción.

También se le informa de donde conseguir la aplicación móvil de control remoto.

Si el usuario quiere cambiar su contraseña puede hacerlo a través de la página de cambio de contraseña accesible desde el menú de configuración, rellenando el siguiente formulario en el que se le solicita la contraseña actual y por duplicado la nueva.

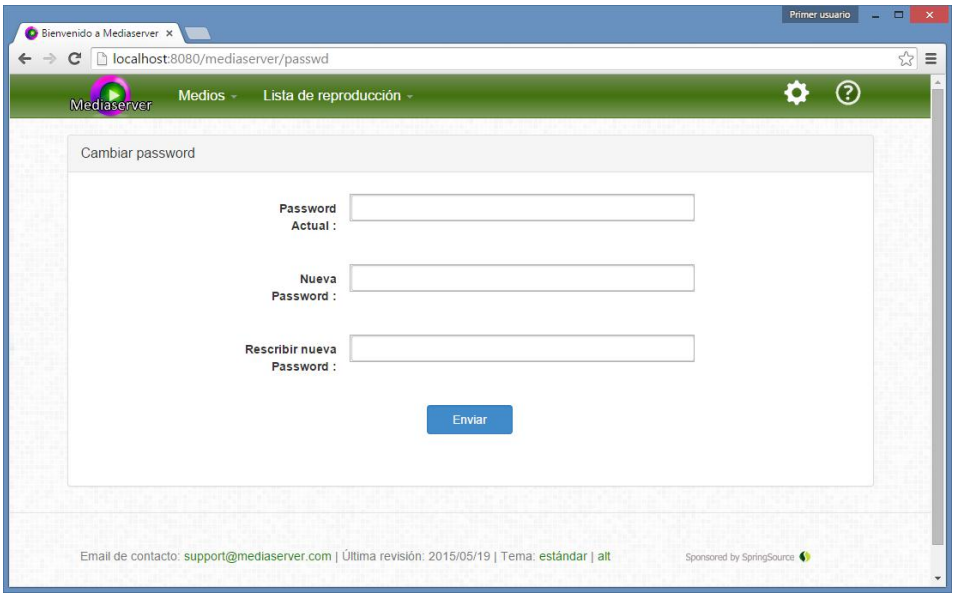


Figura 6.40 Página de cambio de contraseña

6.5.1.3 Interfaz de usuario administrador

Al iniciar sesión como administrador se accede a la página de listado de usuarios, y el menú se amplía con todas las funciones de administración de usuarios, roles y asociaciones de medios y listas, además de las ya presentes como usuario standard.

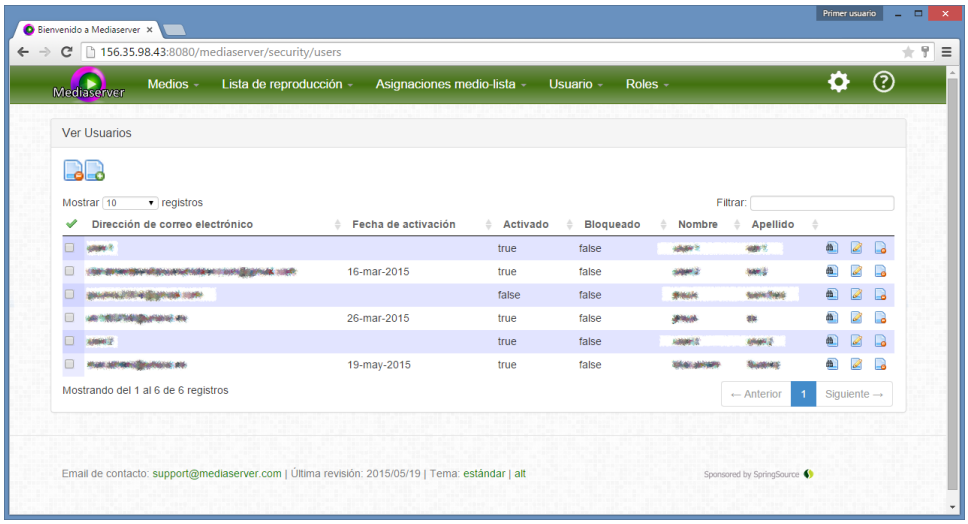


Figura 6.41 Página de listado de usuarios

En la página de listado de usuarios se puede ver la información relativa a todos los usuarios de la aplicación, comprobar su estado de bloqueo y fecha de activación.

También se puede acceder a ver uno de los usuarios en detalle, editarlo o borrarlo.

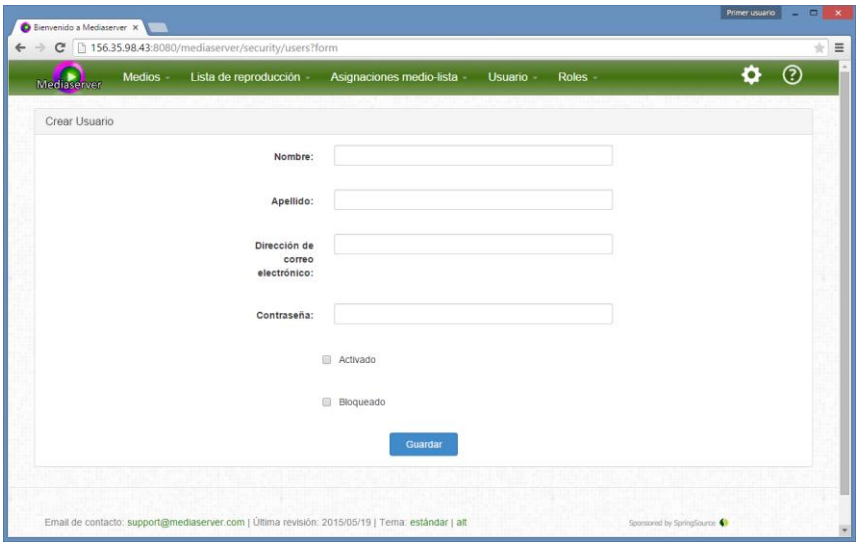


Figura 6.42 Página creación de usuario

Aparte del registro de usuarios mediante el formulario de registro, los usuarios pueden ser creados y activados manualmente por un administrador mediante el formulario de la figura precedente.

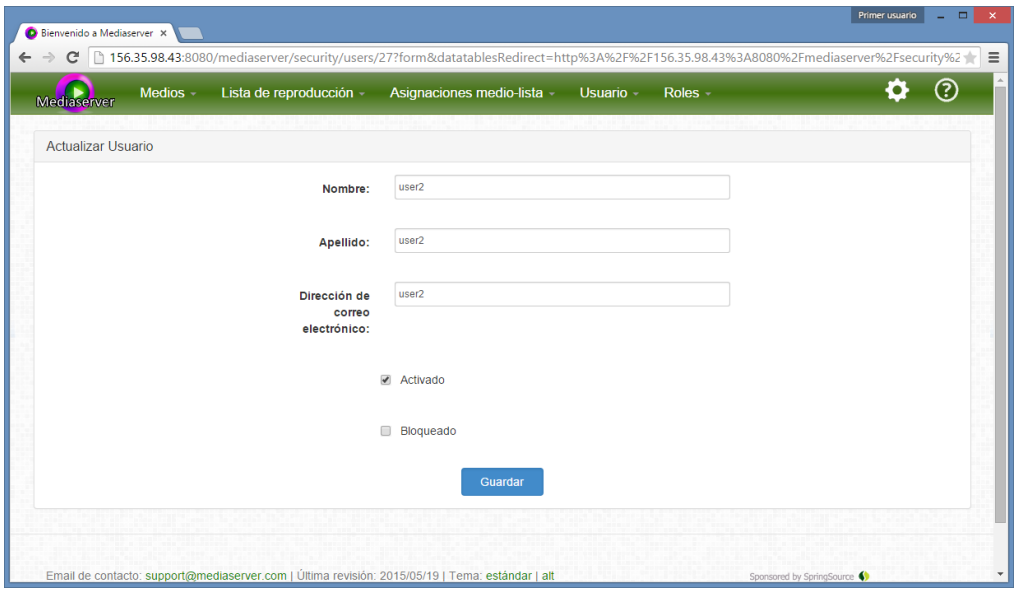


Figura 6.43 Página edición de usuario

Muy similar a la página de creación en ella se pueden modificar los datos y estado del usuario.

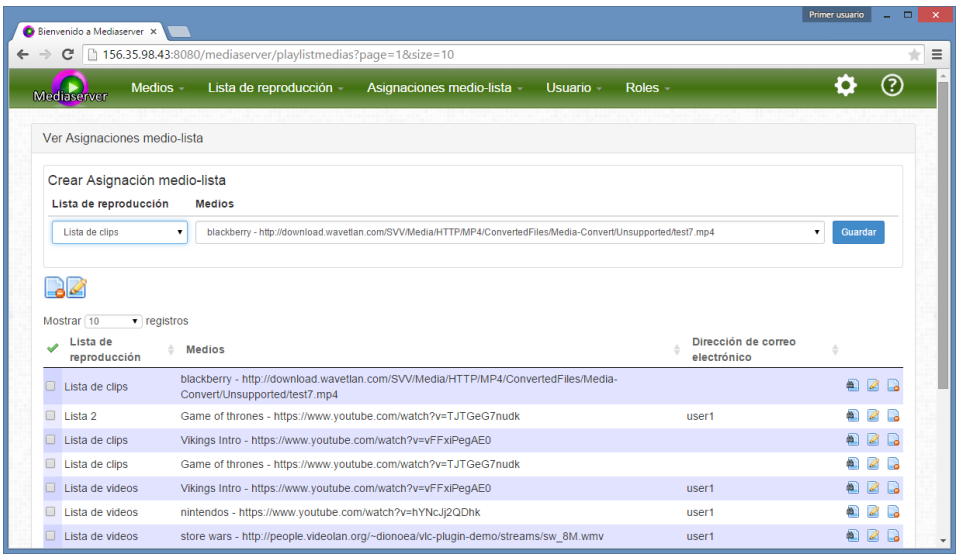


Figura 6.44 Página de listado de asignaciones medio-lista

Esta vista permite a un administrador ver qué medios pertenecen a qué listas de reproducción y a qué usuarios.

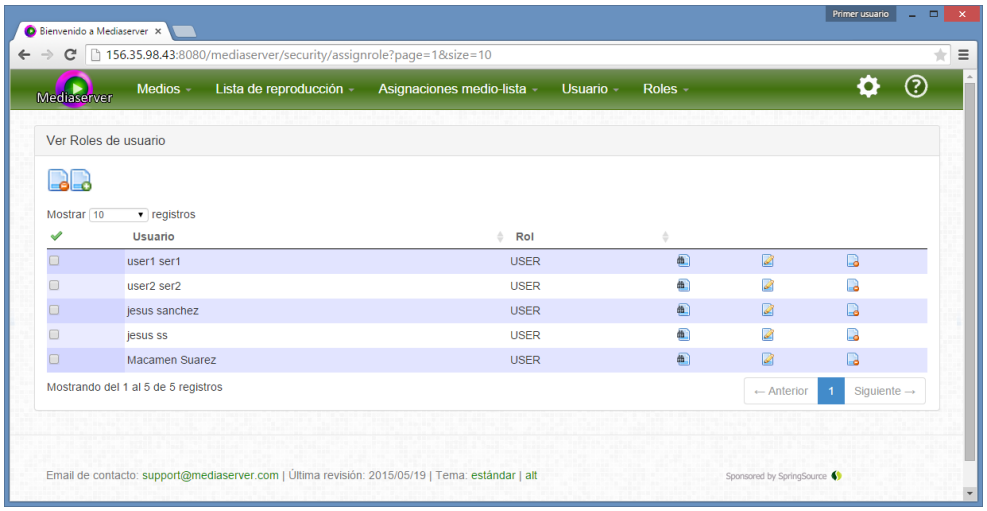


Figura 6.45 Página de listado de roles

En esta página el administrador podrá comprobar qué rol está asociado a cada usuario, acceder a su modificación o eliminar un rol de uno de ellos si es necesario.

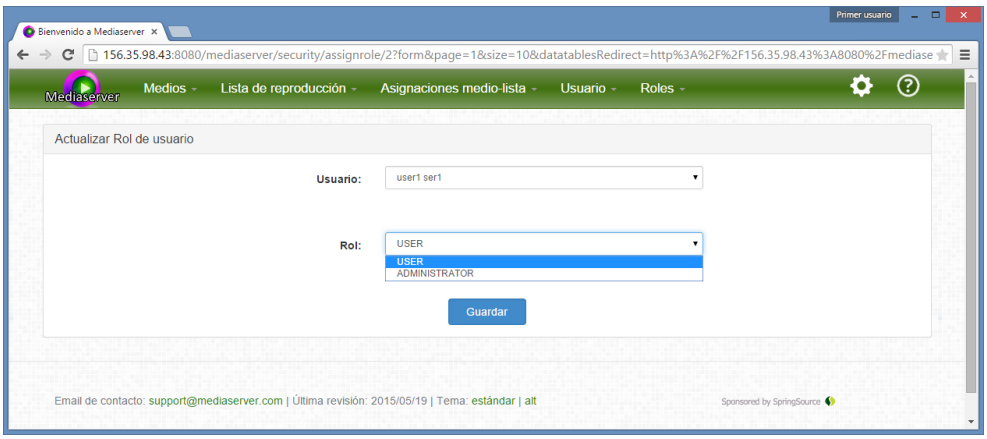


Figura 6.46 Página de editar rol de usuario

En esta página el administrador podrá editar el rol de un usuario determinado.

6.5.2 Interfaz de control remoto móvil

La interfaz de control remoto permitirá al usuario controlar la reproducción que se lleva a cabo en su navegador web, para ello ha de tener abierta la página de visualización de un medio o lista de reproducción, y configurar correctamente la aplicación móvil para conectarse al servidor utilizando las mimas credenciales con las que se ha conectado en el cliente web.

Figura 6.47 Pantalla principal de la aplicación de control remoto

La pantalla principal de la aplicación de control remoto contiene controles para enviar órdenes de control de reproducción al servidor para:

- Iniciar la reproducción.
- Pausar la reproducción.
- Parar la reproducción.

- Pasar al medio anterior y posterior.
- Cambiar el volumen.
- Cambiar el estado de progreso.
- Pasar a pantalla completa o volver a pantalla parcial.

Además dispone de un campo en el que se muestra el último mensaje transmitido correctamente por el servidor. En caso de que se produzca algún error también se mostrará ahí.

Desde esta pantalla se puede acceder la configuración de conexión a través del icono correspondiente en la esquina superior derecha de la barra de menú.

Figura 6.48 Pantalla de configuración de la aplicación de control remoto

En esta pantalla el usuario podrá configurar los parámetros de conexión al servidor estableciendo sus credencia les de usuario (email y contraseña) y la dirección del servidor al que conectarse.

6.6 Especificación Técnica del Plan de Pruebas

El proceso de pruebas se extiende durante todo el proceso de construcción del software y por tanto en esta sección describiremos cómo vamos a aplicar las pruebas diseñadas y especificaremos más otro tipo de pruebas que vamos a realizar al software.

Las máquinas sobre las que realizarán serán:

Para el papel de servidor un PC de sobremesa Dell OptiPlex 390 con las siguientes características:

- Sistema operativo: Windows 7 Profesional 64 bits
- Ram instalada: 4Gb
- Procesador Intel Core i3-2100 CPU @3.10GHz (4CPUs), ~3.1GHz
- Conexión a internet: **ping:**14ms, **bajada:** 92 Mbps, **subida**: 68 Mbps

Para el papel de cliente un PC de sobremesa Asus k5130 con las siguientes características:

- Sistema operativo: Windows 8.1
- Ram instalada: 4Gb
- Procesador Intel Celeron CPU G1620 @ 2.70GHz
- Conexión a internet: **ping:** 3 ms, **bajada:** 47 Mbps, s**ubida**: 51 Mbps

6.6.1 Pruebas Unitarias

Esta sección describiremos las pruebas unitarias se encargarán de testear los métodos de funcionalidades específicas de la aplicación.

Estas pruebas unitarias son mucho más numerosas en la parte web que en la parte móvil ya que esta segunda parte es de mucha menor envergadura y mayor sencillez.

Pruebas sobre Medios	
Prueba	**Resultado Esperado**
Introducir un nombre fuente y tipo correctos	El sistema guarda el medio correctamente.
Introducir un nombre con menos de 3 caracteres o más de 30	El sistema no acepta los datos introducidos porque no cumplen los límites establecidos.
Borrar un medio	El medio se elimina así como las apariciones en listas de reproducción en las que estuviera incluido.
Crear dos medios con los mismos datos	El sistema los guarda correctamente ya que la clave primaria es un id.

Pruebas sobre Listas de reproducción	
Prueba	**Resultado Esperado**
Crear una lista de reproducción con datos correctos	El sistema la guarda correctamente.

Crear una lista de reproducción cuyo nombre sea menor de 3 caracteres o mayor de 30	El sistema no acepta los datos introducidos porque no cumplen los límites establecidos.
Crear dos listas con los mismos datos	El sistema las guarda correctamente ya que la clave primaria es un id.
Añadir un medio varias veces a una misma lista de reproducción	Se añade correctamente.

Pruebas sobre Usuarios	
Prueba	**Resultado Esperado**
Crear un usuario con datos correctos	Se crea y guarda correctamente.
Crear dos usuarios con el mismo email	El segundo usuario no se puede guardar ya que la dirección de email ha de ser única.
Eliminar un usuario	Se elimina el usuario del sistema así como todas las listas de reproducción, medios y roles asociados.

Pruebas sobre Roles	
Prueba	**Resultado Esperado**
Creación de un rol correcto	Se crea y guarda correctamente.
Creación de un rol con una descripción de más de 200 caracteres	El sistema no lo acepta pues excede el número máximo de caracteres permitido para ese campo.

Pruebas sobre Roles de usuarios	
Prueba	**Resultado Esperado**
Asignar un rol a un usuario	El usuario adquiere el nuevo rol.
Eliminar en rol de un usuario	El usuario deja de tener el rol asociado.

Pruebas sobre Recaptcha de registro	
Prueba	**Resultado Esperado**
Introducir valor correcto de la imagen del recaptcha	El recaptcha se valida correctamente.
Introducir un valor erróneo de la imagen del recaptcha	El recaptcha detecta el error y no valida.
Cambiar el recaptcha a modo sonido	La imagen del recaptcha se sustituye por un reproductor de sonido.
Introducir la transcripción correcta del sonido	El recaptcha valida correctamente.
Introducir una transcripción incorrecta del sonido	El recaptcha detecta el error y no valida.

Pruebas sobre email de activación	
Prueba	**Resultado Esperado**
Enviar email a una dirección de correo existente	El email se envía correctamente.

Pruebas sobre obtención de la IP externa del servidor	
Prueba	**Resultado Esperado**
Obtener la IP desde servidores con distintas direcciones IP	En cada caso devuelve la IP externa correctamente.

Pruebas sobre email de restablecimiento de password	
Prueba	**Resultado Esperado**
Enviar email a una dirección de correo existente	El email se envía correctamente.

Pruebas sobre búsqueda de Medios	
Prueba	**Resultado Esperado**
Atender una petición de listado de medios para un usuario standard	Se listan los medios asociados a ese usuario concreto.
Atender a una petición de listado de medios para un usuario administrador	Se listan los medios de todos los usuarios del sistema.
Búsqueda de medios de un tipo determinado	Se listan los medios que se corresponden con el tipo especificado.
Listado filtrado de medios	Se listan los medios que contienen en alguno de sus campos la cadena de texto especificada en el filtro.

Pruebas sobre búsqueda de Listas de reproducción	
Prueba	**Resultado Esperado**
Atender una petición de listado de listas de reproducción para un usuario standard	Se listan las listas de reproducción asociadas a ese usuario concreto.
Atender a una petición de listado de listas de reproducción para un usuario administrador	Se listan las listas de reproducción de todos los usuarios del sistema.

Pruebas sobre webSockets	
Prueba	**Resultado Esperado**
Enviar un mensaje a la dirección de recepción de mensajes	El servidor recibe el mensaje y lo envía por el canal de salida adecuado al usuario que lo envió.

Pruebas sobre recepción de órdenes de control móvil	
Prueba	**Resultado Esperado**
Enviar una orden de control con credenciales correctas	El servidor recibe la orden, comprueba que las credenciales son correctas y retransmite la orden al canal de salida del cliente adecuado. Devuelve a la aplicación móvil la orden que ha transmitido.
Enviar una orden de control con credenciales incorrectas	El servidor recibe la orden, comprueba las credenciales, genera un error y lo devuelve a la aplicación móvil.

Pruebas sobre configuración de la aplicación móvil	
Prueba	**Resultado Esperado**
Cambiar los valores de configuración	Los valores se guardan correctamente.
Cerrar la aplicación y volver a abrirla	Los valores de configuración del usuario se conservan.

Pruebas sobre reproducción	
Prueba	**Resultado Esperado**
Iniciar la reproducción de un medio disponible	La reproducción se inicia.
Pausar una reproducción iniciada	La reproducción se pausa.
Pausar una reproducción parada	La reproducción continúa parada.
Parar una reproducción iniciada	La reproducción se para.
Reproducir hasta el final un medio y lista	La reproducción pasa a estado terminado.
Para una reproducción terminada	La reproducción pasa a estado parada.
Terminar de reproducir uno de los medios de una lista	Se inicia la reproducción del siguiente medio de la lista.
Reproducir medios en distintos formatos y de distintos orígenes	Se reproducen correctamente siempre que estén soportaos por el plugin de reproducción VLC.

Pruebas sobre sincronización	
Prueba	**Resultado Esperado**
Controlar reproducción de varios reproductores web abiertos en el sistema por el mismo usuario	Las órdenes se transmiten a los dos reproductores por igual.

6.6.2 Pruebas de Integración y del Sistema

Las pruebas funcionales son las que aseguran que todos los casos de uso del sistema sean satisfechos correctamente a través de la interfaz ofrecida, y por tanto se organizarán de acuerdo a ellos:

Caso de Uso 1: Registro	
Prueba	**Resultado Esperado**
Rellenar datos correctamente	El sistema los guarda correctamente, se notifica el éxito de la operación y se envía el email de activación.
Introducir un email no válido	El sistema notifica que la dirección introducida no es correcta y solicita introducirla de nuevo.
Introducir un email ya registrado	El sistema notifica que la dirección introducida no está disponible y solicita introducirla de nuevo.
Introducir una contraseña distinta en el campo el confirmación de contraseña	El sistema notifica que las contraseñas no coinciden.
Dejar campos obligatorios sin rellenar	El sistema notifica que han quedado campos en blanco y deben rellenarse.
Transcribir el recaptcha incorrectamente	El sistema notifica el error y carga un nuevo recaptcha.

Caso de Uso 2: Recuperar contraseña	
Prueba	**Resultado Esperado**
Introducir correctamente la dirección del correo de recuperación	El sistema envía correctamente el email de restablecimiento de contraseña.
Introducir una dirección incorrecta	El sistema advierte de que no se ha encontrado ninguna cuenta asociada a dicha dirección de correo.
En el formulario de restablecimiento introducir una contraseña distinta en el campo el confirmación de contraseña	El sistema notifica que las contraseñas no coinciden.
En el formulario de restablecimiento introducir correctamente los datos	La pasword es restablecida correctamente.

Caso de Uso 3: Login	
Prueba	**Resultado Esperado**
Introducir las credenciales correctas como usuario standard	El sistema redirige al usuario al listado de medios.
Introducir las credenciales correctas como usuario administrador	El sistema redirige al administrador al listado de usuarios.
Introducir unas credenciales incorrectas	Se notifica que las credenciales no son válidas.
Introducir unas credenciales correctas con un usuario bloqueado o inactivo	Se notifica el estado de bloqueo o falta de activación y se deniega el acceso.

Caso de Uso 4: Cambiar idioma

Prueba	**Resultado Esperado**
Solicitar un cambio de idioma	El idioma de la interfaz cambia y se mantiene correctamente en las siguientes peticiones.

Caso de Uso 5: Descargar aplicación móvil	
Prueba	**Resultado Esperado**
Acceder al enlace de descarga de la aplicación	Funciona el enlace y se puede descargar la aplicación correctamente.

Caso de Uso 6: Cambiar tema	
Prueba	**Resultado Esperado**
Solicitar un cambio de tema	El tema de la interfaz cambia y se mantiene correctamente en las siguientes peticiones.

Caso de Uso 7: Gestionar medios	
Prueba	**Resultado Esperado**
Crear un nuevo medio introduciendo nombre y ubicación, desde la página de listado.	El medio se crea y añade correctamente a la lista.
Crear un nuevo medio utilizando la página de creación	El medio se crea y a continuación se muestra.
Crear un nuevo medio utilizando la herramienta de sugerencia	El formulario se autocompleta correctamente y el medio se añade de manera correcta.
Editar un medio	Los cambios se guardan correctamente.
Borrar uno o más medios	Los medios se borran correctamente.
Cancelar la edición o borrado	La operación se cancela correctamente.
Buscar medios por tipo	Se muestra el listado de medios sólo del tipo especificado.
Filtrar medios	Se muestra el listado de medios pero sólo aquellos que contienen la cadena de texto especificada en el filtro en alguno de sus campos.
Cambiar número de registros a mostrar	Se cambia correctamente el número máximo de medios que se muestran en cada página.

Caso de Uso 8: Gestionar listas de reproducción	
Prueba	**Resultado Esperado**
Crear una nueva lista de reproducción desde la página de listado	Se crea y se añade correctamente a la tabla de listas.
Crear una nueva lista utilizando la página de creación	La lista se crea y a continuación se muestra.
Editar lista	Los cambios se guardan correctamente.
Borrar una o más listas	Se borran correctamente.
Cancelar la edición o borrado	La operación se cancela correctamente.
Filtrar listas	Se muestra el listado de elementos que contienen la cadena de texto especificada en el filtro en alguno de sus campos.

Cambiar número de registros a mostrar	Se cambia correctamente el número máximo de listas que se muestran en cada página.

Caso de Uso 9: Gestionar elementos de una lista de reproducción	
Prueba	**Resultado Esperado**
Añadir un medio a la lista de reproducción	El medio se añade correctamente.
Editar un medio de la lista	Las modificaciones se guardan correctamente.
Eliminar un medio de la lista	El medio se elimina de la lista.
Cancelar la edición o borrado	La operación se cancela correctamente.

Caso de Uso 10: Reproducir	
Prueba	**Resultado Esperado**
Reproducir un medio	El medio se reproduce correctamente.
Reproducir una lista	Se reproduce correctamente.
Reproducir un medio con una fuente errónea o no disponible	El medio no se reproduce y se notifica el error.
Reproducir una lista vacía	No se muestra el reproductor y solo permite añadir medios a la lista.
Avanzar o retroceder de medio en una lista	Se cambia de medio correctamente.
Utilizar varios reproductores simultáneamente con la misma cuenta	La reproducción se sincroniza compartiendo las órdenes que reciben y envían.
Parar reproducción	La reproducción se para.
Pausar reproducción	La reproducción se pausa.

Caso de Uso 11: Cambiar contraseña	
Prueba	**Resultado Esperado**
Rellenar el formulario correctamente	Se cambia la contraseña y se notifica el éxito de la operación.
Introducir incorrectamente la contraseña actual	Se notifica el error y se pide introducirla correctamente.
Introducir nuevas contraseñas que no coincidan	Se notifica el error y se pide que coincidan.

Caso de Uso 12: Cerrar sesión	
Prueba	**Resultado Esperado**
Cerrar sesión	La sesión finaliza correctamente, se redirige a la página de login como usuario anónimo y el menú cambia con las opciones disponibles en ese momento.

Caso de Uso 13: Consultar ayuda	
Prueba	**Resultado Esperado**
Consultar ayuda	La ayuda se muestra correctamente.

Caso de Uso 14: Gestionar usuarios	
Prueba	**Resultado Esperado**
Crear un nuevo usuario correctamente	El usuario es creado y guardado en la base de datos.
Introducir errores u omisiones en la creación de un nuevo usuario	El sistema notifica los errores y requiere su corrección para poder guardar el usuario.
Editar los datos de un usuario	Los cambios se guardan correctamente.
Eliminar un usuario	El usuario se elimina correctamente.
Bloquear usuario	El usuario queda bloqueado y se impide su inicio de sesión sistema.

Caso de Uso 15: Gestionar roles	
Prueba	**Resultado Esperado**
Asignar un rol a un usuario	El usuario adquiere el nuevo rol.
Eliminar un rol de un usuario	El usuario pierde el rol.

Caso de Uso 16: Gestionar asignaciones medio-lista	
Prueba	**Resultado Esperado**
Editar una asignación	Los cambios se guardan correctamente.
Eliminar una asignación	La asignación se elimina correctamente.

Caso de Uso 17: Configurar preferencias de control remoto	
Prueba	**Resultado Esperado**
Introducir datos correctamente	La configuración se guarda correctamente y permite la transmisión de órdenes.
Rellenar datos de forma incorrecta	Las órdenes no se transmiten.

Caso de Uso 18: Controlar reproducción remota	
Prueba	**Resultado Esperado**
Controlar un reproductor remotamente	Las órdenes se transmiten correctamente y el estado del reproductor cambia según ellas, cambio de medio, estado de reproducción cambio de volumen, cambio a pantalla completa.
Controlar varios reproductores remotamente	Las órdenes de control se transmiten a todos los reproductores asociados a la cuenta de manera sincronizada.
Intentar reproducir sin conexión	Se notifica el error de falta de conexión.

6.6.3 Pruebas de Usabilidad y Accesibilidad

6.6.3.1 Pruebas de Usabilidad

Las pruebas de usabilidad tratan de evaluar la aplicación en su funcionamiento real. Debemos realizar un conjunto de pruebas para verificar que las distintas partes del programa se pueden usar adecuadamente. Los objetivos son:

- Mejorar la calidad de la interacción de los usuarios con nuestra aplicación.
- Reducir el tiempo necesario para hacer las distintas tareas en la aplicación y de esta forma aumentar la productividad.

Se pretenderá comprobar la aplicación con una serie de personas de distintos perfiles y niveles de uso de medios informáticos.

Se buscará algún usuario de cada uno de los siguientes perfiles:

- Persona inexperta en informática.
- Persona con conocimientos medios de informática e internet.
- Experto en informática.

Las pruebas se realizarán según la disponibilidad de cada usuario en su hogar, o en su lugar de trabajo.

Previamente a la realización de las pruebas se contestarán unas breves preguntas generales sobre experiencia previa.

Las pruebas consistirán en la realización de una serie de actividades utilizando la aplicación desarrollada en el proyecto y la respuesta a un cuestionario sobre experiencias durante la prueba. También se observará directamente a los usuarios y recogerán notas y opiniones durante y posteriormente a las actividades.

6.6.3.1.1 Formulario para la evaluación de la usabilidad

Datos generales

Conexión internet
1. Modem 2. ADSL 3. RDSI 4. Cable 5. Satélite 6. No lo sé

Conocimientos

Usa un ordenador frecuentemente
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Parte de mi trabajo 2. Ocio principalmente 3. Ocio y trabajo 4. Aplicaciones estilo Office y navegación
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros para tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias

Tareas a realizar

Usuario anónimo
1. Leer la página principal y resumir el objetivo de la aplicación 2. Encontrar el email de contacto 3. Cambiar el idioma 4. Cambiar el tema de la aplicación 5. Registrarse
Usuario registrado
1. Crear un medio 2. Editar el nombre del medio 3. Visualizar el medio 4. Borrar el medio 5. Crear una lista de reproducción 6. Añadir un medio a la lista 7. Visitar la ayuda 8. Cerrar sesión
Usuario administrador
1. Crear un usuario

2. Editar los datos del usuario
3. Bloquear el usuario creado
4. Cambiar el rol del usuario
5. Eliminar el usuario
6. Eliminar una asignación medio-lista

Usuario móvil

1. Descargar e instalar la aplicación móvil
2. Configurar la aplicación con sus credenciales y dirección del servidor
3. Controlar una reproducción activa en su navegador web

Las tareas de administrador no se realizaran con usuarios del perfil inexperto en informática, ya que su encomienda no está dirigida a este tipo de perfil.

Valoración

General	**Muy cierto**	**Cierto**	**Poco cierto**	**Nada cierto**
Creo que me gustará visitar con frecuencia el sitio				
Encontré el sitio demasiado complejo				
Pienso que el sitio es fácil de usar				
Creo que necesitaría ayuda de un experto para utilizar el sitio				
Hay demasiada inconsistencia en la web				
Creo que la mayoría de la gente podría usar el sitio web rápidamente				
He encontrado el sitio web incómodo de usar				
Necesitaría aprender muchas cosas antes de manejar el sitio web				
Facilidad de uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funcionan las tareas como espera?				
¿El tiempo de respuesta es muy				

prolongado?				
Aspectos gráficos	**Muy adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra				
Los iconos e imágenes usados				
Los colores empleados				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones:				

Los resultados de las pruebas se analizarán conjuntamente para así establecer una conclusión respecto a cuatro aspectos de usabilidad.

6.6.3.2 Pruebas de Accesibilidad

Para las pruebas de accesibilidad utilizaremos la metodología propuesta por la WAI (Web Accessibility Initiative) o Oniciativa de Accesibilidad Web.

Realizaremos una revisión con los siguientes pasos, previa selección de las herramientas a utilizar:

1. Selección de conjunto de páginas representativas.
2. Evaluaciones con herramientas automáticas.
3. Evaluación manual.
4. Análisis de resultados.

A modo de resumen rellenaremos un *checklist* para verificar las pautas *WCAG 1.0* de accesibilidad de la aplicación *web* basándonos en los resultados de las evaluaciones automáticas y Manuales.

No se ha profundizado mucho en las pruebas de accesibilidad móvil ya que sus pautas promulgadas por la ONCE no parecen estar aun claramente establecidas, estandarizadas y extendidas.

6.6.4 Pruebas de Rendimiento

Para evaluar el rendimiento del sistema se hará uso de la herramienta web http://www.webpagetest.org/ la cual realizará un test midiendo los tiempos de carga y tamaños de los distintos recursos que se solicitan al servidor con cada intento de cargar una página.

Esto nos dará idea sobre cuánto tiempo invierte el cliente web del usuario para poder mostrar las páginas de nuestra aplicación, y en qué clase de recursos se gasta ese tiempo.

Capítulo 7. Implementación del Sistema

7.1 Lenguajes de Programación

En este apartado se hace una descripción de los lenguajes de programación y marcado usados en el desarrollo de este proyecto.

7.1.1 Java

La parte central del proyecto se ha realizado fundamentalmente usando Java como lenguaje de programación, en su versión 7.

La elección ha venido motivada por los conocimientos previos y experiencia personal con el lenguaje en esta plataforma por una parte; por otra por las tecnologías relacionadas a utilizar como es el caso de Android cuyo desarrollo se realiza principalmente en este lenguaje; y finalmente por el objetivo de aprender y ampliar conocimientos en frameworks de desarrollo muy usados en la actualidad como es el caso de Spring, que también tiene este lenguaje como base.

Java es un lenguaje de programación de propósito general muy extendido con gran importancia en el ámbito de Internet, desarrollado inicialmente por la compañía Sun Microsystems, adquirida posteriormente por Oracle.

Una de las ventajas características ventajosas que ofrece el lenguaje, y que se ha tenido en cuenta a la hora de elegirlo como base de desarrollo del proyecto es su independencia de la plataforma. Esto quiere decir que sin necesidad de modificación en su código, los programas funcionan en múltiples sistemas operativos como Windows, Linux, Apple, gracias a la Máquina virtual Java implementada para cada uno de ellos. Como consecuencia, la aplicación podría desplegarse y funcionar correctamente en distintos servidores, sobre sistemas operativos diversos, lo cual también es objetivo del presente proyecto.

De entre las bibliotecas utilizadas en el proyecto relacionadas con este lenguaje hemos de destacar la procedente del framework Spring 4 y Datatables para la aplicación web, y la biblioteca de Android, para la aplicación móvil.

7.1.2 CSS

El lenguaje CSS de sus siglas en inglés Cascading Style Sheets (hojas de estilo en cascada) sirve para definir la presentación de un documento estructurado escrito en HTML o XML de cara a los agentes de usuario o navegadores. Está estandarizado por el W3C (World Wide Web Consortium) y que en el proyecto se ha utilizado en su versión 3.

Este lenguaje viene casi siempre de la mano de HTML por lo que su elección no ha requerido de excesivas consideraciones.

El objetivo del CSS es separar la estructura de un documento de su presentación. La información de estilo puede ser adjuntada tanto como un documento separado o en el mismo documento HTML.

Entre las ventajas de su utilización podríamos señalar:

- Control centralizado de la presentación de un sitio completo, con lo que se favorece la reutilización y agiliza la actualización.
- El estilo puede ser modificado localmente por los navegadores, y adaptarse a las necesidades específicas de los usuarios, aumentando el nivel de accesibilidad.
- Una página puede disponer de distintas hojas de estilo según el dispositivo o sus características, adaptándose a una disposición óptima para caso.
- Los documentos HTML son más claros, reducidos y comprensibles al prescindir de definiciones de estilo incrustadas.

Se han empleado recursos del framework de desarrollo web Bootstrap, que incluye una gran biblioteca de CSS aplicable mediante clases a los elementos del proyecto.

7.1.3 HTML

Acrónimo de HyperText Markup Language (lenguaje extensible de marcado de hipertexto), el HTML es el lenguaje de marcado estándar para las páginas web. Para el proyecto se ha utilizado la versión 5 de octubre del 2014

El HTML basa su filosofía de desarrollo en la referenciación, para añadir un elemento externo a la página, este no se incrusta directamente en el código sino que se hace una referencia a la ubicación de dicho elemento mediante texto.

Esto es especialmente importante en el proyecto ya que los medios y recursos que utilizará la aplicación web para la reproducción, no están alojados en el propio servidor de la aplicación sino en servidores de terceros, disponibles en internet.

HTML es el lenguaje estándar para elaboración de páginas web, por lo que su utilización para el desarrollo de un proyecto web no requiere ninguna elección.

7.1.4 Javascript

JavaScript es un lenguaje de programación orientado a objetos, basado en prototipos, imperativo, dinámico y débilmente tipado. Su última versión es la 1.8.5, y es la empleada en el proyecto.

Utilizado principalmente en el lado del cliente y ampliamente extendido combinado junto con el HTML y CSS de las páginas web.

Su utilización en el proyecto viene motivada por la necesidad de incorporar un comportamiento dinámico en el cliente web, ya que han de manejarse reproducciones de video y audio controladas remotamente.

La comunicación bidireccional de WebSockets que en el servidor se implementa utilizado lenguaje java, es implementada en el cliente mediante JavaScript, recogiendo los mensajes que envía el servidor para controlar la reproducción web, y también enviando al servidor los eventos pertinentes del reproductor web para la sincronización. Además el control del reproductor se hace mediante su API de llamadas JavaScript.

Aparte de la reproducción, JavaScript también se usa en el proyecto para realizar las llamadas peticiones AJAX, que permiten el intercambio de información con el servidor para actualizar información de la página sin necesidad de recargar la página completa, mejorando el rendimiento y reduciendo tanto tiempo de espera como carga de trabajo en el servidor.

Una gran ventaja de JavaScript es la liberación de procesamiento en el servidor, distribuyendo los procesos específicos necesarios para cada cliente, en el navegador de cada cliente específico.

La biblioteca más importante utilizada en este ámbito ha sido jQuery 1.11 la cual aumenta el grado de concisión del lenguaje, permitiendo escribir el mismo código que se haría con JavaScript puro de manera más sencilla, compacta y con un mejor soporte por parte de los navegadores web.

También se han utilizado bibliotecas de javascript de Bootstrap para añadir dinamismo a la interfaz, y otras bibliotecas para la implementación del protocolo STOMP sobre WebSockets para la comunicación de control multimedia con el servidor.

7.1.5 JSPX

JSPX es uan versión de Java Servlet Pages que sigue el formato XML.

Se trata de una tecnología para desarrollar páginas web dinámicas basadas en HTML y XML.

La principal ventaja de JSP frente a otros lenguajes es que funciona con Java, lenguaje de propósito general que excede el mundo web y que es apto para crear clases que manejen lógica de negocio y acceso a datos. Esto permite separar en niveles las aplicaciones web, dejando la parte encargada de generar el documento HTML en el archivo JSP.

JSPX se ha utilizado para crear todas las páginas que comprenden la interfaz de la aplicación web.

7.1.6 XML

XML, eXtensible Markup Language (lenguaje de marcas extensible) es un lenguaje de marcas desarrollado por el W3C para almacenar datos de manera legible. Se ha empleado en su versión 1.0

El lenguaje XML se ha utilizado en numerosas partes del proyecto, tanto para descripción de interfaz gráfica en la aplicación móvil, como para guardado de datos, ficheros de configuración, descripción de elementos de interfaz en archivos de tags, archivos de configuración del framework, etc.

7.2 Herramientas y Programas Usados para el Desarrollo

En este apartado se hace una descripción de las herramientas de desarrollo, sistemas adicionales existentes, complementos y otros productos software que hemos necesitado para la implementación de nuestro sistema.

7.2.1 AppServ

Appserv es un programa que instala y configura rápidamente los servicios más comunes para trabajar con páginas web y bases de datos. Concretamente, Apache 2.2.8, PHP 6.0.0-dev, MySQL 6.0.4a y phpMyAdmin 2.10.3.

Se ha empleado especialmente para el manejo y despliegue de la base de datos MySQL donde se guardan los datos de la aplicación y su gestión mediante phpMyAdmin para supervisión y control.

7.2.2 Eclipse Luna para java

El entorno de desarrollo integrado Eclipse se ha utilizado durante el desarrollo de la aplicación móvil combinado con el plugin para desarrollo en Android ADT (Android Development Tools)

7.2.3 Enterprise Architect 8.0

Utilizado para elaborar los diagramas tanto en la fase de análisis como de diseño y documentación.

7.2.4 Navegador Google Chrome

El navegador de google en su versión 43.0.2357.81 ha sido empleado constantemente para probar la aplicación web. Esta versión es la actual en el momento de documentación pero ha variado con las actualizaciones automáticas desde el inicio del proyecto.

Utilizado no solo como cliente standard, sino como herramienta de desarrollo para optimizar el aspecto con cambios de CSS, depuración de código JavaScript gracias al inspector y depurador de código, testeo de rendimiento y tiempos de carga, etc.

7.2.5 Paint.NET

Usado para la edición de imágenes e iconos del proyecto, dado lo sencillo y liviano de su manejo y calidad de resultados.

7.2.6 Spring Tool Suite

En su versión 3.6.0 se ha utilizado como entorno de desarrollo integrado para todo el proceso de implementación implementación del sistema web.

Es un conjunto de herramientas basadas en el IDE Eclipse preparado para el desarrollo de aplicaciones web con el framwork Spring.

7.2.7 Tomcat

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation.

Tomcat es un servidor web sobre el que se despliega la aplicación Web.

Ha sido utilizada la versión 7.0.50 dada su compatibilidad con WebSockets y Datatables.

7.2.8 VLC

VLC es un reproductor multimedia libre y de código abierto multiplataforma y un «framework» que reproduce la mayoría de archivos multimedia, así como DVD, Audio CD, VCD y diversos protocolos de transmisión.

Este reproductor incorpora un plugin NPAPI que es el encargado de reproducir los recursos multimedia en la aplicación web, por lo que es necesario que los clientes instalen este programa en las máquinas donde quieran utilizar la aplicación web.

Se ha utilizado la versión 2.2.1.

7.3 Creación del Sistema

Aquí describiremos todos los aspectos con los que nos hemos encontrado durante la implementación

7.3.1 Problemas Encontrados

A continuación reenumeran algunos de los problemas encontrados en el desarrollo y la solución que se les ha dado.

7.3.1.1 Adaptación del código generado por la herramienta Spring Roo a las necesidades específicas de la aplicación

Originalmente al generar código mediante el intérprete de comandos de Spring Roo, no se muestran todos los ficheros y no se ve todo el código en el IDE. Concretamente se ocultan los archivos aspectJ, no sin razón pues estos son gestionados automáticamente por el framwork y las modificaciones en ellos pueden ser sobrescritas sin aviso previo.

No obstante es bueno mostrarlos e inspeccionarlas para encontrar en ellos las implementaciones de funcionalidades que se deben llevar a las clases principales para adaptar a las necesidades concretas del programa.

7.3.1.2 Comunicación iniciada por parte del servidor al cliente web para la transmisión de órdenes de control de reproducción.

La comunicación de control típica de la aplicación se inicia remotamente en el dispositivo móvil y se ve reflejada en el navegador web, por lo que a efectos de comunicación entre el servidor y el cliente web es necesario que esa comunicación se inicie en el servidor.

Esta necesidad no se ve cubierta por el modelo de comunicación típico de petición- respuesta iniciado desde el cliente, ya que no hay una petición previa.

Para solucionar esta dificultad se han estudiado varias alternativas, desde el polling o muestre periódico, al long polling o muestreo con retención, hasta la finalmente adoptada WebSockets.

El problema de las dos primeras era un mayor consumo de recursos respecto a la tercera y más adecuada, aunque su dificultad de implementación y requerimientos para la incorporación dentro de los frameworks utilizados fuera mayor.

Para la parte del cliente se encontró una biblioteca de comunicación utilizando el protocolo STOMP sobre web sockets, con control sobre desconexión y pérdida de clientes que se podía integrar perfectamente con el otro extremo implementado en java desde la parte servidora.

7.3.1.3 Reproducción multimedia web multiformato

Si bien es cierto que el nuevo HTML5 incorpora controles para la incorporación y reproducción de recursos de audio y video dentro de las páginas web, la cantidad de formatos aceptados de esta forma integrada está muy limitado, y puesto que se pretendía una compatibilidad con un gran número de formatos y orígenes se optó por la utilización del plugin web NPAPI de VLC controlado por su API desde JavaScript.

También se consideraron otras opciones como WebChiera que tal vez se implementen en el futuro para resolver el próximo bloqueo de plugins NPAPI por parte de navegadores como Google Chorme

7.3.1.4 Bloqueo de plugins NPAPI

A un escaso mes para el final del desarrollo del proyecto, Google liberó una actualización de de su navegador en el que la utilización de plugins NPAPI venía desactivado por defecto, lo cual suponía una necesidad crítica para el funcionamiento de la aplicación en dicho navegador. No obstante los plugins de ese tipo se podían aún habilitar en una página de control del navegador.

Buscando información sobre el tema se encontró que cualquier soporte para plugins NPAPI sería finalmente eliminado por completo del navegador Chrome para septiembre del 2015, por lo que a partir de entonces sería necesario encontrar e incorporar algún otro sistema de reproducción web que no basara su funcionamiento en ese tipo de plugin.

De entre los encontrados, algunos informaban que estaban en proceso de solución de tal inconveniente, pero ninguno estaba preparado ya para un recambio inmediato en el proyecto, por lo que se continuó utilizando VLC, localizando perfectamente y minimizando los archivos en los que se hace uso del plugin para que cuando sea el momento de sustituirlo, pueda hacerse de la manera más rápida e inocua posible.

7.3.1.5 Escalabilidad y despliegue distribuido de la aplicación

Para poder escalar la aplicación desplegando varios servidores se pensó en la incorporación de un servidor de balanceo intermedio como Haproxy, pero se encontró el problema de que al balancear las peticiones alternativamente, tras hacer login en uno de los servidores, la siguiente petición se realizaba a otro servidor en el que no se tenía sesión y el cliente era redirigido nuevamente a la página de login.

Esto se solucionó marcando las peticiones con una cookie adicional en el balanceador que identificaba el servidor que había atendido al cliente por primera vez en esa sesión y por lo tanto cada cliente pasaba a ser atendido siempre por el mismo servidor.

No obstante un nuevo problema surgió al incorporar el control remoto móvil cuya sesión es independiente de la sesión web, y al considerar la posibilidad de que un mismo cliente pudiera tener varias sesiones en distintos servidores, que también sería necesario sincronizar.

La opción que se tomó por el momento hasta poder realizar una implementación de control y sincronización tal vez más compleja, es la de mantener un solo servidor de aplicación como servidor principal, y los otros servidores adicionales como servidores de respaldo en caso de que el principal falle.

Esta redirección sería llevada a cabo por el servidor intermedio al cual se dirigirían siempre los clientes.

7.3.1.6 Obtención de la dirección IP pública del servidor para el envío de enlaces de activación y recuperación

Para incorporar a los correos de activación e cuentas y recuperación de passwords un enlace al servidor sin disponer de un dominio público ni conocer de antemano la dirección IP pública del servidor era necesario averiguar esta de algún modo.

El problema se solucionó consultando a servidores externos por la dirección del servidor concreto de la aplicación en ese momento con lo cual se obtiene la IP pública y se puede generar el enlace pertinente en el momento deseado.

7.3.1.7 Utilización de servicios de computación en la nube

Los servicios gratuitos de computación en la nube ofrecen la posibilidad de disponer de máquinas virtuales con sistemas operativos a elección e ips públicas y privadas para desplegar aplicaciones, de forma distribuida, bases de datos y realizar pruebas.

El problema con alguno de estos servicios, como es el caso del utilizado para el proyecto: OpenStack es la baja fiabilidad en cuanto a disponibilidad, largo tiempo hasta el restablecimiento de los servicios, y excesivamente frecuentes pérdidas y reseteos de las máquinas virtuales creadas. Esto echa por tierra todo el trabajo de instalaciones y configuraciones y despliegues que se hayan realizado en las mismas y obligando a repetir todo ese trabajo para reestablecer su estado.

7.3.1.8 Versiones del servidor de aplicaciones web

Aunque idealmente cada versión de un servidor de aplicaciones es retrocompatible con las anteriores versiones, se ha encontrado que el servidor Tomcat presenta algunos problemas en este aspecto, concretamente para soportar por una parte WebSockets y por otra el framework web de representación de datos DataTables, lo cual ha obligado durante el desarrollo y despliegue a probar múltiples versiones del mismo, de las series 7 y 8 en donde unas características funcionaban, pero otras no hasta dar con el que finalmente se ha empleado, el 7.0.50.

7.3.1.9 Integración de WebSockets y gv-NIX Srping Roo mediante Maven

Debido a la reciente incorporación de la tecnología de WebSockets al framework Spring en su versión 4, la tecnología no era soportada por gv-NIX basado en Spring 3, framework con el cual se había estado realizando el desarrollo del proyecto.

Por ello se vio necesario un cambio framework Spring base en el proyecto y utilizando el gestor de dependencias Maven se realizó un trabajoso reemplazo de las dependencias implicadas y numerosas limpiezas y reconstrucciones del proyecto para que todas ellas casaran en su sitio hasta que finalmente se logó integrar gv-NIX con Spring 4.

7.3.1.10 Fiabilidad de la reproducción Web

El plugin de reproducción web de VLC presenta algunos problemas de estabilidad cuando se acumulan muchas órdenes o al recibir una orden pasado un tiempo, lo que hace que el navegador presente un informe de tal circunstancia y sea necesario recargar la página para que la reproducción prosiga.

Probablemente estas inestabilidades sean una de las causas por las que Google trata de acabar con esta clase de plugins que pudieran dar la impresión de son inestabilidad de su navegador.

Como se ha mencionado para otros casos, se han valorado otras opciones de plugins y en cuanto alguno de ellos ofrezca una alternativa más estable, que no requiera NPAPI pero con gran compatibilidad de formatos, se aplicará al proyecto sustituyendo el reproductor actual.

7.3.1.11 Sistema de sugerencia de nuevos medios

Este problema hace referencia a la necesidad que había originalmente de buscar fuentes de nuevos medios fuera del entorno de la propia aplicación, teniendo que abrir páginas como YouTube u otras fuentes de recursos para copiar de ellas la URL del recurso y pegarlo en el formulario de inserción de nuevos medios.

La solución adoptada fue la incorporación de un formulario de diferencia que realiza una búsqueda personalizada de Google en unas páginas de recursos establecidas de antemano, presentando los resultados en un *overlay* o capa superior a la propia página de la aplicación.

Al seleccionar el usuario uno de los resultados, los datos del mismo son ya pasados al formulario de inserción, facilitando así la incorporación de nuevos medios de los que sólo se conoce el nombre sin necesidad de buscarlos en páginas externas.

7.3.1.12 Modificación no deseada del código de la aplicación por la herramienta de desarrollo

Llegado un punto en el desarrollo fue necesario incorporar código de otro proyecto desarrollado con anterioridad y empezaron a producirse modificaciones automáticas no

deseadas por parte de la herramienta de control y generación, que provocaban errores de compilación y retrasaban el trabajo.

Por esto llegado un punto se decidió prescindir del control y actualización de la herramienta generadora de código, haciendo un control manual del mismo.

7.3.1.13 Desarrollo con Phonegap para la aplicación móvil de control

Tras tener una parte significativa de la aplicación web funcional, se trató de llevar esta misma aplicación al dispositivo móvil haciendo uso del framework de desarrollo móvil multiplataforma Phonegap, basado en HTML, CSS y JQuery, con la esperanza de que sería fácil y rápido obtener una aplicación móvil funcional equivalente a la versión web que ya se tenía.

Esto no fue sencillo sobre todo al tratar con el sistema de seguridad de Spring. Se trató de utilizar un filtro de CORS, o compartición de recursos de origen cruzado, para obtener recursos del servidor y crear a partir de ellos las vistas adecuadas para la aplicación móvil, pero los resultados no fueron como se esperaba.

Finalmente se optó por la implementación del control remoto en Android nativo.

7.3.1.14 Riesgos no contemplados

Aunque se contempló cierto número de riesgos que podían tener influencia en el transcurso del proyecto, tal como enfermedad, o problemas con el transporte, incompatibilidades entre tecnologías… aparecieron también otros que no se habían contemplado como es el caso de la realización de obras de mantenimiento en la oficina en la que se desarrolló el proyecto y que hizo perder algún día de desarrollo debido al alto nivel de ruidos.

7.3.2 Descripción Detallada de las Clases

Para una más fácil lectura del documento, la descripción completa de las clases se encuentra en el javadoc presente en el apéndice 13.6.

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

A continuación se detallan los resultados de las pruebas unitarias antes descritas realizadas durante el desarrollo del proyecto.

Aunque el resultado tras las correcciones de todas las pruebas fue positivo, se muestra con resultado negativo también aquellas que no fueron superadas a la primera y se explican las correcciones que fueron necesarias.

Pruebas sobre Medios		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Introducir un nombre fuente y tipo correctos	El sistema guarda el medio correctamente.	
Introducir un nombre con menos de 3 caracteres o más de 30	El sistema no acepta los datos introducidos porque no cumplen los límites establecidos.	
Borrar un medio	El medio se elimina así como las apariciones en listas de reproducción en las que estuviera incluido.	
Crear dos medios con los mismos datos	El sistema los guarda correctamente ya que la clave primaria es un id.	

Inicialmente no se comprobaba el borrado de todos los elementos PlayListMedia vinculados al medio antes de borrarlo, por lo que para mantener la integridad referencial, el borrado fallaba. Tras las modificaciones, se comprueba que el medio no tenga referencias, y se eliminan todos los objetos PlayListMedia asociados, con lo que el borrado funciona correctamente, y ninguna lista de reproducción contiene medios inexistentes

Pruebas sobre Listas de reproducción		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Crear una lista de reproducción con datos correctos	El sistema la guarda correctamente.	
Crear una lista de reproducción cuyo nombre sea menor de 3 caracteres o mayor de 30	El sistema no acepta los datos introducidos porque no cumplen los límites establecidos.	
Crear dos listas con los mismos datos	El sistema las guarda correctamente ya que la clave primaria es un id.	
Añadir un medio varias veces a una misma lista de reproducción	Se añade correctamente.	

Pruebas sobre Usuarios		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Crear un usuario con datos correctos	Se crea y guarda correctamente.	
Crear dos usuarios con el mismo email	El segundo usuario no se puede guardar ya que la dirección de email ha de ser única.	
Eliminar un usuario	Se elimina el usuario del sistema así como todas las listas de reproducción, medios y roles asociados.	

Inicialmente fallaba el borrado por mantener la integridad referencial, ya que no se comprobaba que el usuario tuviera medios o listas asociados. Tras las modificaciones se comprueba y se eliminan en caso de que existan.

Pruebas sobre Roles		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Creación de un rol correcto	Se crea y guarda correctamente.	
Creación de un rol con una descripción de más de 200 caracteres	El sistema no lo acepta pues excede el número máximo de caracteres permitido para ese campo.	

Pruebas sobre Roles de Usuarios		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Asignar un rol a un usuario	El usuario adquiere el nuevo rol.	
Eliminar en rol de un usuario	El usuario deja de tener el rol asociado.	

Pruebas sobre Recaptcha de registro		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Introducir valor correcto de la imagen del recaptcha	El recaptcha se valida correctamente.	
Introducir un valor erróneo de la imagen del recaptcha	El recaptcha detecta el error y no valida.	
Cambiar el recaptcha a modo sonido	La imagen del recaptcha se sustituye por un reproductor de sonido.	
Introducir la transcripción correcta del sonido	El recaptcha valida correctamente.	
Introducir una transcripción incorrecta del sonido	El recaptcha detecta el error y no valida.	

Pruebas sobre email de activación		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Enviar email a una dirección de correo existente	El email se envía correctamente.	

Inicialmente hubo problemas con el envío de correo, por una parte por la implementación del método donde no es establecían todos los parámetros necesarios para el envió, y posteriormente por la configuración del antivirus de la máquina servidora que bloqueaba el envío.

Pruebas sobre obtención de la IP externa del servidor		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Obtener la IP desde servidores con distintas direcciones IP	En cada caso devuelve la IP externa correctamente.	

Pruebas sobre email de restablecimiento de password		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Enviar email a una dirección de correo existente	El email se envía correctamente.	

Pruebas sobre búsqueda de Medios		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Atender una petición de listado de medios para un usuario standard	Se listan los medios asociados a ese usuario concreto.	
Atender a una petición de listado de medios para un usuario administrador	Se listan los medios de todos los usuarios del sistema.	
Búsqueda de medios de un tipo determinado	Se listan los medios que se corresponden con el tipo especificado.	
Listado filtrado de medios	Se listan los medios que contienen en alguno de sus campos la cadena de texto especificada en el filtro.	

Pruebas sobre		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Atender una petición de listado de listas de reproducción para un usuario standard	Se listan las listas de reproducción asociadas a ese usuario concreto.	
Atender a una petición de listado de listas de	Se listan las listas de reproducción de todos los usuarios del sistema.	

reproducción para un usuario administrador		

Pruebas sobre webSockets		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Enviar un mensaje a la dirección de recepción de mensajes	El servidor recibe el mensaje y lo envía por el canal de salida adecuado al usuario que lo envió.	

Pruebas sobre recepción de órdenes de control móvil		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Enviar una orden de control con credenciales correctas	El servidor recibe la orden, comprueba que las credenciales son correctas y retransmite la orden al canal de salida del cliente adecuado. Devuelve a la aplicación móvil la orden que ha transmitido.	
Enviar una orden de control con credenciales incorrectas	El servidor recibe la orden, comprueba las credenciales, genera un error y lo devuelve a la aplicación móvil.	

Pruebas sobre configuración de la aplicación móvil		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Cambiar los valores de configuración	Los valores se guardan correctamente.	
Cerrar la aplicación y volver a abrirla	Los valores de configuración del usuario se conservan.	

Pruebas sobre reproducción		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Iniciar la reproducción de un medio disponible	La reproducción se inicia.	
Pausar una reproducción iniciada	La reproducción se pausa.	
Pausar una reproducción parada	La reproducción continúa parada.	
Parar una reproducción iniciada	La reproducción se para.	
Reproducir hasta el final un medio y lista	La reproducción pasa a estado terminado.	

Para una reproducción terminada	La reproducción pasa a estado parada.	
Terminar de reproducir uno de los medios de una lista	Se inicia la reproducción del siguiente medio de la lista.	
Reproducir medios en distintos formatos y de distintos orígenes	Se reproducen correctamente siempre que estén soportaos por el plugin de reproducción VLC.	

Pruebas sobre sincronización		
Prueba	**Resultado Obtenido**	**Resultado Esperado**
Controlar reproducción de varios reproductores web abiertos en el sistema por el mismo usuario	Las órdenes se transmiten a los dos reproductores por igual.	

8.2 Pruebas de Integración y del Sistema

A continuación se muestran los resultados de las pruebas de integración antes descritas organizadas según los casos de uso

Aunque el resultado tras las correcciones de todas las pruebas fue positivo, se muestra con resultado negativo también aquellas que no fueron superadas a la primera y se explican las correcciones que fueron necesarias.

Caso de Uso 1: Registro		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Rellenar datos correctamente	El sistema los guarda correctamente, se notifica el éxito de la operación y se envía el email de activación.	
Introducir un email no válido	El sistema notifica que la dirección de correo es errónea.	
Introducir un email ya registrado	El sistema notifica que la dirección introducida no está disponible y solicita introducirla de nuevo.	
Introducir una contraseña distinta en el campo el confirmación de contraseña	El sistema notifica que las contraseñas no coinciden.	
Dejar campos obligatorios sin rellenar	El sistema notifica que han quedado campos en blanco y deben rellenarse.	
Transcribir el recaptcha incorrectamente	El sistema notifica el error y carga un nuevo recaptcha.	

Inicialmente no se comprobaba que el email estuviera correctamente formado. Tras las modificaciones se comprueba que al menos tenga el formato de una dirección de correo, con la "@" intermedia. Aunque sigue sin poder saberse si la dirección existe realmente hasta que se envía el correo de activación.

Caso de Uso 2: Recuperar contraseña		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Introducir correctamente la dirección del correo de recuperación	El sistema envía correctamente el email de restablecimiento de contraseña.	
Introducir una dirección incorrecta	El sistema advierte de que no se ha encontrado ninguna cuenta asociada a dicha dirección de correo.	
En el formulario de restablecimiento introducir una contraseña distinta en el campo el confirmación de contraseña	El sistema notifica que las contraseñas no coinciden.	
En el formulario de restablecimiento introducir correctamente los datos	La pasword es restablecida correctamente.	

El envío de email de recuperación no había sido adaptado de la misma manera que el email de activación por lo que no enviaba los correos correctamente, tras la adaptación si lo hace.

Caso de Uso 3: Login		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Introducir las credenciales correctas como usuario standard	El sistema redirige al usuario al listado de medios.	
Introducir las credenciales correctas como usuario administrador	El sistema redirige al administrador al listado de usuarios.	
Introducir unas credenciales incorrectas	Se notifica que las credenciales no son válidas.	
Introducir unas credenciales correctas con un usuario bloqueado o inactivo	Se notifica el estado de bloqueo o falta de activación y se deniega el acceso.	

Caso de Uso 4: Cambiar idioma		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Solicitar un cambio de idioma	El idioma de la interfaz cambia y se mantiene correctamente en las siguientes peticiones.	

Caso de Uso 5: Descargar aplicación móvil		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Acceder al enlace de descarga de la aplicación	Funciona el enlace y se puede descargar la aplicación correctamente.	

Caso de Uso 6: Cambiar tema		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Solicitar un cambio de tema	El tema de la interfaz cambia y se mantiene correctamente en las siguientes peticiones.	

Caso de Uso 7: Gestionar medios		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Crear un nuevo medio introduciendo nombre y ubicación, desde la página de listado.	El medio se crea y añade correctamente a la lista.	
Crear un nuevo medio utilizando la página de creación	El medio se crea y a continuación se muestra.	

Crear un nuevo medio utilizando la herramienta de sugerencia	El formulario se autocompleta correctamente y el medio se añade de manera correcta.	
Editar un medio	Los cambios se guardan correctamente.	
Borrar uno o más medios	Los medios se borran correctamente.	
Cancelar la edición o borrado	La operación se cancela correctamente.	
Buscar medios por tipo	Se muestra el listado de medios sólo del tipo especificado.	
Filtrar medios	Se muestra el listado de medios pero sólo aquellos que contienen la cadena de texto especificada en el filtro en alguno de sus campos.	
Cambiar número de registros a mostrar	Se cambia correctamente el número máximo de medios que se muestran en cada página.	

Caso de Uso 8: Gestionar listas de reproducción		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Crear una nueva lista de reproducción desde la página de listado	Se crea y se añade correctamente a la tabla de listas.	
Crear una nueva lista utilizando la página de creación	La lista se crea y a continuación se muestra.	
Editar lista	Los cambios se guardan correctamente.	
Borrar una o más listas	Se borran correctamente.	
Cancelar la edición o borrado	La operación se cancela correctamente.	
Filtrar listas	Se muestra el listado de elementos que contienen la cadena de texto especificada en el filtro en alguno de sus campos.	
Cambiar número de registros a mostrar	Se cambia correctamente el número máximo de listas que se muestran en cada página.	

Caso de Uso 9: Gestionar elementos de una lista de reproducción		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Añadir un medio a la lista de reproducción	El medio se añade correctamente.	
Editar un medio de la lista	Las modificaciones se guardan correctamente.	

Eliminar un medio de la lista	El medio se elimina de la lista.	
Cancelar la edición o borrado	La operación se cancela correctamente.	

El medio se añadía a la lista de reproducción pero el reproductor no la reproducía hasta que no se recargase la página por lo que se fuerza una recarga para que la lista del reproductor multimedia esté en sincronía con la tabla de datos que muestra los medios de la lista.

Caso de Uso 10: Reproducir		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Reproducir un medio	El medio se reproduce correctamente.	
Reproducir una lista	Se reproduce correctamente.	
Reproducir un medio con una fuente errónea o no disponible	El medio no se reproduce y se notifica el error.	
Reproducir una lista vacía	No se muestra el reproductor y solo permite añadir medios a la lista.	
Avanzar o retroceder de medio en una lista	Se cambia de medio correctamente.	
Utilizar varios reproductores simultáneamente con la misma cuenta	La reproducción se sincroniza compartiendo las órdenes que reciben y envían.	
Parar reproducción	La reproducción se para.	
Pausar reproducción	La reproducción se pausa.	

En esta prueba se detecta que el reproductor dependiente del plugin VLC tiene problemas ocasionales de fiabilidad y bloquea se ejecución en la página que obliga a recargarla cuando esto sucede para seguir funcionando con normalidad.

Caso de Uso 11: Cambiar contraseña		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Rellenar el formulario correctamente	Se cambia la contraseña y se notifica el éxito de la operación.	
Introducir incorrectamente la contraseña actual	Se notifica el error y se pide introducirla correctamente.	
Introducir nuevas contraseñas que no coincidan	Se notifica el error y se pide que coincidan.	

Caso de Uso 12: Cerrar sesión		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Cerrar sesión	La sesión finaliza correctamente, se redirige a la página de login como usuario anónimo y el menú cambia con las opciones disponibles en ese momento.	

Caso de Uso 13: Consultar ayuda		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Consultar ayuda	La ayuda se muestra correctamente.	

Caso de Uso 14: Gestionar usuarios		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Crear un nuevo usuario correctamente	El usuario es creado y guardado en la base de datos.	
Introducir errores u omisiones en la creación de un nuevo usuario	El sistema notifica los errores y requiere su corrección para poder guardar el usuario.	
Editar los datos de un usuario	Los cambios se guardan correctamente.	
Eliminar un usuario	El usuario se elimina correctamente.	
Bloquear usuario	El usuario queda bloqueado y se impide su inicio de sesión sistema.	

El usuario no se activaba por defecto por lo que no podía acceder a la aplicación, añadiendo esta activación inicial se corrigió el problema.

En fases tempranas de pruebas el usuario se creaba sin asignarle un rol inicial, por lo que no podía interactuar debidamente con el sistema por falta del mismo. Se solucionó el problema creando los usuarios con rol USER por defecto.

Caso de Uso 15: Gestionar roles		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Asignar un rol a un usuario	El usuario adquiere el nuevo rol.	
Eliminar un rol de un usuario	El usuario pierde el rol.	

Caso de Uso 16: Gestionar asignaciones medio-lista		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Editar una asignación	Los cambios se guardan correctamente.	
Eliminar una asignación	La asignación se elimina correctamente.	

Caso de Uso 17: Configurar preferencias de control remoto		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Introducir datos correctamente	La configuración se guarda correctamente y permite la transmisión de órdenes.	
Rellenar datos de forma incorrecta	Las órdenes no se transmiten.	

Caso de Uso 18: Controlar reproducción remota		
Prueba	**Resultado Esperado**	**Resultado Obtenido**
Controlar un reproductor remotamente	Las órdenes se transmiten correctamente y el estado del reproductor cambia según ellas, cambio de medio, estado de reproducción cambio de volumen, cambio a pantalla completa.	
Controlar varios reproductores remotamente	Las órdenes de control se transmiten a todos los reproductores asociados a la cuenta de manera sincronizada.	
Intentar reproducir sin conexión	Se notifica el error de falta de conexión.	

8.3 Pruebas de Usabilidad y Accesibilidad

A continuación se detallan los resultados de las pruebas de usabilidad y accesibilidad antes descritas realizadas durante el desarrollo del proyecto

8.3.1 Pruebas de Usabilidad

8.3.1.1 Cuestionarios

A partir de los cuestionarios que se diseñaron anteriormente y de los procedimientos explicados, mostramos aquí el resultado de aplicarlos sobre los usuarios de pruebas.

En total se han utilizado 5 usuarios para realizar las pruebas de usabilidad.

En primer lugar se muestran las respuestas dadas en el cuestionario de datos generales.

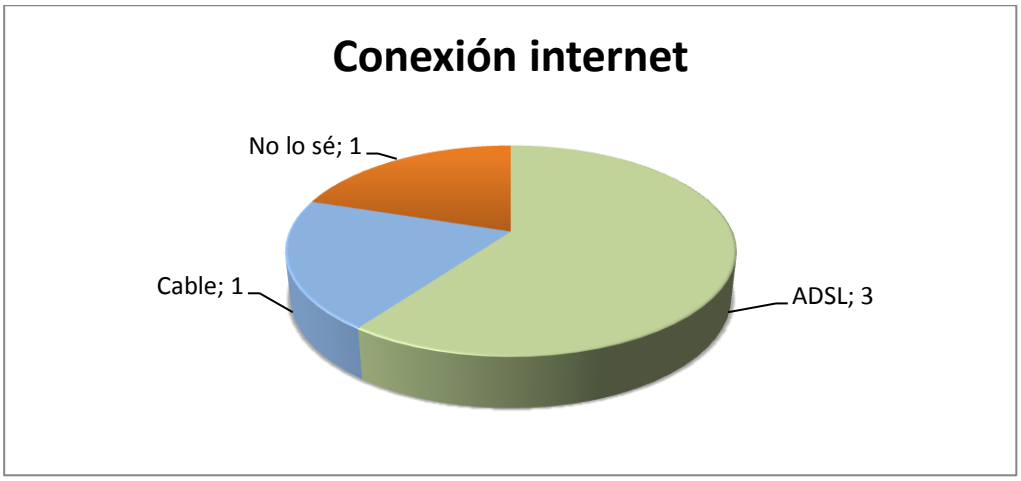


Figura 8.1 Resultado cuestionario sobre tipo de conexión a internet

Figura 8.2 Resultado cuestionario sobre actividades para las que usa el ordenador

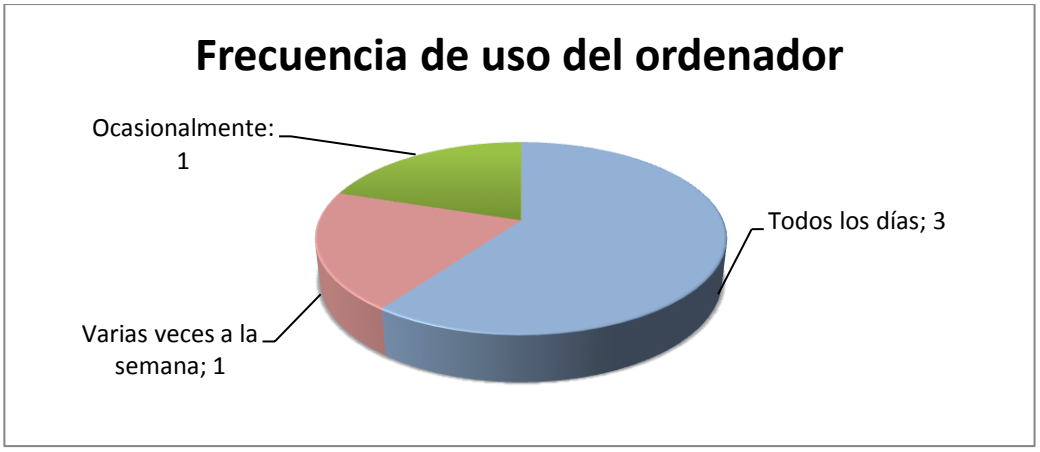


Figura 8.3 Resultado cuestionario sobre frecuencia de uso del ordenador

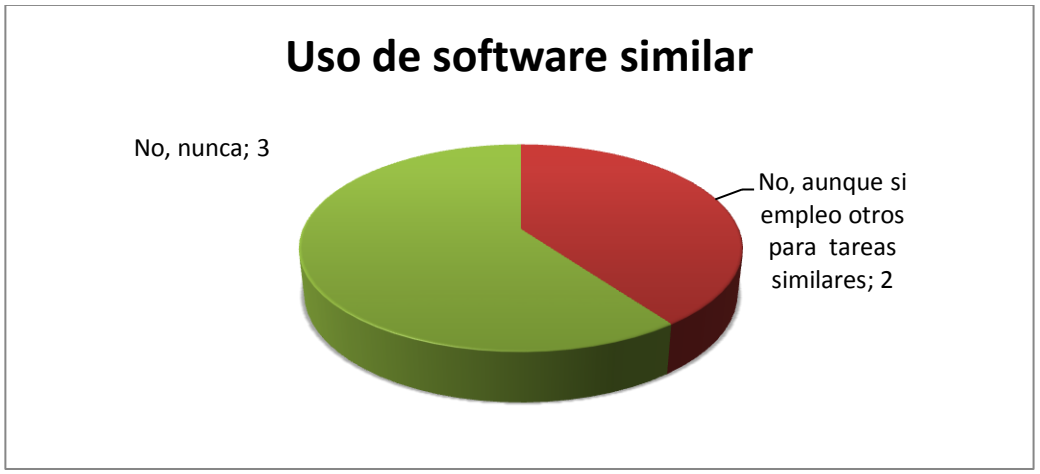


Figura 8.4 Resultado cuestionario sobre utilización de software similar

Figura 8.5 Resultado cuestionario sobre característica más valorada en un programa

A continuación se muestran los resultados de las valoraciones ofrecidas por los usuarios tras realizar las tareas pedidas:

General	Muy cierto	Cierto	Poco cierto	Nada cierto
Creo que me gustará visitar con frecuencia el sitio	1	3	1	
Encontré el sitio demasiado complejo		1	2	2
Pienso que el sitio es fácil de usar	1	3	1	
Creo que necesitaría ayuda de un experto para utilizar el sitio		1	1	3
Hay demasiada inconsistencia en la web			1	4
Creo que la mayoría de la gente podría usar el sitio web rápidamente	2	2	1	
He encontrado el sitio web incómodo de usar			1	4
Necesitaría aprender muchas cosas antes de manejar el sitio web	1		1	3

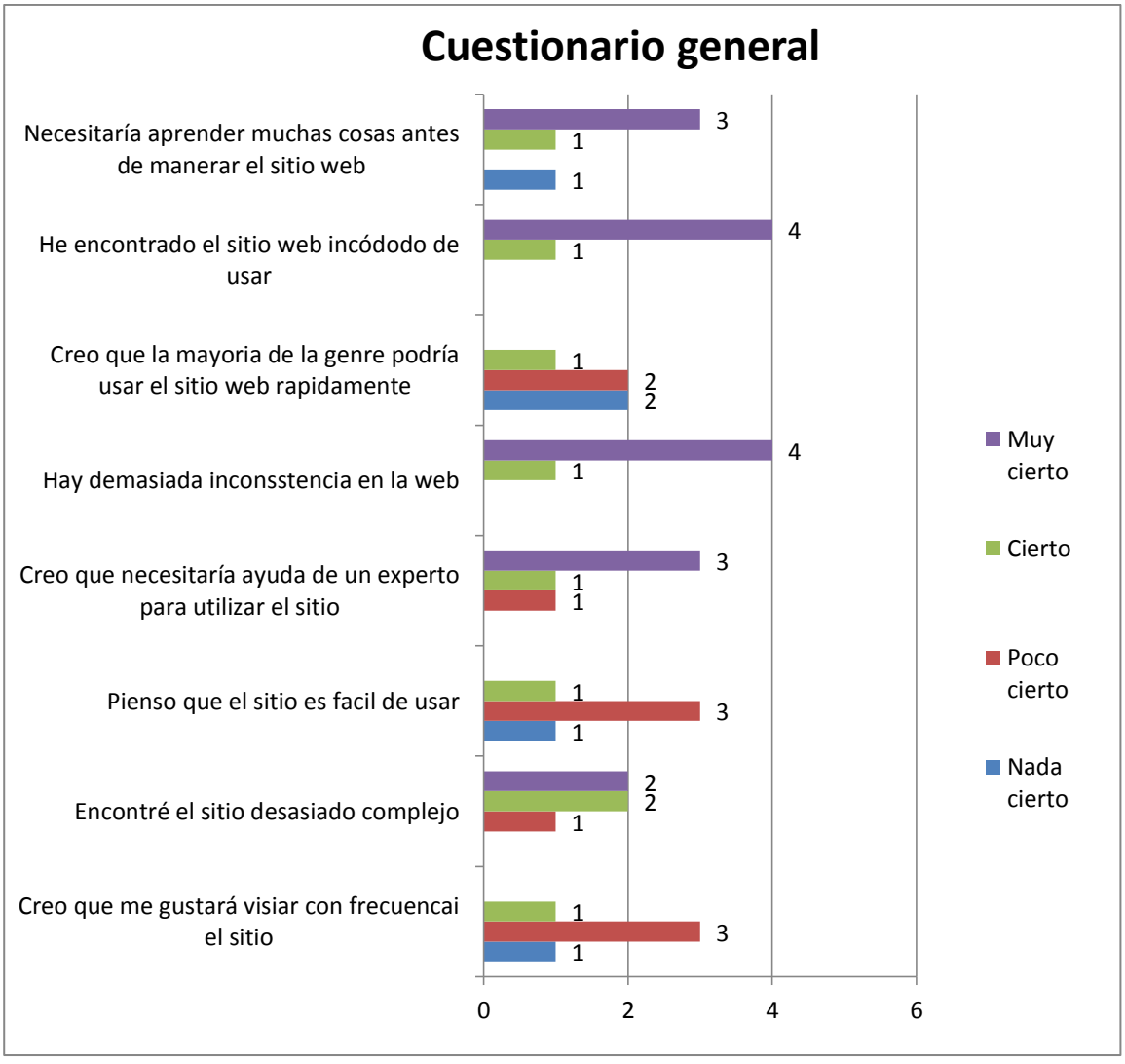


Figura 8.6 Resultado cuestionario sobre usabilidad general

Facilidad de uso	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Sabe dónde está dentro de la aplicación?	4	1		
¿Existe ayuda para las funciones en caso de que tenga dudas?		2	3	
¿Le resulta sencillo el uso de la aplicación?		3	2	

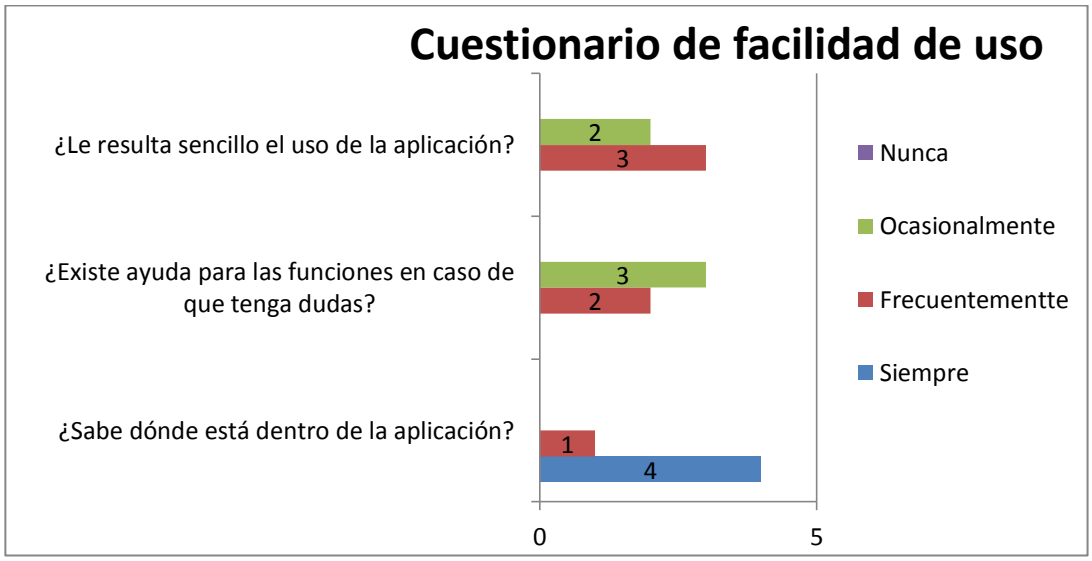


Figura 8.7 Resultado cuestionario sobre facilidad de uso

Funcionalidad	Siempre	Frecuentemente	Ocasionalmente	Nunca
¿Funcionan las tareas como espera?		5		
¿El tiempo de respuesta es muy prolongado?		1	3	1

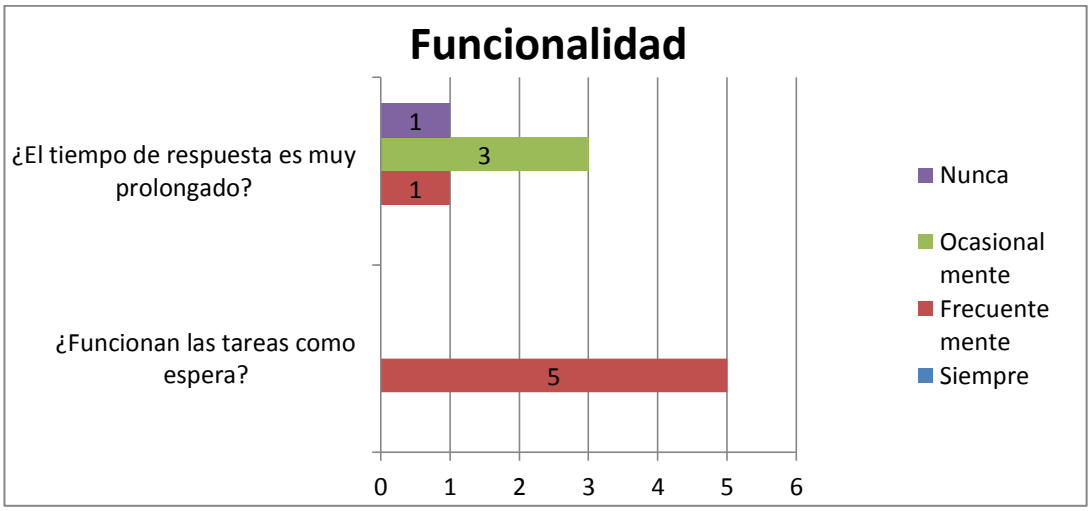


Figura 8.8 Resultado cuestionario sobre funcionalidad

Aspectos gráficos	Muy adecuado	Adecuado	Poco Adecuado	Nada Adecuado
El tipo y tamaño de letra	4	1		
Los iconos e imágenes usados	5			
Los colores empleados	5			

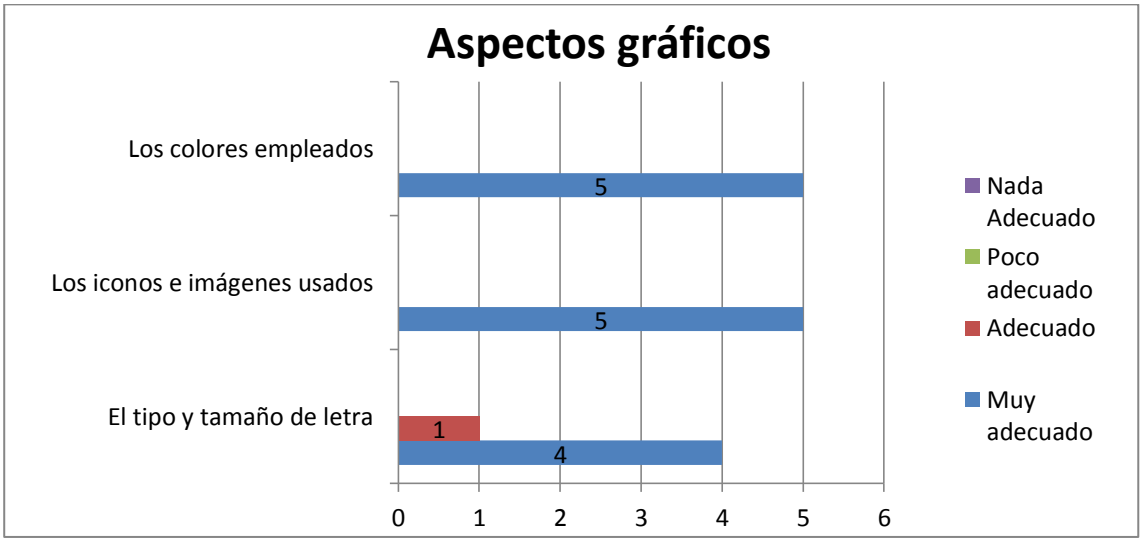


Figura 8.5 Resultado cuestionario sobre aspectos gráficos

Diseño de la Interfaz	Si	No	A veces
¿Le resulta fácil de usar?	4		1
¿El diseño de las pantallas es claro y atractivo?	3		2
¿Cree que el programa está bien estructurado?	4		1

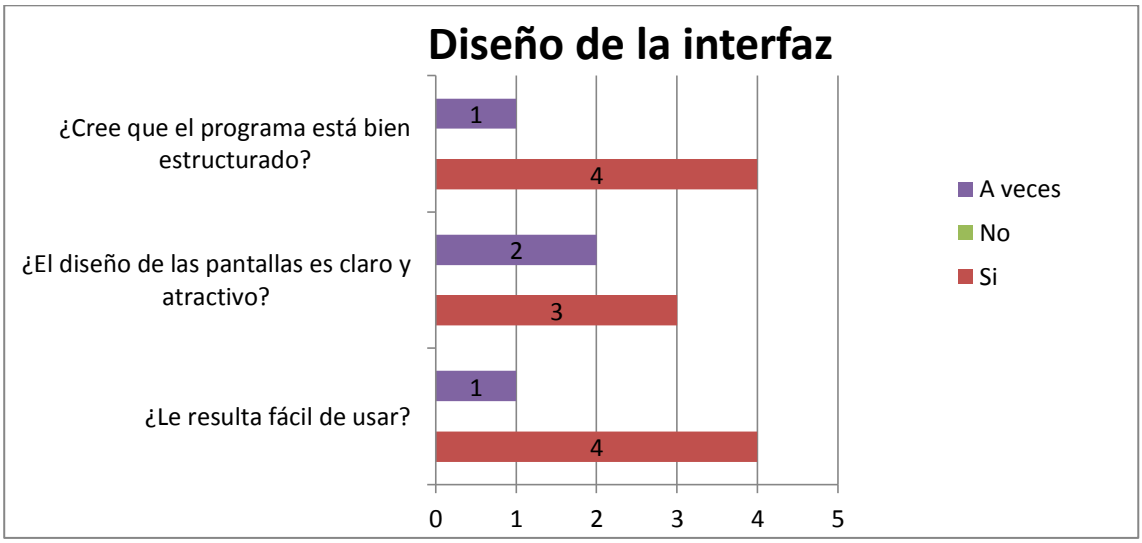


Figura 8.9 Resultado cuestionario sobre diseño de la interfaz

8.3.1.2 Guía de Evaluación Heurística de Sitios Web

En este apartado realizamos una autoevaluación de la usabilidad de la aplicación web utilizando como referencia la Guía de Evaluación Heurística de Sitios Web creada por Yusef Hassan Montero (http://www.nosolousabilidad.com/articulos/heuristica.htm):

Criterios	¿Cumplido?
Generales	
¿Cuáles son los objetivos del sitio web? ¿Son concretos y bien definidos? ¿Los contenidos y servicios que ofrece se corresponden con esos objetivos?	✓
¿Tiene una URL correcta, clara y fácil de recordar? ¿Y las URL de sus páginas internas? ¿Son claras y permanentes?	✓
¿Muestra de forma precisa y completa qué contenidos o servicios ofrece realmente el sitio web? El diseño de la página de inicio debe ser diferente al resto de páginas y cumplir la función de 'escaparate' del sitio.	✓
¿La estructura general del sitio web está orientada al usuario? Los sitios web deben estructurarse pensando en el usuario, sus objetivos y necesidades. La estructura interna de la empresa u organización, cómo funciona o se organiza no interesan al usuario.	✓
¿El *look & feel* general se corresponde con los objetivos, características, contenidos y servicios del sitio web? Ciertas combinaciones de colores ofrecen imágenes más o menos formales, serias o profesionales.	✓
¿Es coherente el diseño general del sitio web? Se debe mantener una coherencia y uniformidad en las estructuras y colores de todas las páginas. Esto sirve para que el usuario no se desoriente en su navegación.	✓
¿Es reconocible el diseño general del sitio web? Cuánto más se parezca el sitio web al resto de sitios web, más fácil será de usar.	✓
¿El sitio web se actualiza periódicamente? ¿Indica cuándo se actualiza? Las fechas que se muestren en la página deben corresponderse con actualizaciones, noticias, eventos...no con la fecha del sistema del usuario.	✓
Identidad e Información	
¿Se muestra claramente la identidad de la empresa-sitio a través de todas las páginas?	✓
El Logotipo, **¿es significativo, identificable y suficientemente visible?**	✓
El eslogan o *tagline*, **¿expresa realmente qué es la empresa y qué servicios ofrece?**	**N/A**
¿Se ofrece algún enlace con información sobre la empresa, sitio web, 'webmaster',...?	✓
¿Se proporciona mecanismos para ponerse en contacto con la empresa? (email, teléfono, dirección postal, fax...)	✓
¿Se proporciona información sobre la protección de datos de carácter personal de los clientes o los derechos de autor de los contenidos del sitio web?	✗

En artículos, noticias, informes... **¿Se muestra claramente información sobre el autor, fuentes y fechas de creación y revisión del documento?**	**N/A**
Lenguaje y Redacción	
¿El sitio web habla el mismo lenguaje que sus usuarios? Se debe evitar usar un lenguaje corporativista. Así mismo, hay que prestarle especial atención al idioma, y ofrecer versiones del sitio en diferentes idiomas cuando sea necesario.	
¿Emplea un lenguaje claro y conciso?	
¿Es amigable, familiar y cercano? Es decir, lo contrario a utilizar un lenguaje constantemente imperativo, mensajes crípticos, o tratar con "desprecio" al usuario.	
¿1 párrafo = 1 idea? Cada párrafo es un objeto informativo. Trasmita ideas, mensajes...Se deben evitar párrafos vacíos o varios mensajes en un mismo párrafo.	
Rotulado	
Los rótulos, **¿son significativos?** Ejemplo: evitar rótulos del tipo "haga clic aquí".	
¿Usa rótulos estándar? Siempre que exista un "estándar" comúnmente aceptado para el caso concreto, como "Mapa del Sitio" o "Acerca de...".	
¿Usa un único sistema de organización, bien definido y claro? No se deben mezclar diferentes. Los sistemas de organización son: alfabético, geográfico, cronológico, temático, orientado a tareas, orientado al público y orientado a metáforas.	
¿Utiliza un sistema de rotulado controlado y preciso? Por ejemplo, si un enlace tiene el rótulo "Quiénes somos", no puede dirigir a una página cuyo encabezamiento sea "Acerca de"	
El título de las páginas, **¿Es correcto? ¿Ha sido planificado?** Relacionado con la capacidad para poder buscar y encontrar el sitio *web*.	
Estructura y Navegación	
La estructura de organización y navegación, **¿Es la más adecuada?** Hay varios tipos de estructuras: jerárquicas, hipertextual, facetada,...	
En el caso de estructura jerárquica, **¿Mantiene un equilibrio entre Profundidad y Anchura?**	
En el caso de ser puramente hipertextual, **¿Están todos los clúster de nodos comunicados?** Aquí se mide la distancia entre nodos.	
¿Los enlaces son fácilmente reconocibles como tales? ¿Su caracterización indica su estado (visitados, activos,...)? Los enlaces no sólo deben reconocerse como tales, sino que su caracterización debe indicar su estado, y ser reconocidos como una unidad	
En menús de navegación, **¿Se ha controlado el número de elementos y de términos por elemento para no producir sobrecarga memorística?** No se deben superar los 7±2 elementos, ni los 2 o, como mucho, 3 términos por elemento.	

¿Es predecible la respuesta del sistema antes de hacer clic sobre el enlace? Relacionado con el nivel de significación del rótulo del enlace, aunque también con: el uso de globos de texto, información contextual, la barra de estado del navegador,...	
¿Se ha controlado que no haya enlaces que no llevan a ningún sitio? Enlaces que no llevan a ningún sitio: Los enlaces rotos, y los que enlazan con la misma página que se está visualizando (por ejemplo enlaces a la "home" desde la misma página de inicio)	
¿Existen elementos de navegación que orienten al usuario acerca de dónde está y cómo deshacer su navegación? ...como *breadcrumbs*, enlaces a la página de inicio,...recuerde que el logo también es recomendable que enlace con la página de inicio.	
Las imágenes enlace, **¿se reconocen como clicables? ¿Incluyen un atributo 'title' describiendo la página de destino?** En este sentido, también hay que cuidar que no haya imágenes que parezcan enlaces y en realidad no lo sean.	
¿Se ha evitado la redundancia de enlaces?	
¿Se ha controlado que no haya páginas "huérfanas"? Páginas huérfanas: que aún siendo enlazadas desde otras páginas, éstas no enlacen con ninguna.	
Layout de la Página	
¿Se aprovechan las zonas de alta jerarquía informativa de la página para contenidos de mayor relevancia? (como por ejemplo la zona central)	
¿Se ha evitado la sobrecarga informativa? Esto se consigue haciendo un uso correcto de colores, efectos tipográficos y agrupaciones para discriminar información. Los grupos diferentes de objetos informativos de una página deben ser 7±2.	
¿Es una interfaz limpia, sin ruido visual?	
¿Existen zonas en "blanco" entre los objetos informativos de la página para poder descansar la vista?	
¿Se hace un uso correcto del espacio visual de la página? Es decir, que no se desaproveche demasiado espacio con elementos de decoración, o grandes zonas en "blanco", y que no se adjudique demasiado espacio a elementos de menor importancia.	 **mejorable**
¿Se utiliza correctamente la jerarquía visual para expresar las relaciones del tipo "parte de" entre los elementos de la página? (La jerarquía visual se utiliza para orientar al usuario)	
¿Se ha controlado la longitud de página? Se debe evitar en la medida de lo posible el *scrolling*. Si la página es muy extensa, se debe fraccionar.	
Búsqueda (búsqueda de medios por tipo)	
¿Se encuentra fácilmente accesible? Es decir: directamente desde la home, y a ser posible desde todas las páginas del sitio, y colocado en la zona superior de la página.	
¿Es fácilmente reconocible como tal?	
¿Permite la búsqueda avanzada? (siempre y cuando, por las características del sitio web, fuera de utilidad que la ofreciera)	

¿Muestra los resultados de la búsqueda de forma comprensible para el usuario?	
¿La caja de texto es lo suficientemente ancha?	
¿Asiste al usuario en caso de no poder ofrecer resultados para una consultada dada?	
Elementos Multimedia	
¿Las fotografías están bien recortadas? ¿Son comprensibles? ¿Se ha cuidado su resolución?	
¿Las metáforas visuales son reconocibles y comprensibles por cualquier usuario? (prestar especial atención a usuarios de otros países y culturas)	
¿El uso de imágenes o animaciones proporciona algún tipo de valor añadido?	
¿Se ha evitado el uso de animaciones cíclicas?	
Ayuda	
Si posee una sección de Ayuda, **¿Es verdaderamente necesaria?** Siempre que se pueda prescindir de ella simplificando los elementos de navegación e interacción, debe omitirse esta sección.	
En enlace a la sección de Ayuda, **¿Está colocado en una zona visible y "estándar"?** La zona de la página más normal para incluir el enlace a la sección de Ayuda, es la superior derecha.	
¿Se ofrece ayuda contextual en tareas complejas? (transferencias bancarias, formularios de registro...)	
Si posee *FAQs*, **¿Es correcta tanto la elección como la redacción de las preguntas? ¿Y las respuestas?**	**N/A**
Accesibilidad (debería cubrirse con los test de Accesibilidad posteriores)	
¿El tamaño de fuente se ha definido de forma relativa, o por lo menos, la fuente es lo suficientemente grande como para no dificultar la legibilidad del texto?	
¿El tipo de fuente, efectos tipográficos, ancho de línea y alineación empleadas facilitan la lectura?	
¿Existe un alto contraste entre el color de fuente y el fondo?	
¿Incluyen las imágenes atributos 'alt' que describan su contenido?	
¿Es compatible el sitio web con los diferentes navegadores? ¿Se visualiza correctamente con diferentes resoluciones de pantalla? Se debe prestar atención a: *JScript*, *CSS*, tablas, fuentes...	
¿Puede el usuario disfrutar de todos los contenidos del sitio web sin necesidad de tener que descargar e instalar *plugins* adicionales?	**Debe disponer del plugin de reproducción**
¿Se ha controlado el peso de la página? Se deben optimizar las imágenes, controlar el tamaño del código *JScript*...	

¿Se puede imprimir la página sin problemas? Leer en pantalla es molesto, por lo que muchos usuarios preferirán imprimir las páginas para leerlas. Se debe asegurar que se puede imprimir la página (no salen partes cortadas), y que el resultado es legible.	
Control y Retroalimentación	
¿Tiene el usuario todo el control sobre el interfaz? Se debe evitar el uso de ventanas pop-up, ventanas que se abren a pantalla completa, banners intrusivos...	
¿Se informa constantemente al usuario acerca de lo que está pasando? Si el usuario tiene que esperar hasta que se termine una operación, se debe mostrar un mensaje indicándoselo y que debe esperar, con el tiempo de espera estimado o una barra de progreso.	
¿Se informa al usuario de lo que ha pasado? Por ejemplo, cuando un usuario valora un artículo o responde a una encuesta, se le debe informar de que su voto ha sido procesado correctamente.	
Cuando se produce un error, **¿se informa de forma clara y no alarmista al usuario de lo ocurrido y de cómo solucionar el problema?** Siempre es mejor intentar evitar que se produzcan errores a tener que informar al usuario del error.	**No siempre**
¿Posee el usuario libertad para actuar? NO restringir la libertad del usuario: Uso de animaciones que no pueden ser "saltadas", páginas en las que desaparecen los botones de navegación, no impida al usuario poder usar el botón derecho de su ratón...	
¿Se ha controlado el tiempo de respuesta? Esto tiene que ver con el peso de cada página (accesibilidad) y tiene relación con el tiempo que tarda el servidor en finalizar una tarea y responder. El tiempo máximo que esperará un usuario son 10 segundos	
Aclaraciones	
¿Se ha evaluado adecuadamente la orientación del usuario? (Donde estoy, como volver, que he visitado, que va a pasar)	
¿Se ha usado correctamente la publicidad?	

Cabe señalar respeto a la información clara y no alarmista de los errores, que al trabajar con plugins de reproducción, no siempre se dispone de información concreta sobre la causa de un error en el mismo por lo que no se puede transmitir esa información al usuario, aunque sí que se informa de la aparición de error en sí.

No se consideró necesaria la inclusión de información acerca de la protección de datos personales de los clientes, ya que no se almacenan datos de este tipo más allá del propio nombre del usuario.

La aplicación ha sido probada con los navegadores Chrome, IE, Opera y Firefox

No se ha incluido publicidad salvo una pequeña referencia en el pie al framework de desarrollo.

8.3.2 Pruebas de Accesibilidad

A continuación realizaremos una evaluación de accesibilidad del sitio para considerar si realmente se han cumplido los objetivos marcados en este aspecto.

Las herramientas utilizadas serán las siguientes:

- **W3C CSS Validator.** Valida el código CSS usado en los documentos. http://jigsaw.w3.org/css-validator/

- **TAW.** Herramienta para el análisis de la accesibilidad de sitios web, alcanzando de una forma integral y global a todos los elementos y páginas que lo componen (WCAG 2.0). http://www.tawdis.net

- **HERA.** Utilidad para revisar la accesibilidad de las páginas web de acuerdo con las recomendaciones de las Directrices de Accesibilidad para el Contenido Web 1.0 (WCAG 1.0) http://www.sidar.org/hera/

- Navegadores **Chrome, IE, Opera y Firefox**

- Navegador de texto **Lynx**

8.3.2.1 Selección de páginas representativas de la aplicación

Ya que examinar todas las páginas de la aplicación resultaría un trabajo demasiado laborioso y que no aportaría demasiado, se ha hecho una selección de las páginas más representativas del sitio a las que se le realizará la evaluación. Las páginas en cuestión son las siguientes:

- **index.jspx** : página principal de la aplicación.
- **login.jspx** : página de inicio de sesión de la aplicación.
- **medias/show.jspx** : página de visualización de medios de la aplicación con reproductor incrustado
- **medias/list.jspx** : página de listado de medios con creación inline, contiene tablas.
- **forgot_password/form.jspx** : página de formulario de recuperación de contraseña.

8.3.2.2 Uso de herramientas de evaluación automática

A continuación se hace uso de las herramientas anteriormente citadas mostrando alguno de sus resultados.

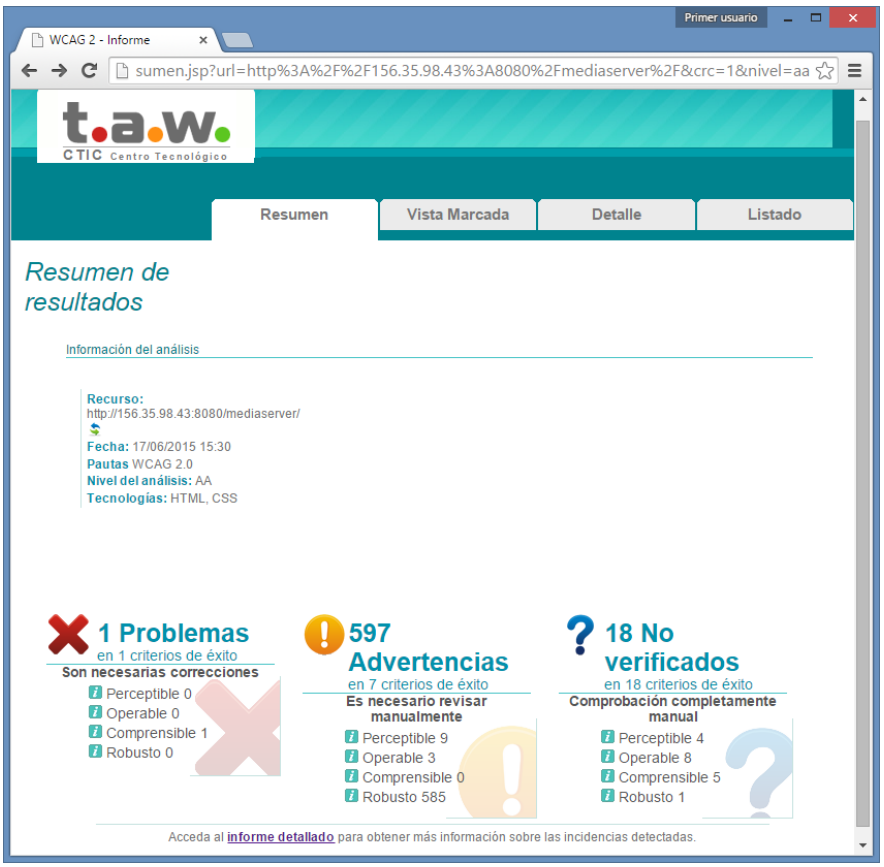


Figura 8.10 Resultado de la evaluación automática de WCAG 2.0 con la herramienta TAW

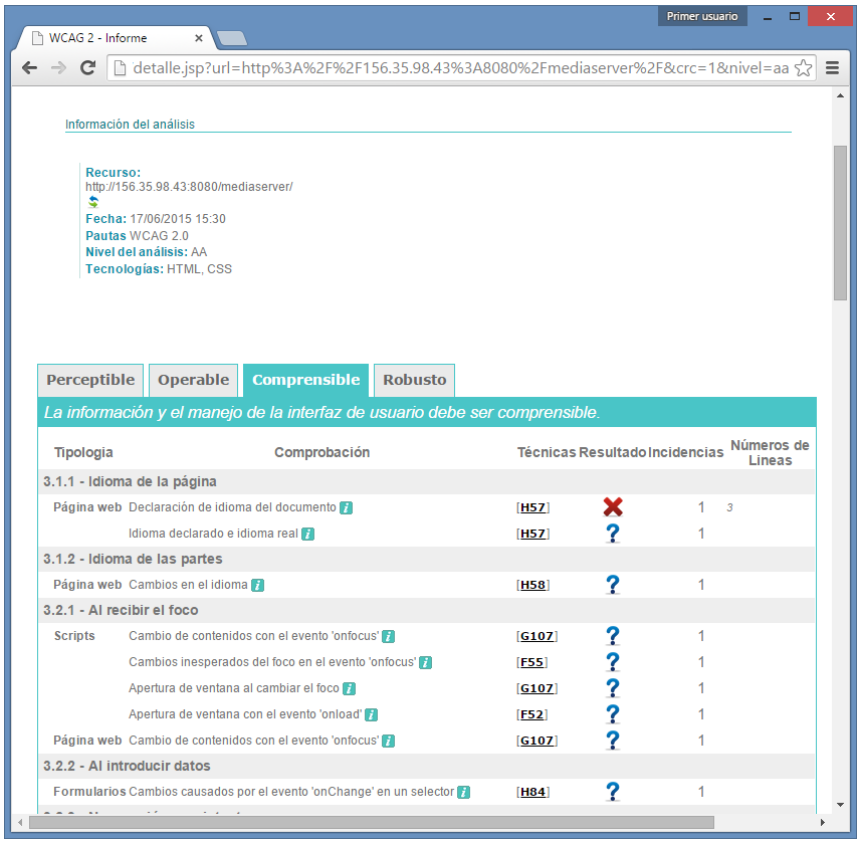


Figura 8.11 Detalle del error detectado con la herramienta TAW de evaluación automática

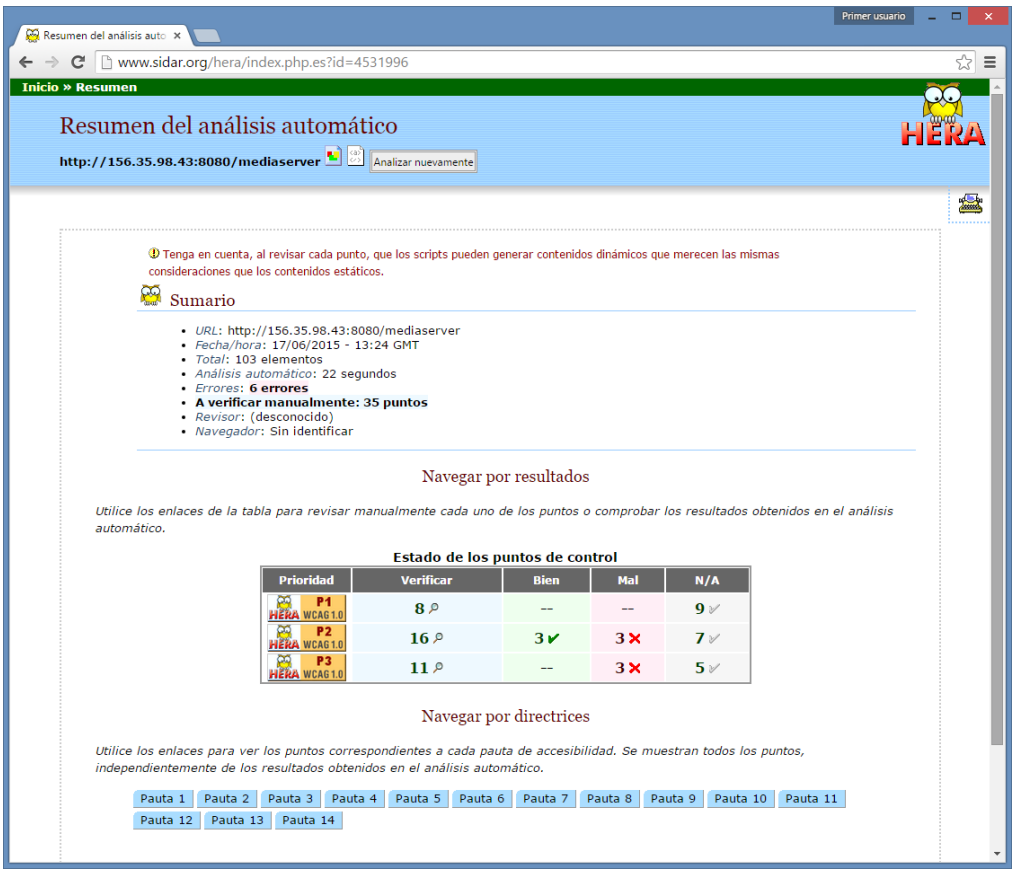


Figura 8.12 Resultado de la evaluación automática con la herramienta HERA

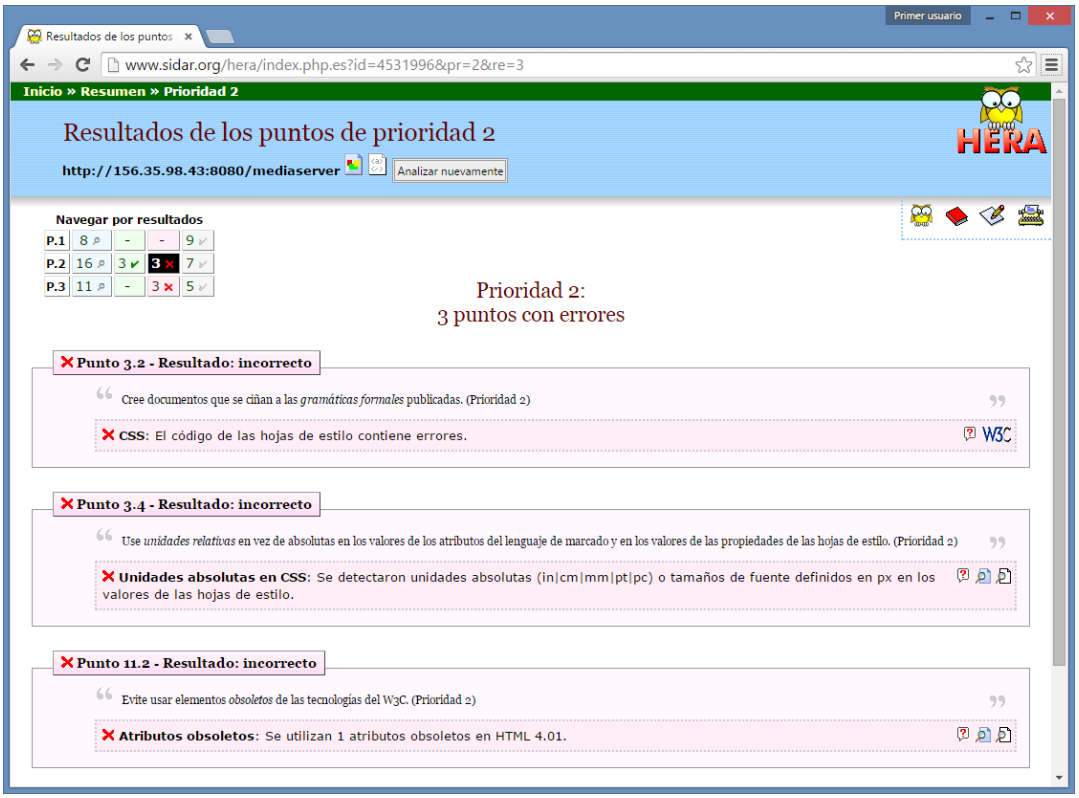


Figura 8.13 Detalle de errores de prioridad 2 detectados con la herramienta HERA

La validación de CSS señala una cantidad muy elevada de errores debidos al uso de las bibliotecas del framework bootstrap, con multitud de reglas y propiedades específicas para distintos navegadores, que no son reconocidas por la herramienta de evaluación automática http://jigsaw.w3.org/css-validator/ .

Estos errores en el CSS también se ven en la evaluación de la herramienta HERA en los problemas detectados con prioridad 2.

Respecto a la prioridad 3, la herramienta HERA detecta que hay enlaces adyacentes sin caracteres imprimibles que los separen, aunque estos enlaces se corresponden con la selección de idioma situados en una lista del menú de configuración, que visualmente no tienen posibilidad de confusión.

También se señala la ausencia de atajos de teclado, y por último, que no se indica el idioma principal de la página dentro de la cabecera.

Este último punto es el que señala también la herramienta TAW en su evaluación de pautas WACAG 2.0, (la de HERA se trataba de pautas WACAG 1.0)

En cualquier caso la especificación del idioma no es un tema trivial en la aplicación, pues por una parte las páginas están construidas en base a “tiles” o fragmentos, acoplados por el framework según sea necesario; y por otra parte el lenguaje es establecido primeramente por detección del lenguaje del navegador, y después por una cookie generada según la elección del usuario.

8.3.2.3 Pruebas de evaluación manual de las páginas.

Como vemos en las evaluaciones automáticas, muchos de los apartados de avaluación, aunque no aparecen como incorrectos, sí que recomiendan su revisión manual, para comprobar que efectivamente son correctos.

En esta sección utilizaremos distintos navegadores para probar las visualizaciones de las páginas de la selección en distintos tamaños y circunstancias.

Figura 8.14 Vista de la página principal (index.jspx) en dimensiones reducidas con navegador Chrome 43

Como se ve en la captura anterior, el menú se convierte en desplegable para no ocupar excesivo espacio en la parte superior de la página, y poder adaptarse también así a dispositivos móviles.

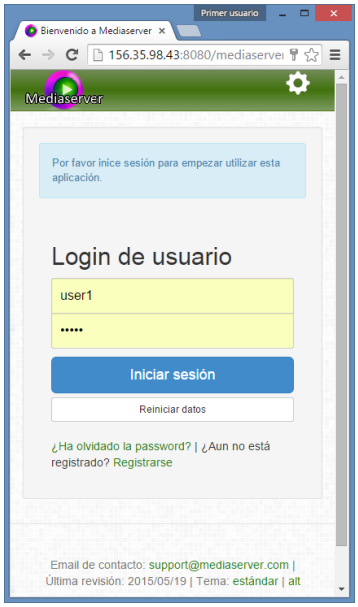


Figura 8.15 Vista de la página de login (login.jspx) en dimensiones reducidas con navegador Chrome 43

Figura 8.16 Vista de la página principal (index.jpsx) con zoom ampliado con navegador Opera 12.17

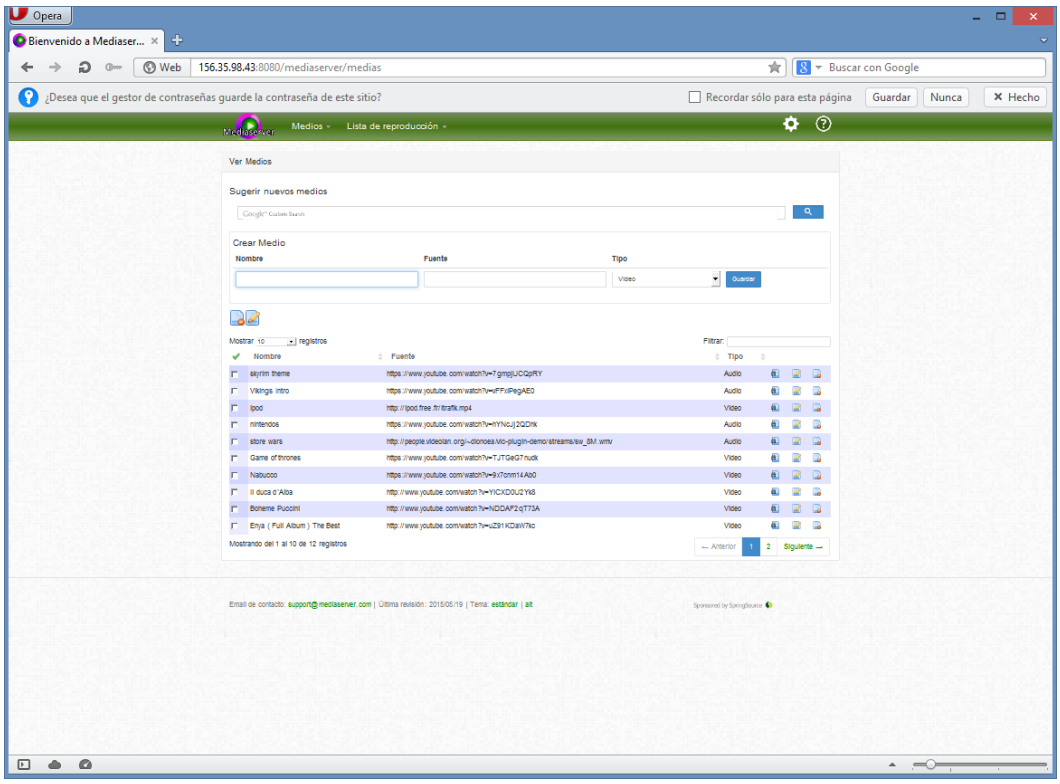


Figura 8.17 Vista de la página de listado de medios (medias/list.jspx) con zoom alejado con navegador Opera 12.17

Tanto ampliando como disminuyendo el zoom del navegador en distintas páginas su organización de contenidos es coherente y las páginas siguen pudiendo utilizarse sin problema.

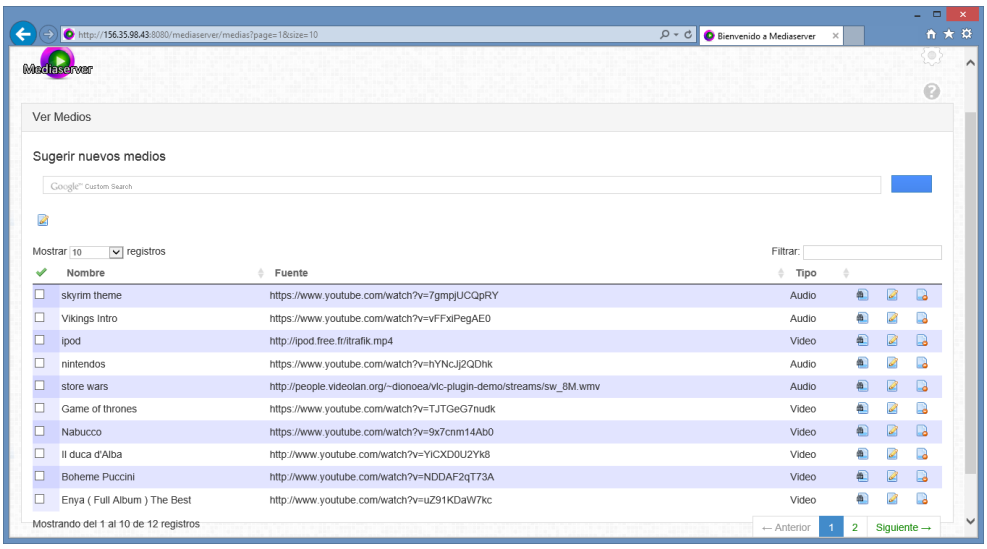


Figura 8.18 Vista de la página de listado de medios (medias/list.jspx) con navegador Internet Explorer 11

En este navegador como se ve en la figura, no se puede distinguir bien la barra de menú principal, aunque la tabla de medios se ve correctamente. En general este navegador ha mostrado este problema con todas las páginas de la aplicación probadas.

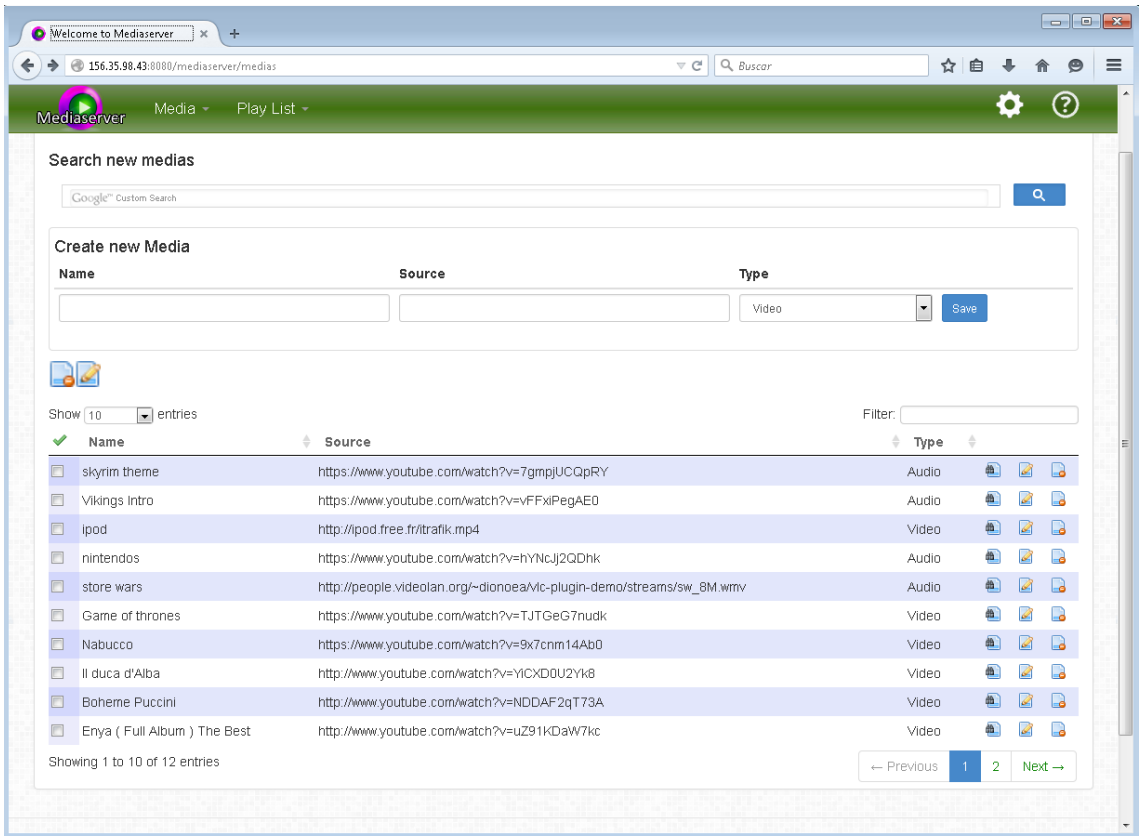


Figura 8.19 Vista de la página de listado de medios (medias/list.jspx) con navegador Firefox 37.0.1

Figura 8.20 Vista de la página de reproducción de un medio (medias/show.jspx) con navegador Firefox 37.0.1

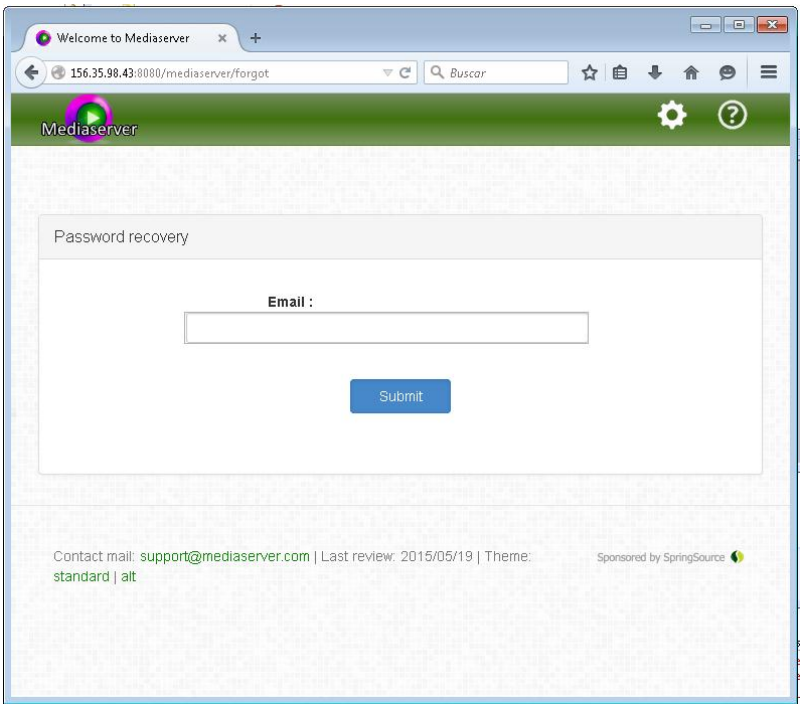


Figura 8.21 Vista de la página de recuperación de contraseña (forgot_password/form.jspx) con navegador Firefox 37.0.1

Como se ve en las capturas anteriores todas las páginas de la aplicación se ven correctamente en el navegador Firefox en distintas condiciones de tamaño.

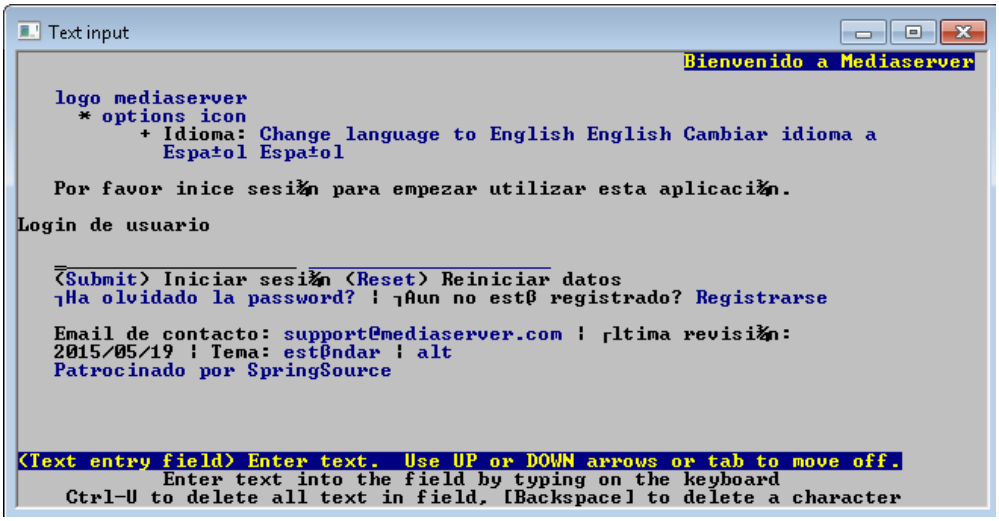


Figura 8.22 Vista de la página de login (login.jspx) con navegador Lynx

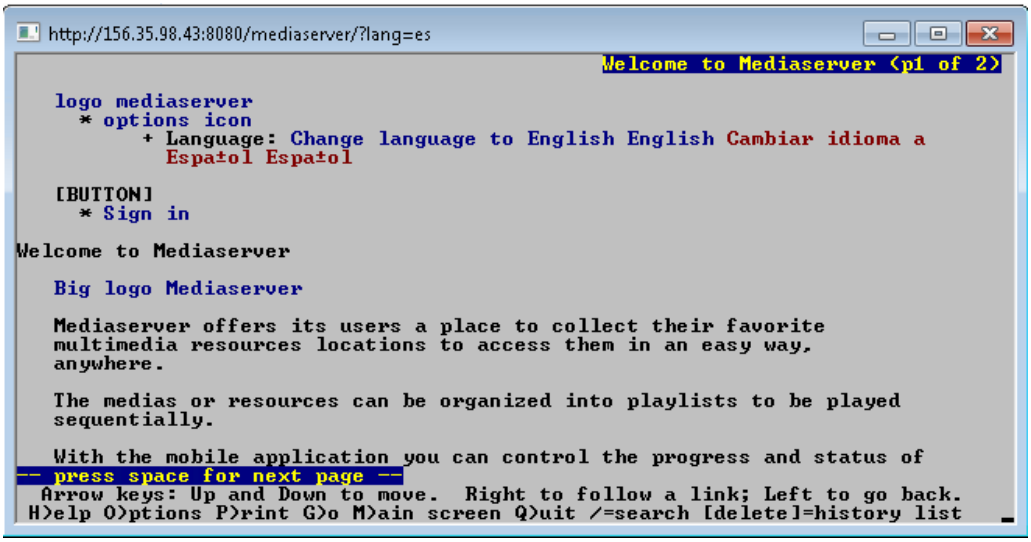


Figura 8.23 Vista de la página principal (index.jspx) con el servidor Lynx

Aunque distinta de la vista gráfica convencional, la página principal del sitio a través de un navegador de texto como Lynx aún se ve estructurada, permite acceder a la información de una manera comprensible, y navegar por el sitio con razonable sencillez

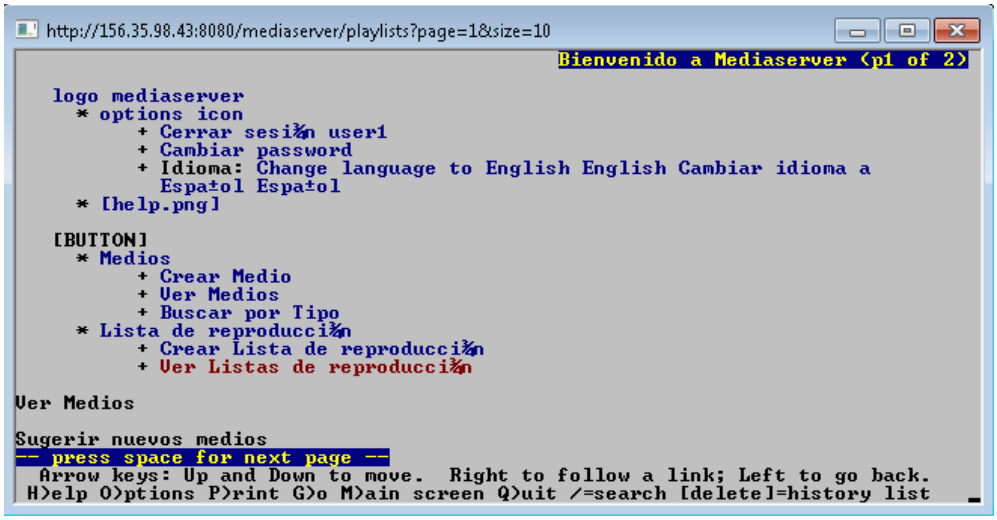


Figura 8.24 Vista de la página de listado de medios con navegador Lynx

El Listado de medios es una página que contiene datos representados mediante tablas, que son cargadas dinámicamente. No se ven correctamente en este navegador textual.

8.3.2.4 Análisis de resultados de las pruebas

- El uso de frameworks recientes como Spring4, gv-Nix y bibliotecas de estilo y comportamiento, como Bootstrap, aunque ofrecen facilidades de desarrollo y mejoras de aspecto visuales, pueden suponer un riesgo para el cuidado de la accesibilidad, ya que incorporan numerosas anotaciones específicas de distintos navegadores, que no son validadas por las herramientas automáticas.
- El control del código generado por un framework es mucho más complejo que el desarrollado manualmente en páginas individuales.
- Además algunos elementos como los menús no se ven correctamente en todos los navegadores como es el caso e Internet Explorer.
- La utilización de plugins y el abundante uso de scripts que modifican el contenido de las páginas también es un elemento que compromete la accesibilidad del sitio, pero tratándose de una aplicación web para la reproducción de contenidos multimedia y su control remoto, la necesidad de estos elementos dinámicos resulta imprescindible.

8.3.2.5 Checklist del WCAG 1.0

La siguiente tabla es el *checklist* que el *WCAG* proporciona para verificar las pautas de accesibilidad de la aplicación *web.*

Puntos de verificación Prioridad 1:

En general (Prioridad 1)	Sí	No	N/A
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.	X		
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.	X		
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).	X		
6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.	X		
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.	X		
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.	X		
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.	X		
Y si utiliza imágenes y mapas de imagen (Prioridad 1)	**Sí**	**No**	**N/A**
1.2 Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.			X
9.1 Proporcione mapas de imagen controlados por el cliente en lugar de por el servidor, excepto donde las zonas sensibles no puedan ser definidas con una forma geométrica.			X
Y si utiliza tablas (Prioridad 1)	**Sí**	**No**	**N/A**
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.	X		
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.			X
Y si utiliza marcos ("frames") (Prioridad 1)	**Sí**	**No**	**N/A**
12.1 Titule cada marco para facilitar su identificación y navegación.	X		
Y si utiliza "applets" y "scripts" (Prioridad 1)	**Sí**	**No**	**N/A**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.	X		
Y si utiliza multimedia (Prioridad 1)	**Sí**	**No**	**N/A**
1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la información importante de la banda visual de una presentación multimedia.		X	
1.4 Para toda presentación multimedia dependiente del tiempo (por ejemplo, una película o animación) sincronice alternativas equivalentes		X	

(por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			
Y si todo lo demás falla (Prioridad 1)	**Sí**	**No**	**N/A**
11.4 Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías *W3C*, sea accesible, tenga información (o funcionalidad) equivalente y sea actualizada tan a menudo como la página (original) inaccesible.			**X**

Puntos de verificación Prioridad 2:

En general (Prioridad 2)	**Sí**	**No**	**N/A**
2.2 Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan el suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o en pantallas en blanco y negro [Prioridad 2 para las imágenes. Prioridad 3 para los textos].	**X**		
3.1 Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.			**X**
3.2 Cree documentos que estén validados por las gramáticas formales publicadas.	**X**		
3.3 Utilice hojas de estilo para controlar la maquetación y la presentación.	**X**		
3.4 Utilice unidades relativas en lugar de absolutas al especificar los valores en los atributos de los marcadores de lenguaje y en los valores de las propiedades de las hojas de estilo.	**X**		
3.5 Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.	**X**		
3.6 Marque correctamente las listas y los ítems de las listas.	**X**		
3.7 Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.	**X**		
6.5 Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.		**X**	
7.2 Hasta que las aplicaciones de usuario permitan controlarlo, evite el parpadeo del contenido (por ejemplo, cambio de presentación en periodos regulares, así como el encendido y apagado).	**X**		
7.4 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.	**X**		
7.5 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener el redireccionamiento automático, no utilice marcadores para redirigir las páginas automáticamente. En su lugar, configure el servidor para que ejecute esta posibilidad.	**X**		
10.1 Hasta que las aplicaciones de usuario permitan desconectar la apertura de nuevas ventanas, no provoque apariciones repentinas de nuevas ventanas y no cambie la ventana actual sin informar al usuario.	**X**		
11.1 Utilice tecnologías *W3C* cuando estén disponibles y sean apropiadas para la tarea y use las últimas versiones que sean soportadas.	**X**		
11.2 Evite características desaconsejadas por las tecnologías *W3C*.	**X**		

12.3 Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.	X		
13.1 Identifique claramente el objetivo de cada vínculo.	X		
13.2 Proporcione metadatos para añadir información semántica a las páginas y sitios.	X		
13.3 Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).			X
13.4 Utilice los mecanismos de navegación de forma coherente.	X		
Y si utiliza tablas (Prioridad 2)	**Sí**	**No**	**N/A**
5.3 No utilice tablas para maquetar, a menos que la tabla tenga sentido cuando se alinee. Por otro lado, si la tabla no tiene sentido, proporcione una alternativa equivalente (la cual debe ser una versión alineada).	X		
5.4 Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.			X
Y si utiliza marcos ("frames") (Prioridad 2)	**Sí**	**No**	**N/A**
12.2 Describa el propósito de los marcos y cómo éstos se relacionan entre sí, si no resulta obvio solamente con el título del marco.	X		
Y si utiliza formularios (Prioridad 2)	**Sí**	**No**	**N/A**
10.2 Hasta que las aplicaciones de usuario soporten explícitamente la asociación entre control de formulario y etiqueta, para todos los controles de formularios con etiquetas asociadas implícitamente, asegúrese de que la etiqueta está colocada adecuadamente.	X		
12.4 Asocie explícitamente las etiquetas con sus controles.	X		
Y si utiliza "applets" y "scripts" (Prioridad 2)	**Sí**	**No**	**N/A**
6.4 Para los *scripts* y *applets*, asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.	X		
7.3 Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.	X		
8.1 Haga los elementos de programación, tales como *scripts* y *applets*, directamente accesibles o compatibles con las ayudas técnicas [Prioridad 1 si la funcionalidad es importante y no se presenta en otro lugar; de otra manera, Prioridad 2].	X		
9.2 Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.	X		
9.3 Para los "scripts", especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.	X		

Puntos de verificación Prioridad 3:

En general (Prioridad 3)	**Sí**	**No**	**N/A**
4.2 Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			X
4.3 Identifique el idioma principal de un documento.		X	
9.4 Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.	X		
9.5 Proporcione atajos de teclado para los vínculos más importantes (incluidos los de los mapas de imagen de cliente), los controles de formulario y los grupos de controles de formulario.		X	
10.5 Hasta que las aplicaciones de usuario (incluidas las ayudas	X		

técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.			
11.3 Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).	**X**		
13.5 Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.			**X**
13.6 Agrupe los vínculos relacionados, identifique el grupo (para las aplicaciones de usuario) y, hasta que las aplicaciones de usuario lo hagan, proporcione una manera de evitar el grupo.	**X**		
13.7 Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.			**X**
13.8 Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.	**X**		
13.9 Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).	**X**		
13.10 Proporcione un medio para saltar sobre un *ASCII art* de varias líneas.			**X**
14.2 Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.	**X**		
14.3 Cree un estilo de presentación que sea coherente para todas las páginas.	**X**		
Y si utiliza imágenes o mapas de imagen (Prioridad 3)	**Sí**	**No**	**N/A**
1.5 Hasta que las aplicaciones de usuario interpreten el texto equivalente para los vínculos de los mapas de imagen de cliente, proporcione vínculos de texto redundantes para cada zona activa del mapa de imagen de cliente.	**X**		
Y si utiliza tablas (Prioridad 3)	**Sí**	**No**	**N/A**
5.5 Proporcione resúmenes de las tablas.	**X**		
5.6 Proporcione abreviaturas para las etiquetas de encabezamiento.	**X**		
10.3 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten correctamente los textos contiguos, proporcione un texto lineal alternativo (en la página actual o en alguna otra) para *todas* las tablas que maquetan texto en paralelo, en columnas de palabras.			**X**
Y si utiliza formularios (Prioridad 3)	**Sí**	**No**	**N/A**
10.4 Hasta que las aplicaciones de usuario manejen correctamente los controles vacíos, incluya caracteres por defecto en los cuadros de edición y áreas de texto.	**X**		

A la hora de realizar las comprobaciones para rellenar esta tabla se utilizaron las herramientas HERA y TAW, que señalaron algunos errores como la ausencia de un idioma principal definido en la cabecera de la página.

Señalar también que respecto a los puntos no cumplidos de prioridad 1 y 2 referentes a contenidos multimedia, esto se debe a que dichos contenidos son añadidos dinámicamente por los usuarios, por lo que no se puede garantizar que los mismos cumplan las condiciones de accesibilidad tales como la presencia de subtítulos que se piden.

8.3.3 Observaciones de las pruebas de usuario y modificaciones motivadas por ellas

De las pruebas con usuarios se han extraído las siguientes cuestiones:

- La página inicial de la aplicación era la página de login por lo que los usuarios nuevos no tenían clara la finalidad del sistema antes de registrarse.
    - Se cambió esta página inicial por una página con información de la aplicación sus objetivos y ventajas.

- La página principal contenía un logotipo demasiado grande que para algunos tamaños de pantalla dificultaba el verla entera sin tener que hacer scroll.
    - Se redujo el tamaño de la imagen y la distribución de la página.

- Las opciones de cambio de idioma, cierre de sesión, cambio de password, etc. estaban situadas en la parte del pie de la página y se sugirió que estarían mejor situadas arriba a la derecha en una zona de utilidades.
    - Se creó una opción de menú junto a la ayuda para acceder a estas opciones.
    - Se eliminó un enlace a la página principal presente en el pie del diseño.

- La bandera del Reino Unido estaba asociada al cambio de idioma a inglés.
    - Se sustituyó por una a medias con la bandera de E.E.U.U.

- En algunos formularios de guardado de entidades, el botón para enviarlo contenía el texto “enviar” que era poco explicativo de la operación que se iba a realizar de guardado.
    - Se sustituyeron dichos textos por “guardar”, u otros textos más adecuados a cada contexto específico.

- El tamaño de los botones de guardar o cancelar cambios resultaba demasiado pequeño y su orden inadecuado respecto a los botones anteriores.
    - Se reajustó para hacerlos más llamativos e intuitivos.
- La ayuda de la aplicación no estaba del todo completa, faltaba explicar cómo confirmar el uso de plugins en la página de reproducción, y contenía algún error tipográfico.
    - La redacción de la ayuda fue completada y los errores subsanados.
    - Se explicó el funcionamiento de sugerencia de medios en la ayuda.

- La distribución de los campos del formulario de añadir nuevos medios en la página propia y en la inserción inline en la página de listar medios eran diferentes, pudiendo llevar a confusión a los usuarios sobre si hacían lo mismo.
    - Se unificó su diseño y distribución. De los elementos de ambos formularios.

- Al ver los medios de una lista de reproducción solo se mostraba el nombre del mismo, lo cual podía no dejar claro de qué medio se trataba en el caso de haber varios con nombre parecido

  - Se completó esta información mostrando la fuente del archivo también.

- A la hora de añadir un medio a una lista se desplegaba un selector de medios donde estos no estaban ordenados alfabéticamente, sino por fecha de creación, dificultando la búsqueda.
  - Se reordenaron los medios alfabéticamente.

- Al reproducir una lista se ofrecía poca información sobre el estado de reproducción, se sugirió añadir el número de medio que se estaba reproduciendo o su nombre.
  - Se añadió un texto adicional al estado de reproducción mostrando el número del medio actual y el número total de medios en dicha lista.

- En el pie de la página debería aparecer información de contacto, y fecha de última revisión
  - Se añadió al pie de la página información de contacto y última revisión.

- Errores tipográficos en la página de login.
  - Subsanados.

- Se sugirió poner el botón de cerrar sesión en una opción de menú aparte.
  - Esta petición se descartó porque al incluir el nombre del usuario, podría hacer que la distribución del menú pasara a ocupar más de una línea estorbando en el resto de la interfaz de la página. Considerando además que el cierre de sesión no es una acción habitual pues la aplicación no contiene información comprometida del usuario, y la sesión caduca automáticamente al cabo de cierto tiempo.

- Al iniciar sesión se mostraba la página principal, ya conocida por los usuarios en vez de una página en la que realmente pudieran hacer cosas “útiles”.
  - Tras iniciar sesión los usuarios estándar son enviados a la página de distado de medios donde pueden ver sus medios disponibles y añadir otros nuevos.
- Los botones de control de reproducción en la web incluían el texto de su función superpuesto, que ya era sobradamente entendida con sus iconos.
  - Se eliminó este texto y se cambió por un tooltip para despejar cualquier duda restante.

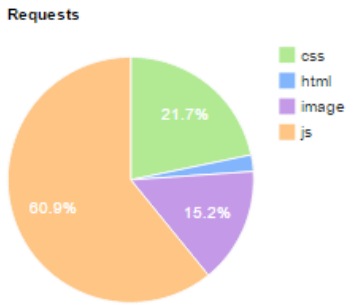
- La importancia y utilidad de la aplicación móvil no se resaltaba de manera que quedase clara su necesidad para los nuevos usuarios.
  - Se explicó en la página principal la utilidad de la aplicación móvil y como obtenerla de manera sencilla, abriendo la misma página desde su dispositivo móvil en el que aparecerá un botón de descarga.

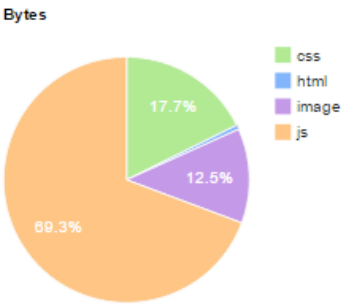
8.4 Pruebas de Rendimiento

Explicamos el desarrollo, resultados y conclusiones derivados de la ejecución de todas las pruebas de rendimiento que especificadas para el sistema, según lo que hemos diseñado anteriormente.

Se ha utilizado la herramienta web de test de rendimiento disponible en http://www.webpagetest.org/

8.4.1 Peticiones y tamaño de los recursos pedidos según el tipo





MIME Type	Requests
js	28
css	10
image	7
html	1
flash	0
font	0
other	0

MIME Type	Bytes
js	1234641
css	314576
image	221892
html	10410
flash	0
font	0
other	0

Figura 8.24 Reparto de peticiones y peso de las mismas al solicitar la página principal de la aplicación

Como se aprecia en los gráficos y tablas de la figura anterior, el mayor peso en cuanto a número de peticiones cuando se solicita una página de la aplicación se corresponde con los ficheros javascript, que suman más del 60% de las peticiones, así como del tamaño.

Tras los ficheros de javascriopt, el siguiente contenido en cuanto a volumen son los ficheros de CSS que dan estilo a la aplicación.

Este reparto de pesos se debe a la utilización de bibliotecas Bootstrap, JQuery, Dojo, etc.

No obstante estos recursos se reutilizan en la mayoría de páginas de la aplicación por lo que no es necesario traerlos del servidor de nuevo en todas las peticiones de páginas sino que se mantienen y reutilizan desde la caché del navegador

8.4.2 Tiempos de carga de la página de carga

							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
5.568s	0.530s	4.391s	5.591s	4505	101	0	5.568s	47	1,740 KB	5.729s	48	1,741 KB

RUM First Paint	domContentLoaded	loadEvent
4.335s	4.336s - 4.364s (0.028s)	5.560s - 5.564s (0.004s)

Figura 8.25 Tiempo de carga de la página

El tiempo de carga de la página en la primera ocasión es de más de 5 segundos, lo cual es bastante elevado, no obstante en sucesivas peticiones de páginas de la aplicación, éstas cargarán más rápido al no tener que traer todos los recursos javaascript y CSS.

El índice de velocidad de la tabla anterior (Speed index) es el tiempo medio medido en milisegundos en el que las partes visibles de la página son mostradas.

8.4.2.1 Vista en cascada

La vista en cascada se aprecia el reparto elemento a elemento del tiempo de carga de la página.

Como se puede ver, algunos elementos se cargan paralelamente, hay un retraso de unos 100 milisegundos entre que se realiza la petición del recurso en concreto y se inicia la transacción del mismo.

Los ficheros más grandes y que más demoran en recibirse se corresponden con archivos de CSS y javascript, como:

- **CSS:** bootstrap.min.css, tundra.css
- **Javascript:** dojo.js jquery-min.js jquery-ui.min.js, tinymce.min.js

Aparte de los anteriores, la imagen del logotipo de la página principal (mediaServerBig.png), también tarda cierto tiempo ya que su tamaño es mayor que el resto de imágenes utilizadas en la aplicación.

Respecto a los gráficos que utilización de CPU, y ancho de banda, señalar que como es lógico la mayor utilización de ancho de banda se produce cuando se descargan los contenidos más grandes y prolongados, y la mayor utilización de CPU se produce cuando se tienen todos los contenidos principales para empezar a dibujar la pantalla, punto cercano a los 4.5 segundos que coindice con el índice de velocidad que mencionamos en el apartado anterior de tiempos de carga.

Una opción que se podría considerar en el futuro es evaluar el contenido y necesidad de todos estos ficheros en la aplicación, y reunir el contenido de los mismos que efectivamente se

utilice en un número menor de ficheros, para reducir el número de peticiones que un cliente tiene que hacer antes de poder cargar una página de la aplicación.

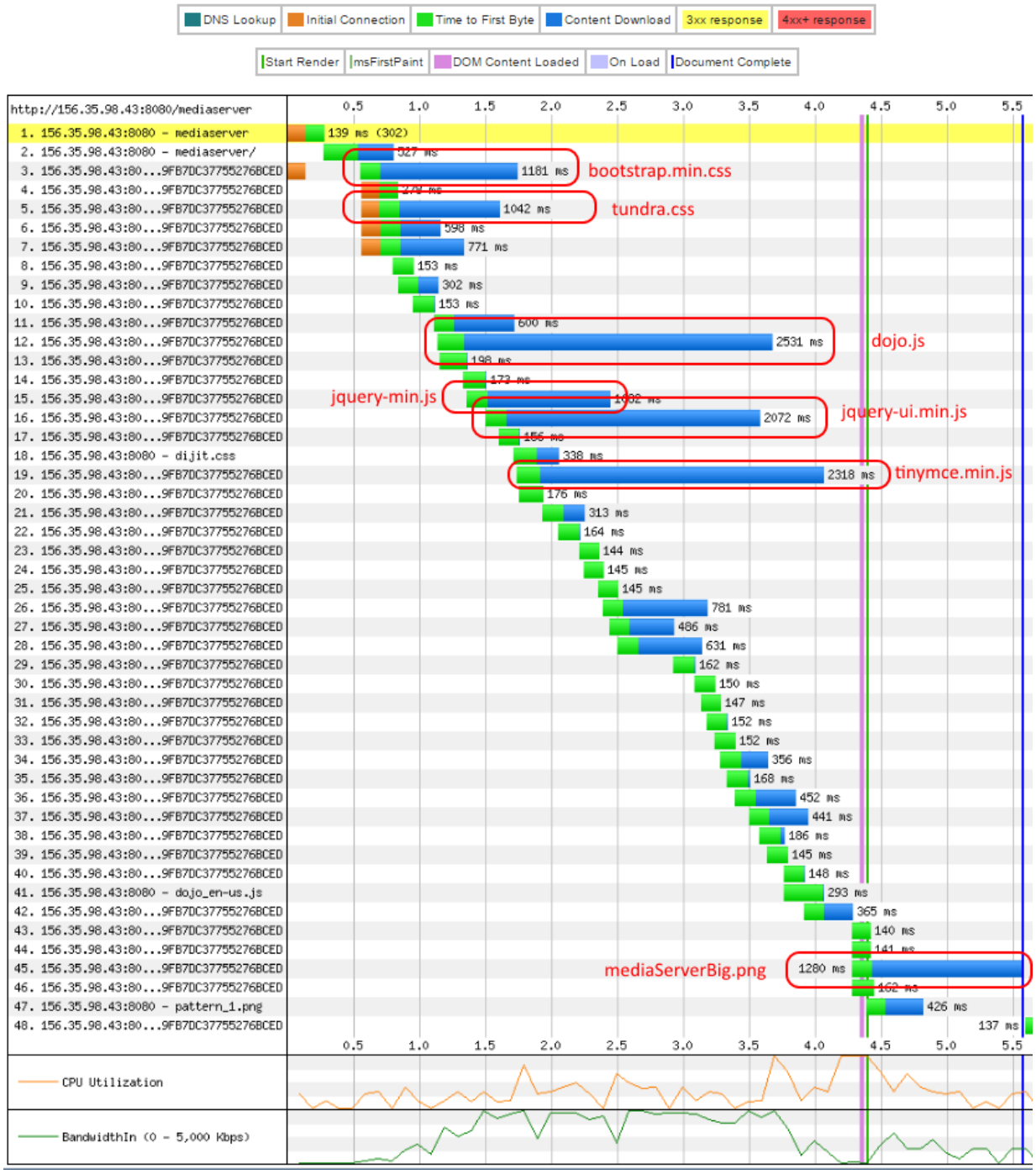


Figura 8.25 Vista en cascada de los tiempos de carga de los recursos de la página principal de la aplicación

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

Para instalar y poner en funcionamiento el sistema Mediaserver es necesario:

9.1.1 Java

El equipo en el que se ejecute debe tener instalado Java en su versión 7 o posterior. Se puede adquirir y consultar más información en:

http://www.java.com/es/download/

Añadimos la variable JAVA_HOME a las variables de entorno del sistema.

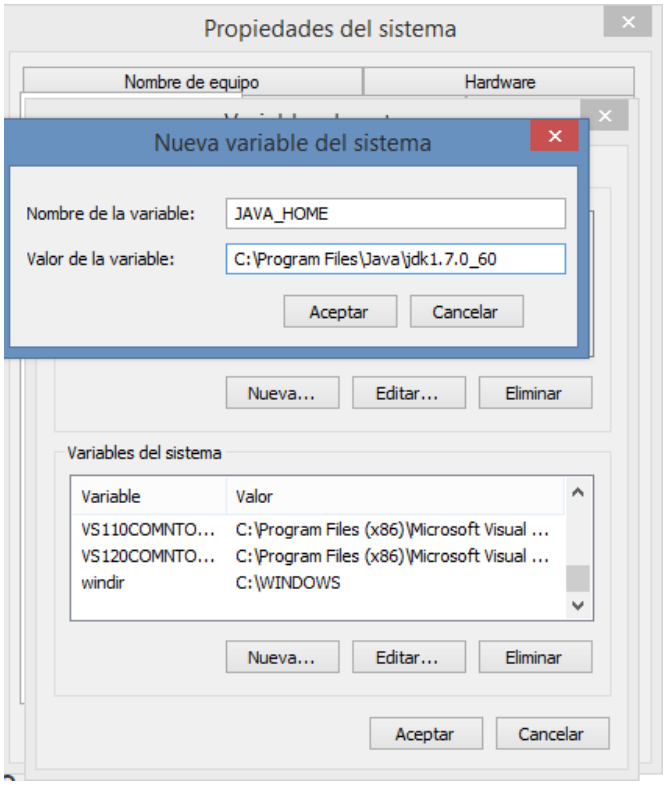


Figura 9.1 Crear variableJAVA_HOME

Añadimos al PATH la nueva variable.

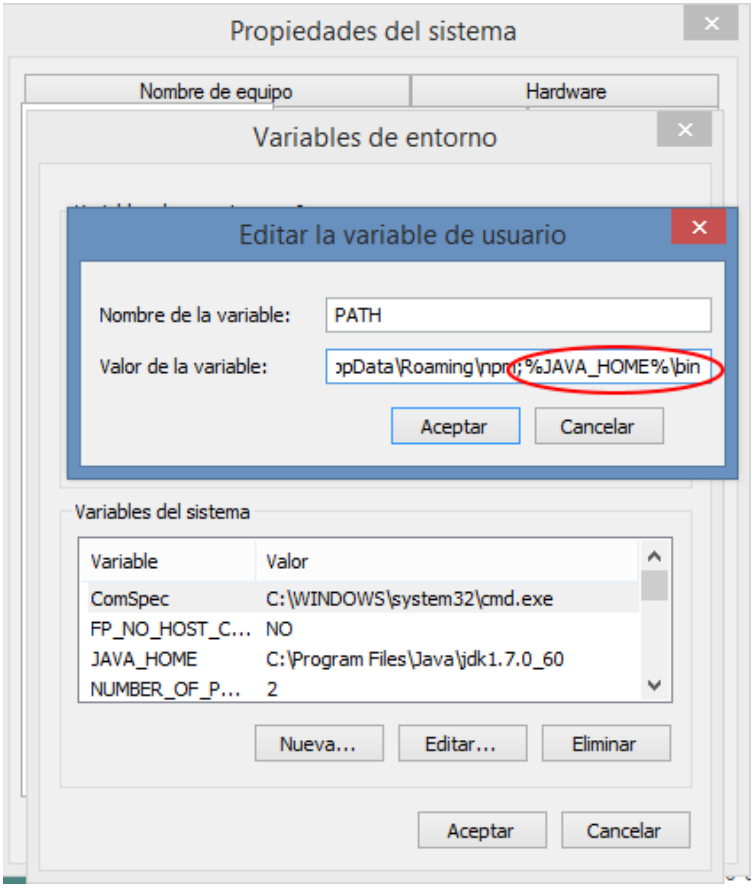


Figura 9.2 Añadir JAVA_HOME al PATH del sistema

9.1.2 MySQL

Ha de disponerse de una base de datos MySql. Esto se puede hacer de manera sencilla en Windows instalando un paquete como Appserv (2.5.7), compuesto por:

- Apache Web Server (versión 2.2.3)
- Lenguaje PHP (versión 5)
- Base de datos MySQL (versión 5.0.24)
- Manejador de base de datos phpMyAdmin (versión 2.9.0.2)

Este paquete se puede obtener de:

http://www.appservnetwork.com/index.php?newlang=spanish

Haciendo clic en el icono de instalación, se abrirá un instalador como el siguiente:

Figura 9.3 Instalador de AppServ

Tras aceptar la licencia y especificar el directorio de instalación, seleccionaremos los componentes que deseamos instalar:

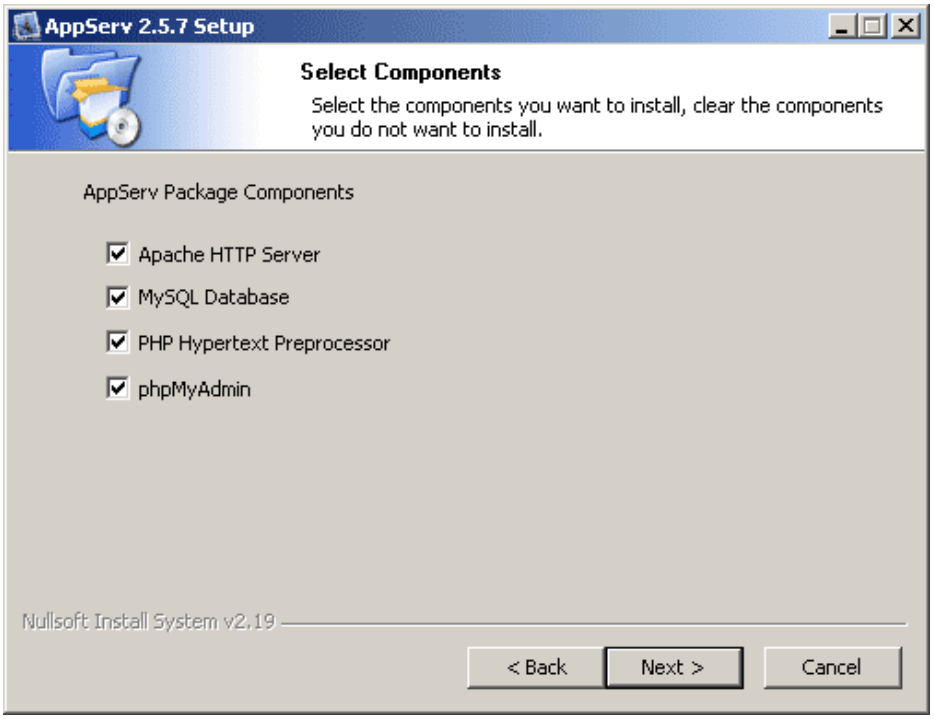


Figura 9.4 Selección de componentes de instalación AppServ

En el siguiente paso especificaremos el nombre del servidor y el email del administrador por si fuera necesaria alguna notificación por este medio.

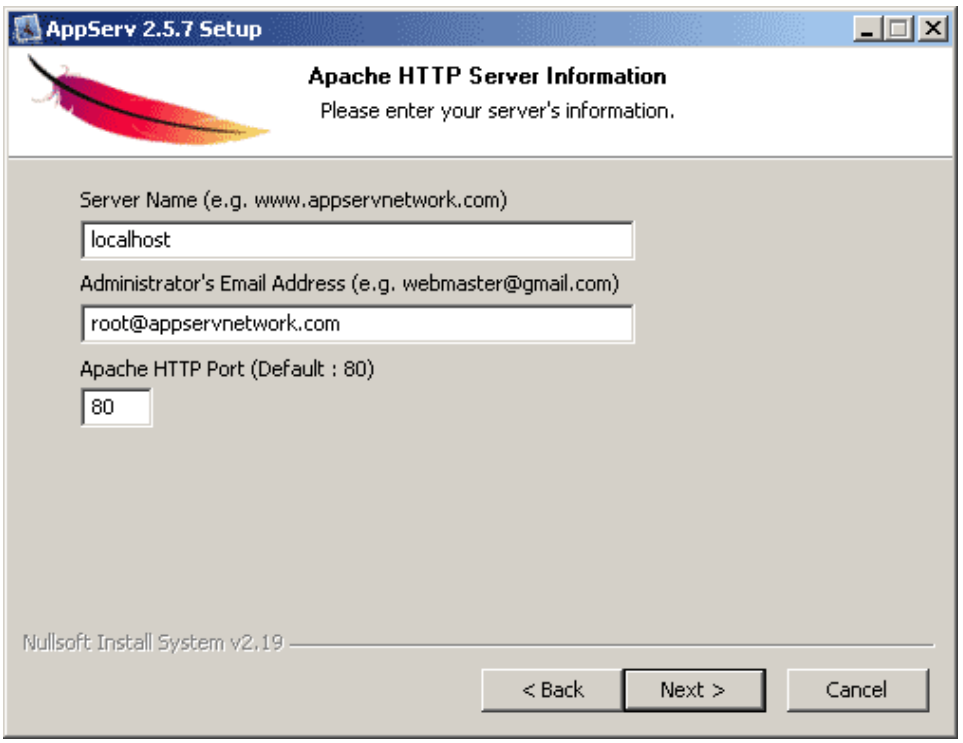


Figura 9.5 Instalación Apache AppServ

A continuación configuraremos la base de datos MySQL indicando la pawword del usuario principal de la misma (root) que luego será usuado para acceder a la base de datos.

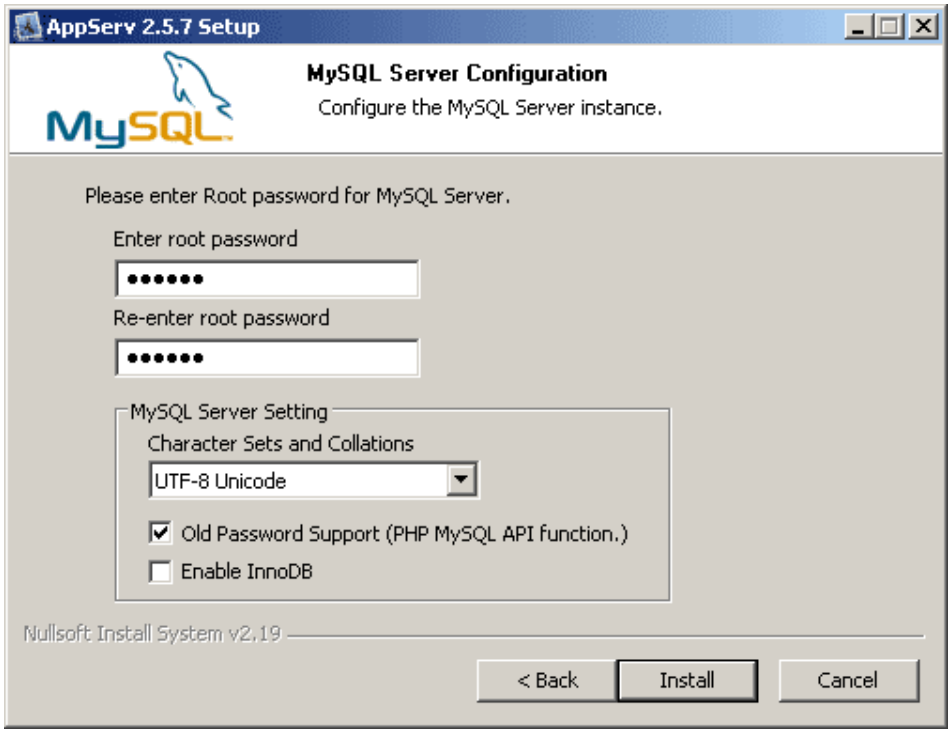


Figura 9.6 Instalación MySQL AppServ

Para culminar la instalación podemos elegir iniciar inmediatamente Apache server y la base de datos MySQL

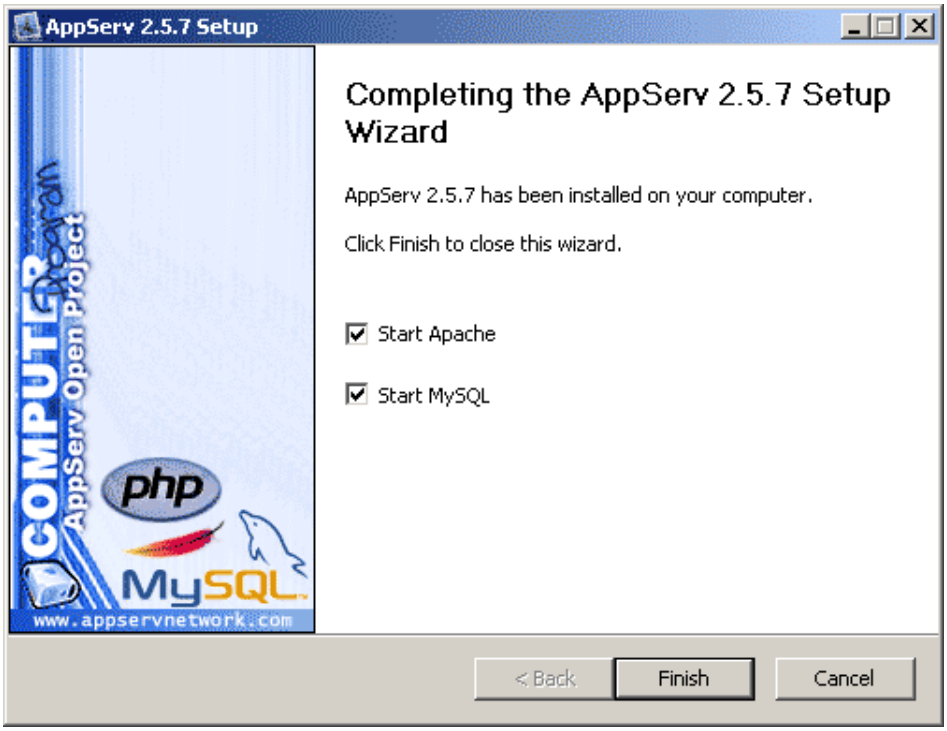


Figura 9.7 Conclusión instalación AppServ

Una vez hecho esto ya se puede acceder a la página de configuración y control de la base de datos en la dirección: http://localhost y desde ella crear el usuario y base de datos específica para la aplicación Mediaserver.

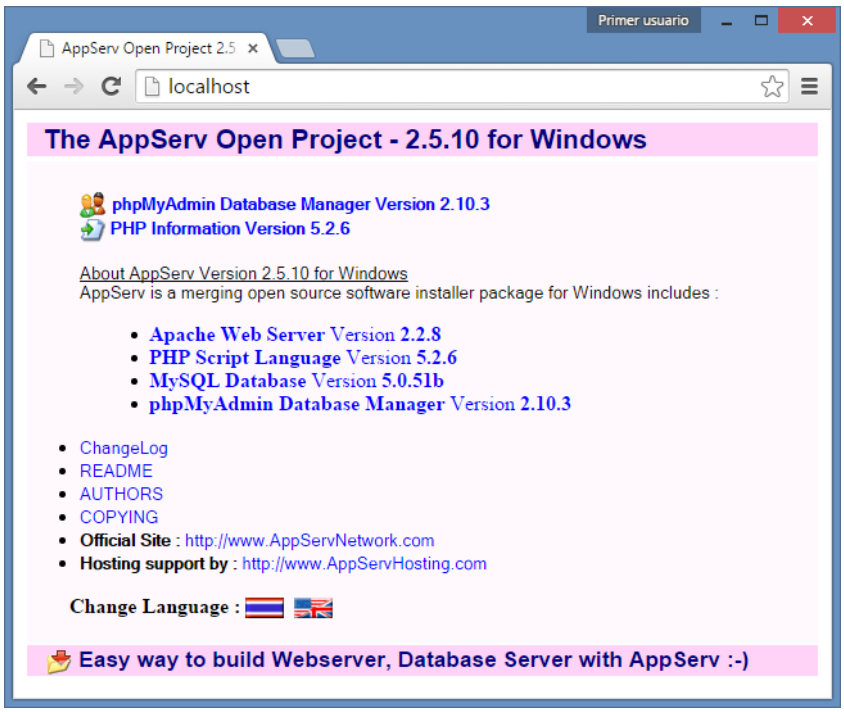


Figura 9.8 Acceso a phpMyAdmin

Accederemos con el nombre de usuario 'root' y la password que hayamos especificado durante la instalación.

Hace falta crear la base de datos 'mediaserver' y usuario de la aplicación, en este caso llamado 'local'-'local' y conferirle privilegios sobre la base de datos de la aplicación.

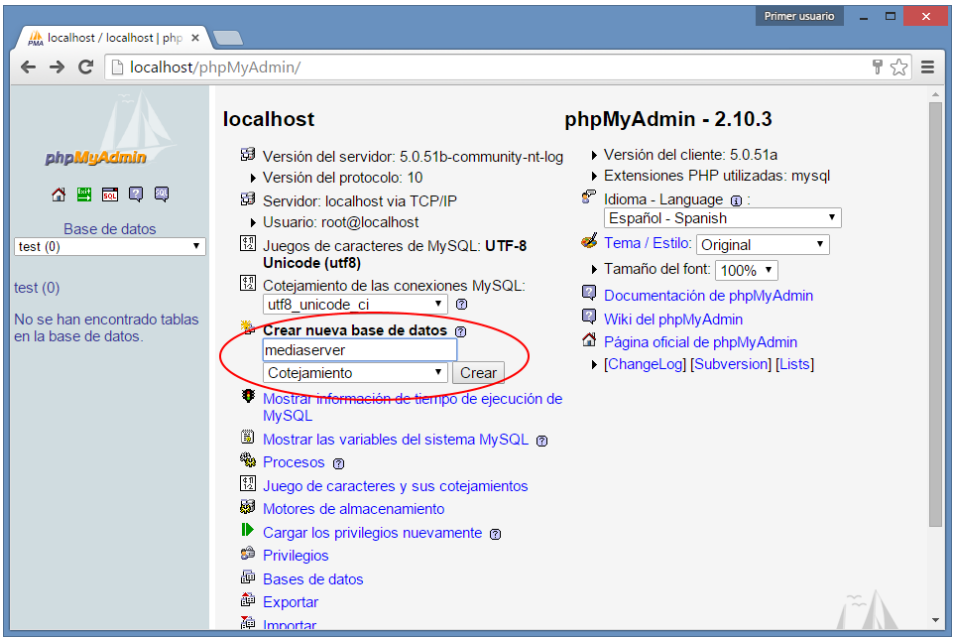


Figura 9.9 Creación de una nueva base de datos

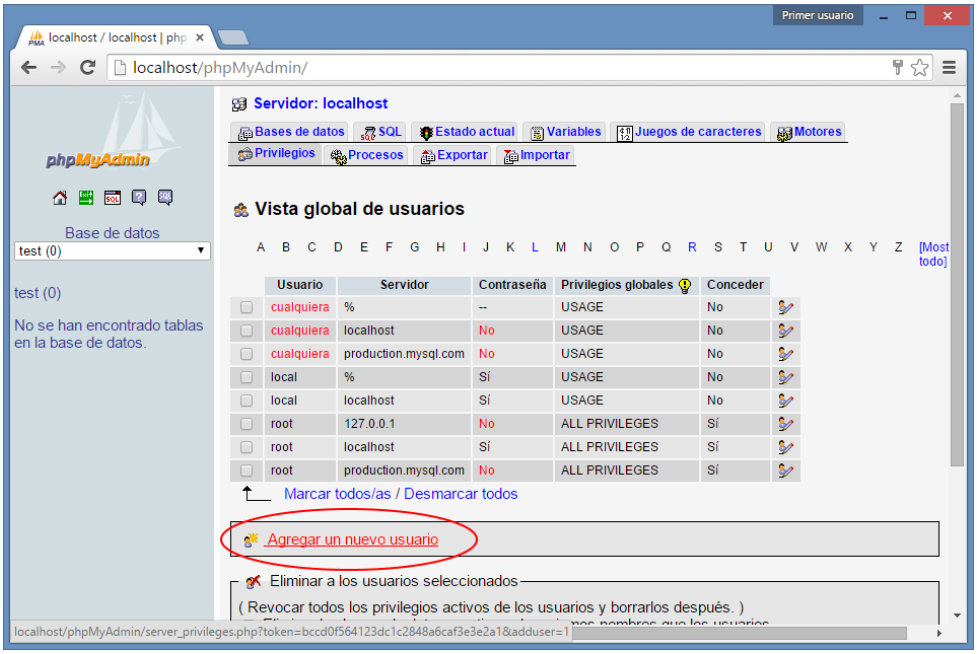


Figura 9.10 Agregar un usuario a la base de datos

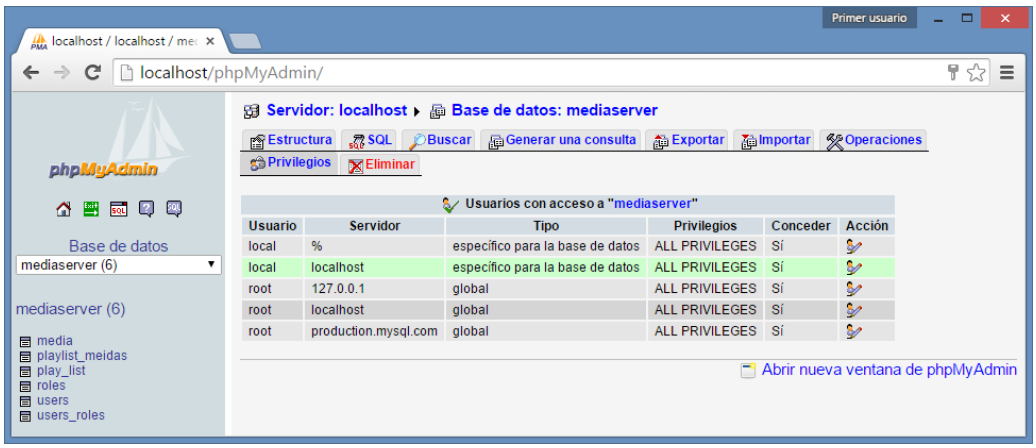


Figura 9.11 Conceder privilegios al usuario sobre la base de datos de la aplicación

Con el siguiente script crearemos la estructura de tablas y datos iniciales necesarios de la base de datos.

```
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
--
-- Base de datos: `mediaserver`
--
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `media`
--
CREATE TABLE `media` (
  `id` bigint(20) NOT NULL auto_increment,
  `name` varchar(30) default NULL,
  `source` varchar(255) default NULL,
  `type` int(11) default NULL,
  `version` int(11) default NULL,
  `owner` bigint(20) default NULL,
  PRIMARY KEY  (`id`),
  KEY `FK_t9vitkb20v9q24pkwhjco4m19` (`owner`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=20 ;
--
-- Estructura de tabla para la tabla `playlist_meidas`
--
CREATE TABLE `playlist_meidas` (
  `id` bigint(20) NOT NULL auto_increment,
  `version` int(11) default NULL,
  `media_id` bigint(20) NOT NULL,
  `playlist_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `FK_amvcjyyvvj79le2xwxtgyr22i` (`media_id`),
  KEY `FK_5331tdtulbe2dhg26h1gkhhc6` (`playlist_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=5 ;
--
-- Estructura de tabla para la tabla `play_list`
--
CREATE TABLE `play_list` (
  `id` bigint(20) NOT NULL auto_increment,
  `play_list_name` varchar(30) default NULL,
  `version` int(11) default NULL,
  `owner` bigint(20) default NULL,
  PRIMARY KEY  (`id`),
  KEY `FK_jnafh6otrab9dpuq75hocshhv` (`owner`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=4 ;
--
-- Estructura de tabla para la tabla `roles`
--
CREATE TABLE `roles` (
  `id` bigint(20) NOT NULL auto_increment,
  `description` varchar(200) NOT NULL,
  `name` varchar(255) NOT NULL,
  `version` int(11) default NULL,
  PRIMARY KEY  (`id`),
```

```
  UNIQUE KEY `UK_ofx66keruapi6vyqpv6f2or37` (`name`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=3 ;
--
-- Volcar la base de datos para la tabla `roles`
--
INSERT INTO `roles` VALUES (1, 'user', 'USER', 0);
INSERT INTO `roles` VALUES (2, 'admin', 'ADMINISTRATOR', 0);
--
-- Estructura de tabla para la tabla `users`
--
CREATE TABLE `users` (
  `id` bigint(20) NOT NULL auto_increment,
  `activation_date` datetime default NULL,
  `email_address` varchar(255) NOT NULL,
  `enabled` tinyint(1) default NULL,
  `first_name` varchar(255) NOT NULL,
  `last_name` varchar(255) NOT NULL,
  `locked` tinyint(1) default NULL,
  `password` varchar(255) default NULL,
  `version` int(11) default NULL,
  PRIMARY KEY  (`id`),
  UNIQUE KEY `UK_1ar956vx8jufbghpyi09yr16l` (`email_address`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
--
-- Estructura de tabla para la tabla `users_roles`
--
CREATE TABLE `users_roles` (
  `id` bigint(20) NOT NULL auto_increment,
  `version` int(11) default NULL,
  `role_id` bigint(20) NOT NULL,
  `user_id` bigint(20) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `FK_k2mq1ee4ob6uw649wgaus1ate` (`role_id`),
  KEY `FK_1hjw31qvltj7v3wb5o31jsrd8` (`user_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 AUTO_INCREMENT=2 ;
--
-- Filtros para la tabla `media`
--
ALTER TABLE `media`
  ADD CONSTRAINT `FK_t9vitkb20v9q24pkwhjco4m19` FOREIGN KEY (`owner`) REFERENCES `users`
(`id`);
--
-- Filtros para la tabla `playlist_meidas`
--
ALTER TABLE `playlist_meidas`
  ADD CONSTRAINT `FK_5331tdtulbe2dhg26h1gkhhc6` FOREIGN KEY (`playlist_id`) REFERENCES
`play_list` (`id`),
  ADD CONSTRAINT `FK_amvcjyyvvj79le2xwxtgyr22i` FOREIGN KEY (`media_id`) REFERENCES
`media` (`id`);
--
-- Filtros para la tabla `play_list`
--
ALTER TABLE `play_list`
  ADD CONSTRAINT `FK_jnafh6otrab9dpuq75hocshhv` FOREIGN KEY (`owner`) REFERENCES `users`
(`id`);
--
-- Filtros para la tabla `users_roles`
--
ALTER TABLE `users_roles`
  ADD CONSTRAINT `FK_1hjw31qvltj7v3wb5o31jsrd8` FOREIGN KEY (`user_id`) REFERENCES
`users` (`id`),
  ADD CONSTRAINT `FK_k2mq1ee4ob6uw649wgaus1ate` FOREIGN KEY (`role_id`) REFERENCES
`roles` (`id`)
```

Figura 9.12 Consola SQL para preparar la base de datos de la aplicación.

9.1.3 Tomcat

Es necesario disponer de un servidor de aplicaciones java compatible con WebSockets y DataTables, como es Apache tomcat 7.0.50, disponible en.

http://archive.apache.org/dist/tomcat/tomcat-7/v7.0.50/

Dentro de la carpeta webapps situaremos el archivo “mediaserver.war” de la aplicación.

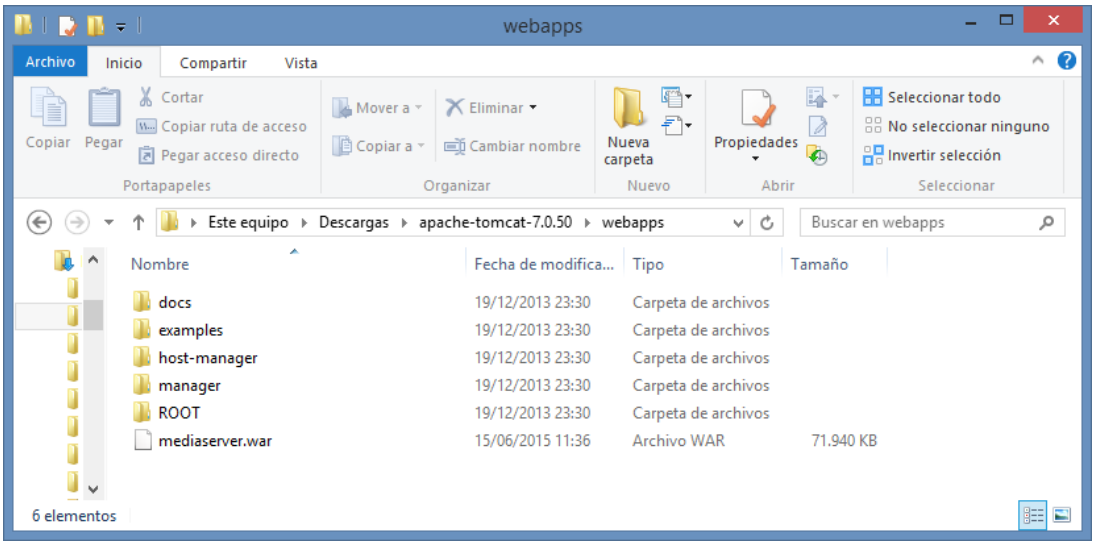


Figura 9.13 Colocando la aplicación en la carpeta de despliegue

9.1.4 Plugin web VLC

Como usuario de la aplicación es necesario tener instalado el plugin web de VLC, que viene incluido al instalar las últimas versiones del reproductor en su versión de escritorio.

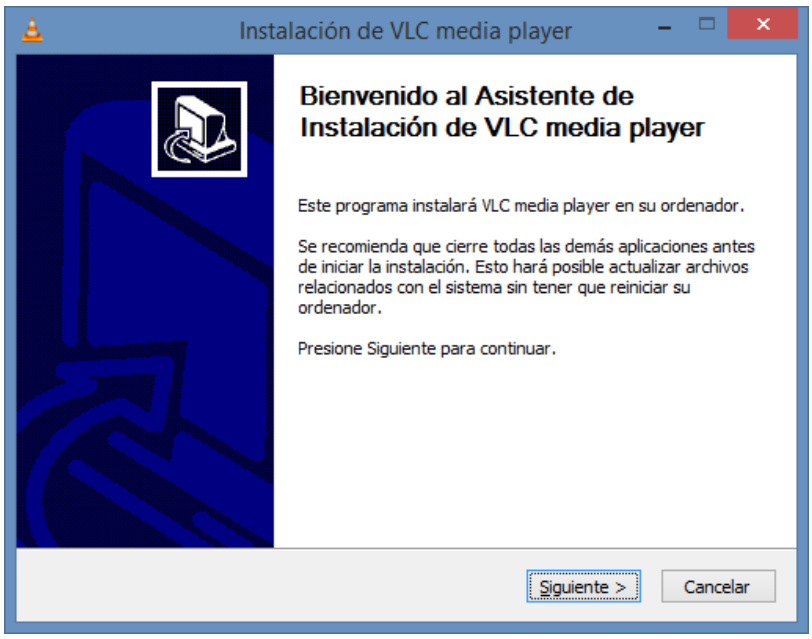


Figura 9.14 Instalación VLC player

Seleccionamos la instalación de complementos web.

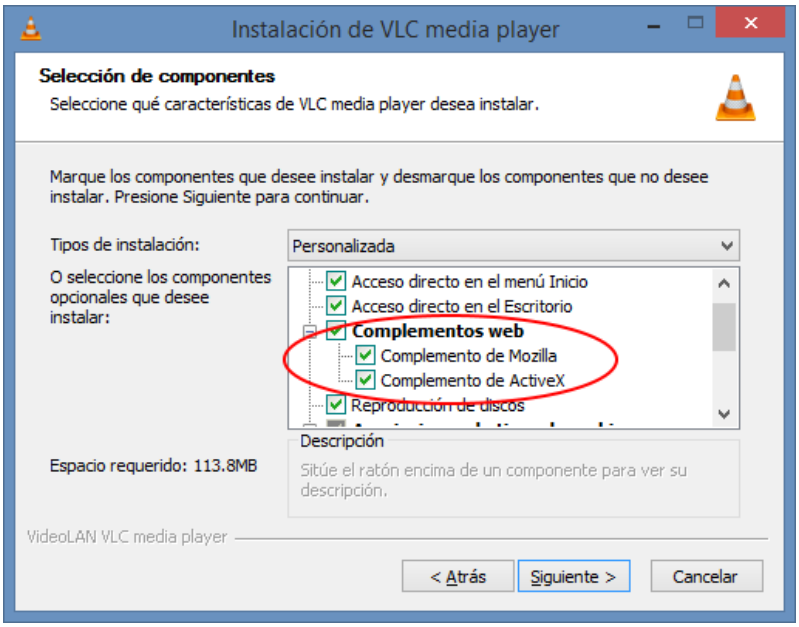


Figura 9.15 Selección de complementos web en la Instalación VLC player

También es necesario un navegador web moderno, compatible con el plugin y con javascript, tal como Chrome, Opera, o Firefox.

En el caso de Chrome es necesario habilitar el uso de plugins NPAPI en la siguiente página de configuración para que la reproducción funcione correctamente, y desbloquear su uso una vez llegado a la página de reproducción.

chrome://flags/#enable-npapi

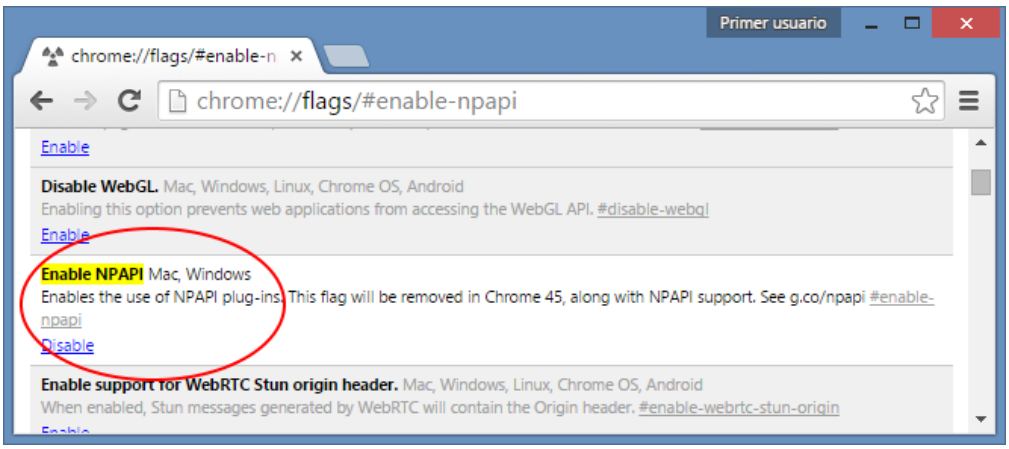


Figura 9.16 Habilitar plugins NPAPI en Google Chrome

9.1.5 Aplicación MediaserverRC

Para utilizar la aplicación Android de control remoto es necesario descargarla e instalarla desde el enlace proporcionado en la web de Mediaserver:

https://drive.google.com/file/d/0B6z1VakuQS8ob0hTUHJCRzYzM0E/view?usp=sharing

Figura 9.17 Página principal de la aplicación desde navegador móvil

9.2 Manual de Ejecución

Para ejecutar el servidor Mediaserver debemos abrir una consola de comandos, situarnos en el directorio bin de la carpeta del tomcat, y ejecutar “startup”.

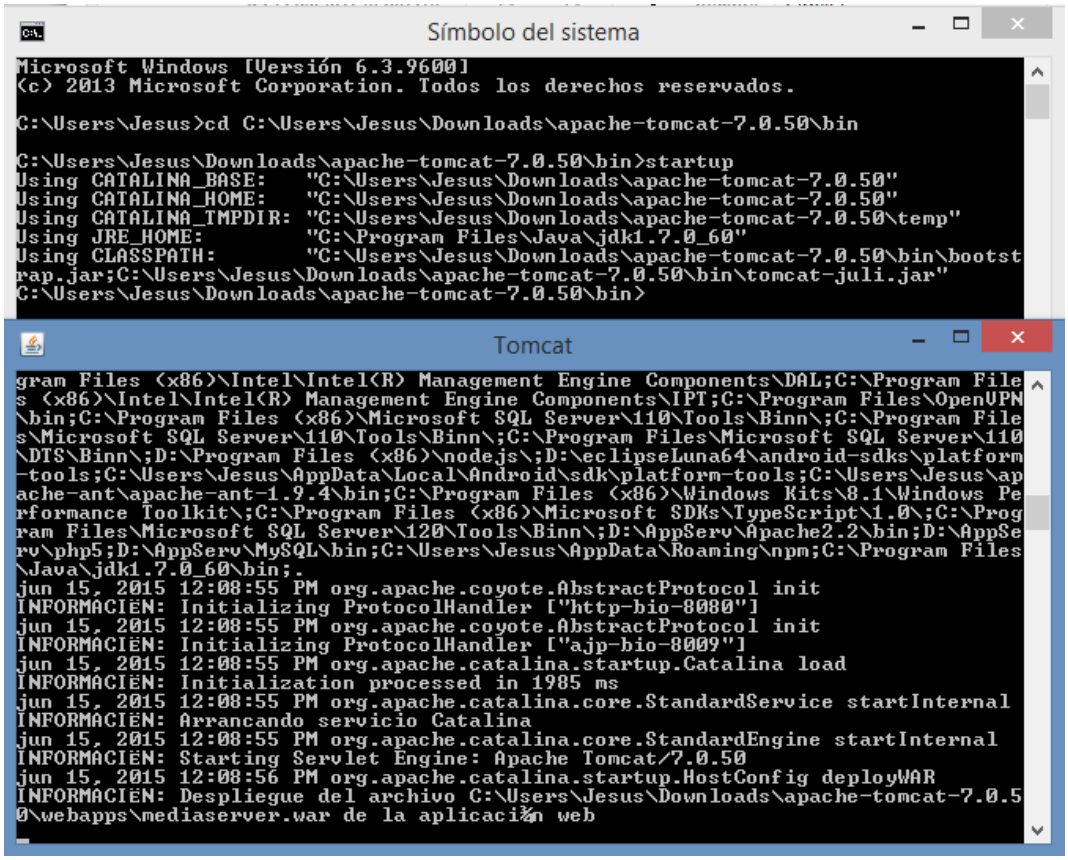


Figura 9.18 Ejecución del servidor de aplicaciones Tomcat

Si todo ha ido correctamente, la aplicación estaría accesible en la dirección:

http://localhost:8080/mediaserver

Los parámetros de configuración de la conexión a la base de datos, o envío de emails se pueden modificar en los archivos email.properties y database.properties en la carpeta de despliegue de la aplicación dentro del subdirectorio:

webapps\mediaserver\WEB-INF\classes\META-INF\spring

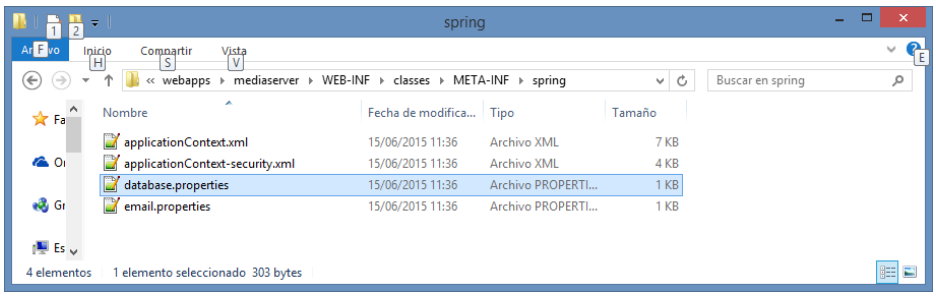


Figura 9.19 Modificación de propiedades de email y base de datos

Como usuario standard de la aplicación, solo hace falta acceder a la dirección de un servidor mediaserver para utilizar la aplicación, por ejemplo:

http://156.35.98.43:8080/mediaserver

Para utilizar la aplicación de control remoto no hace falta más que abrirla y configurarla en el dispositivo móvil como se explicará en el manual de usuario.

9.3 Manual de Usuario

En esta sección describiremos cómo funciona cada una de las partes de nuestra aplicación, las opciones de la misma y acciones posibles para cada uno de los usuarios del sistema.

9.3.1 Usuario Standard

9.3.1.1 Parte Web

Al entrar en el sitio web, primero no encontramos con la página principal de la aplicación.

Figura 9.20 Página principal de la aplicación

En ella nos encontramos con la descripción de la aplicación, su objetivo y utilidad. Además disponemos de unas opciones básicas de configuración, como son el cambio de idioma, accesible a través del menú de la esquina superior derecha o el cambio de tema, en el pie de la página.

El menú principal, como usuario anónimo solo ofrece la opción de iniciar sesión dirigiendo al usuario a la página de login. Si tras haber iniciado sesión volvemos visitar la página principal, está ya contará con el menú completo acorde al rol del usuario que haya iniciado sesión.

Al seleccionar la opción de iniciar sesión accederemos a la página de login.

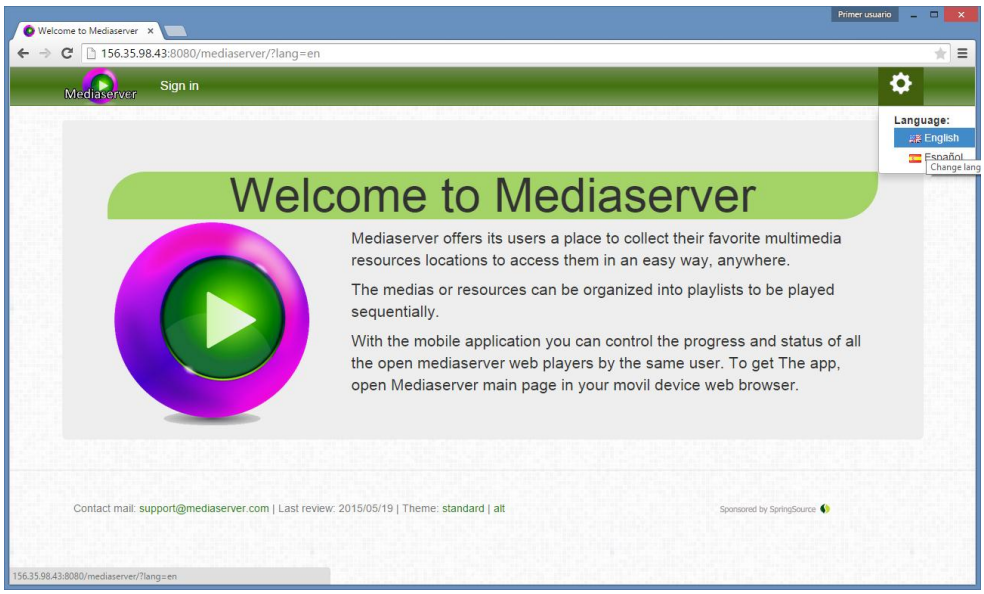


Figura 9.21 Página principal de la aplicación tras solicitar un cambio de idioma

Al solicitar un cambio de idioma, la página se recarga con los textos adaptados al idioma seleccionado y esta configuración se conserva de ahí en adelante mostrando todas las páginas en el idioma elegido. El idioma puede cambiarse de nuevo en cualquier momento y en cualquiera de las páginas de la aplicación.

Figura 9.22 Página principal de la aplicación tras solicitar un cambio de tema

Al solicitar un cambio de tema, la página se recarga cambiando el esquema de colores básico por el alternativo para facilitar que el usuario aprecie los contenidos si padece algún defecto visual que le haga distinguir mejor los mismos con el nuevo esquema de colores, o simplemente le gusta más.

La elección de tema se conserva para toda la navegación sucesiva, aunque el se puede cambiar de nuevo en cualquier momento y en cualquiera de las páginas de la aplicación.

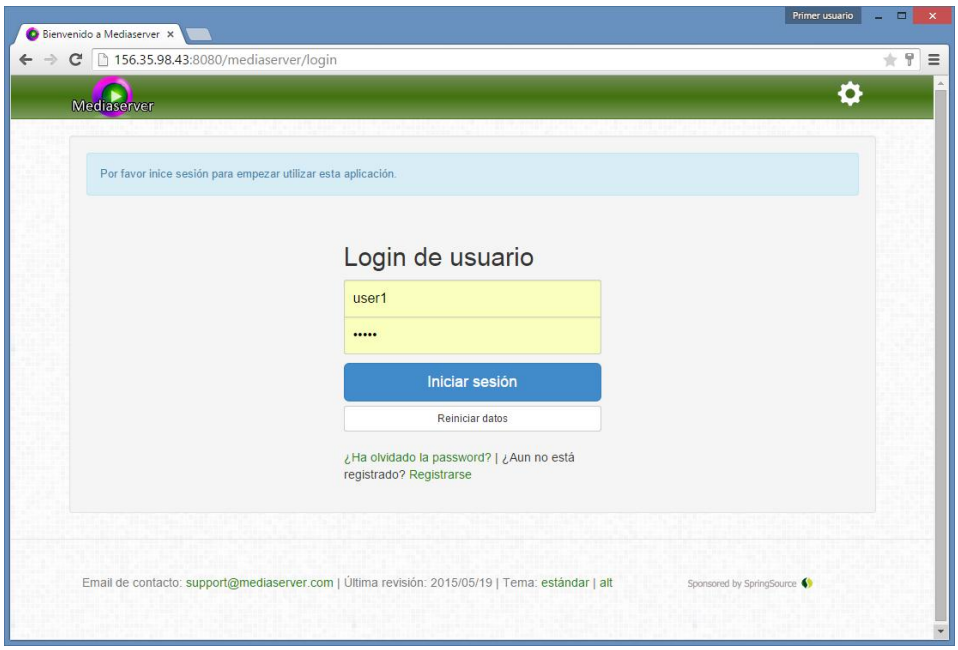


Figura 9.23 Página de inicio de sesión

Antes de empezar a utilizar las funciones de la aplicación el usuario ha de iniciar sesión en la pantalla de login a la que se accede a través de la opción de menú de la página principal de que hemos descrito anteriormente.

Si el usuario no dispone de una cuenta, se le ofrece un enlace al formulario de registro de nuevos usuarios, y si dispone de cuenta pero ha olvidado su contraseña también se le ofrece una opción para reestablecerla previa confirmación mediante su correo.

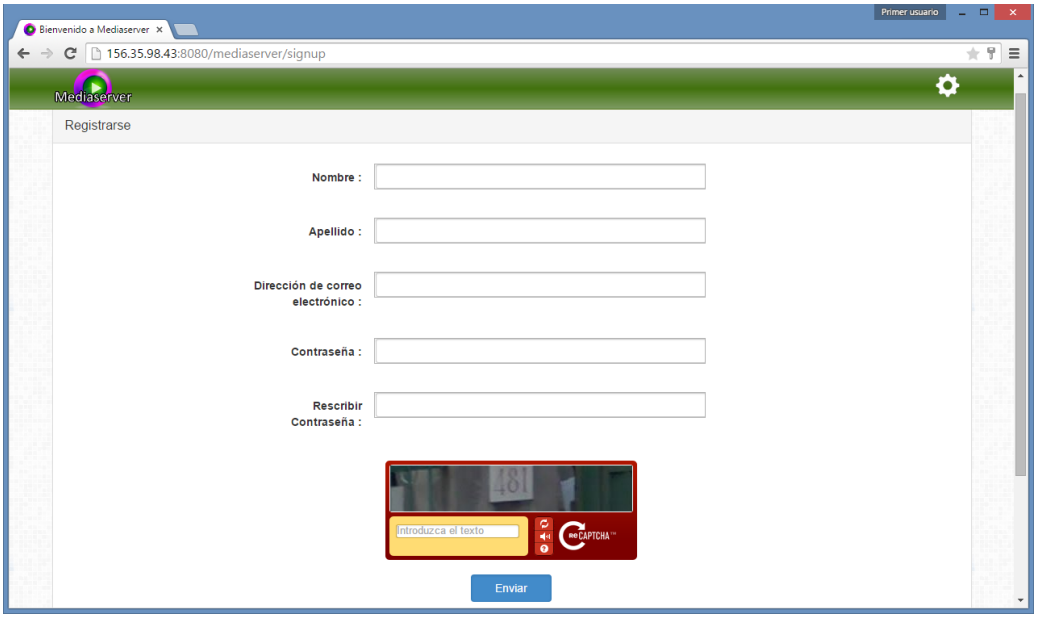


Figura 9.24 Página de registro

Para registrarse como nuevo usuario es necesario completar el formulario de la imagen anterior con los datos del usuario: nombre apellido, dirección de correo electrónico (que

funcionará como nombre de usuario y debe ser única en el sistema) y contraseña por duplicado para evitar errores fortuitos. Ambas contraseñas como es de lógico han de coincidir.

También ha de rellenarse un recaptcha para evitar creaciones automáticas de cuentas en el sistema.

Ante cualquier error en el formulario el usuario será avisado para poder corregir los datos introducidos.

Si el registro ha sido correcto, aparecerá la siguiente página de notificación indicando que hemos de recibir un email de activación en nuestro correo.

Figura 9.25 Notificación de registro correcto

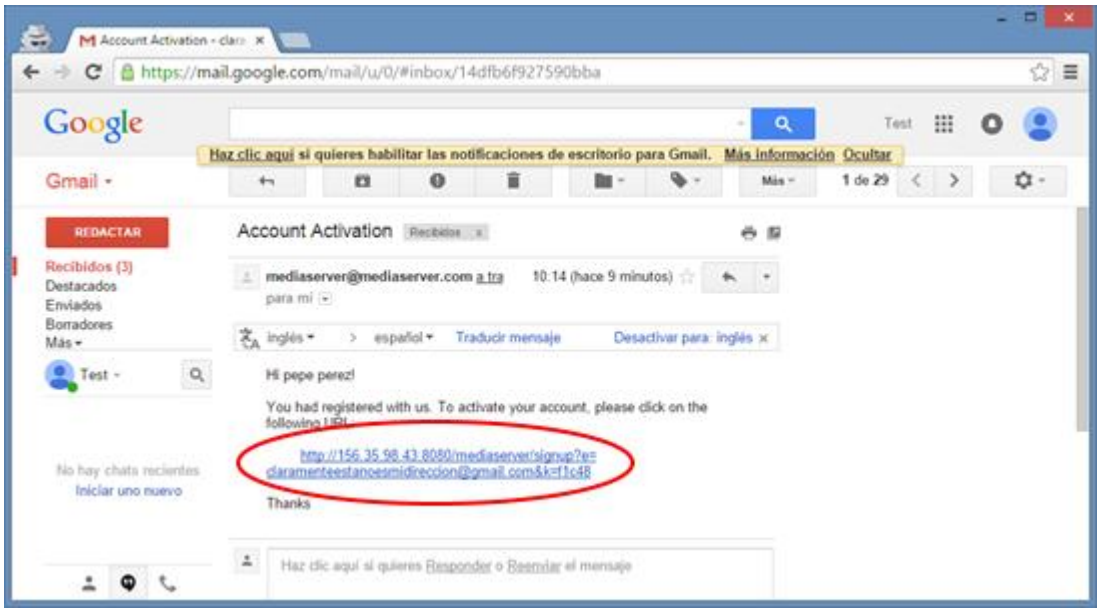


Figura 9.26 Email de activación

En la bandeja de nuestro correo encontraremos un email procedente del servidor, con un enlace de activación al que debemos acceder.

El enlace nos llevará de nuevo a la página de login, en la que aparecerá un mensaje informándonos de que la cuenta ha sido activada correctamente.

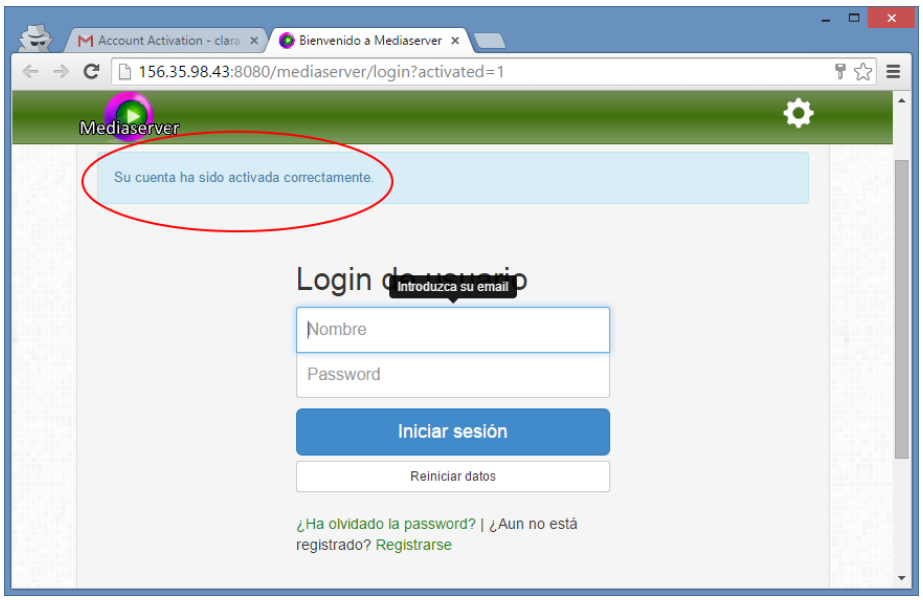


Figura 9.27 Vuelta al login a través del enlace de activación.

Para iniciar sesión introduciremos la dirección de correo y contraseña tal como las establecimos en la página de registro.

Si por cualquier motivo se nos olvida la contraseña o la hemos escrito mal en el registro y no podemos acceder a la cuenta, podemos solicitar su restablecimiento a través de la opción de recuperación, situada bajo el formulario de login con el texto "¿Ha olvidado la password?", lo cual nos conducirá al siguiente formulario

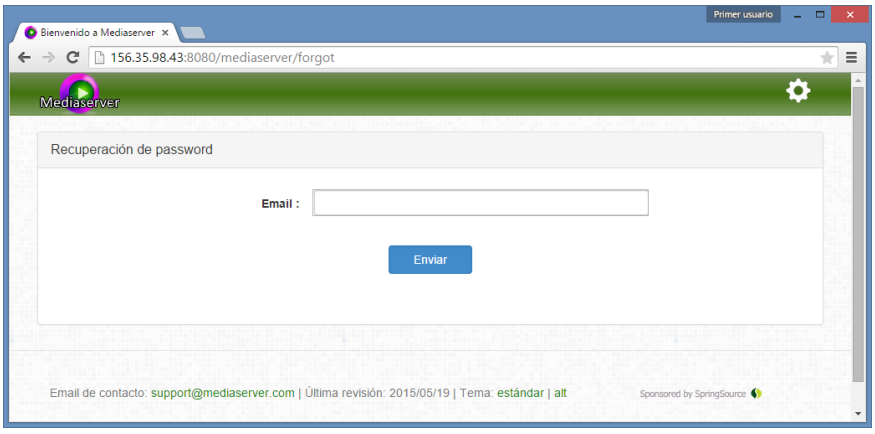


Figura 9.28 Página de recuperación de contraseña

Para recuperar la contraseña ha de introducirse la dirección de email del usuario para poder enviar el email con el enlace para el restablecimiento.

Si se ha introducido correctamente la dirección de email, aparecerá la siguiente página de notificación indicando que hemos de recibir un email de restablecimiento en nuestro correo.

Figura 9.29 Notificación de envío de email de restablecimiento de password

En la bandeja de nuestro correo encontraremos un email procedente del servidor, con un enlace de restablecimiento al que debemos acceder.

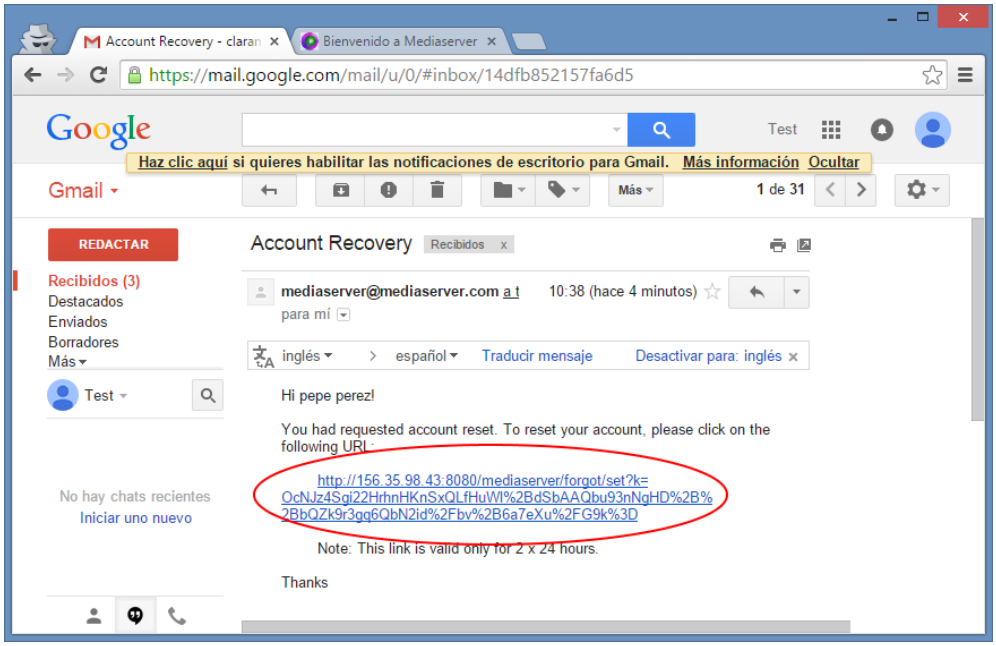


Figura 9.30 Email de restablecimiento de password

El enlace nos llevará a una página como la siguiente:

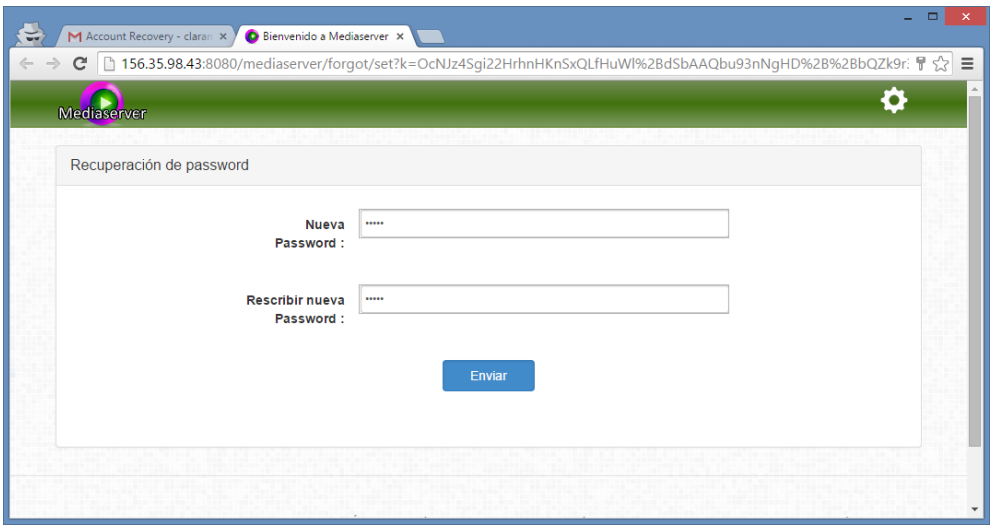


Figura 9.31 Formulario de restablecimiento de password

En este formulario introduciremos por duplicado la nueva password que se establecerá como nueva contraseña del usuario. Al finalizar el proceso volveremos a la página de login.

Una vez el iniciada sesión introduciendo las credenciales en la página de login, accederemos al listado de medios.

El menú habrá cambiado con nuevas opciones, para la gestión y visualización de medios y listas de reproducción, ayuda, cambio de password y cierre de sesión.

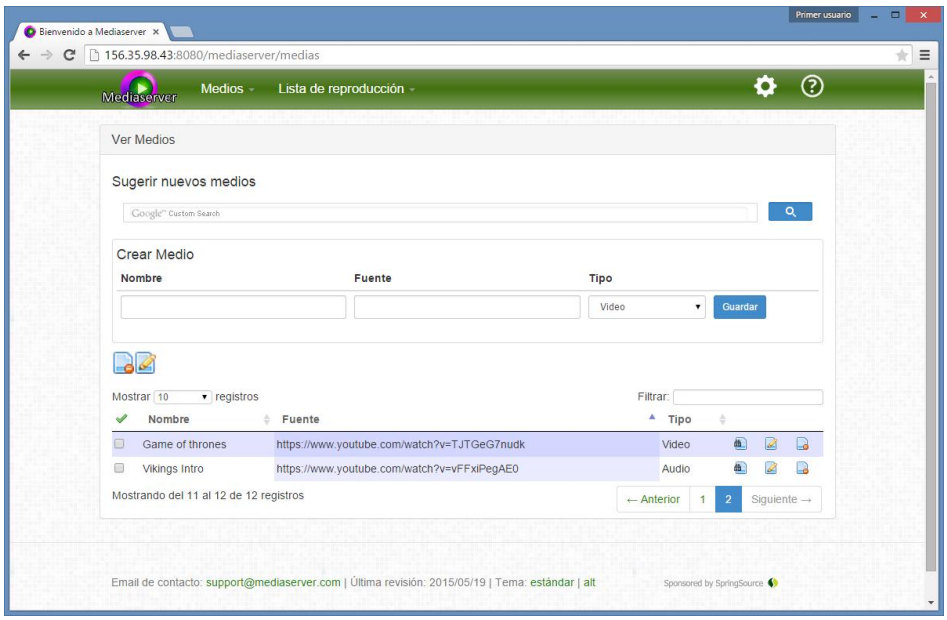


Figura 9.32 Página de listado de medios

En esta página, podremos revisar los medios que ya hemos guardado, editarlos, borrarlos o crear nuevos medios.

Para crear un nuevo medio solamente debemos introducir un nombre, fuente y tipo de medio y hacer clic en el botón de guardar en el formulario “Crear Medio”

Como es posible que no sepamos la dirección o fuente de los medios que nos interesan, disponemos de una útil herramienta de búsqueda y sugerencia, donde encontrar nuevos medios a partir de únicamente su nombre.

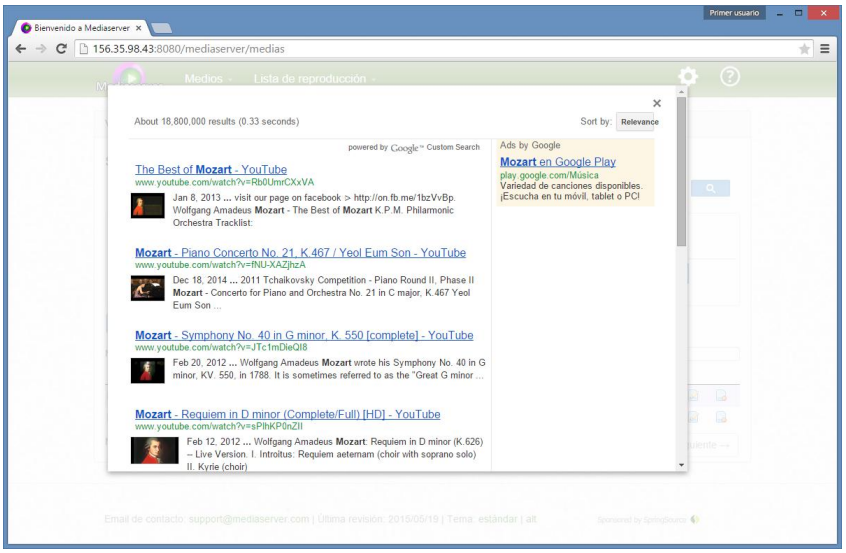


Figura 9.33 Búsqueda de nuevos medios con la herramienta de sugerencia

Al hace una búsqueda con la herramienta de sugerencia, aparecerán los resultados en una página superpuesta para poder elegir cuál de ellos nos interesa, y al hacerlo, los datos del medio se rellenarán automáticamente en el formulario para añadirlo a la colección haciendo clic en el botón “Guardar”, lo cual hará que se añada al final de la lista.

La relación de elementos se puede reordenar según las distintas columnas, se pueden filtrar los medios mostrados, cambiar el número máximo de medios por página, cambiar de página, etc.

A continuación se muestra en detalle la parte de la tabla del listado de medios:

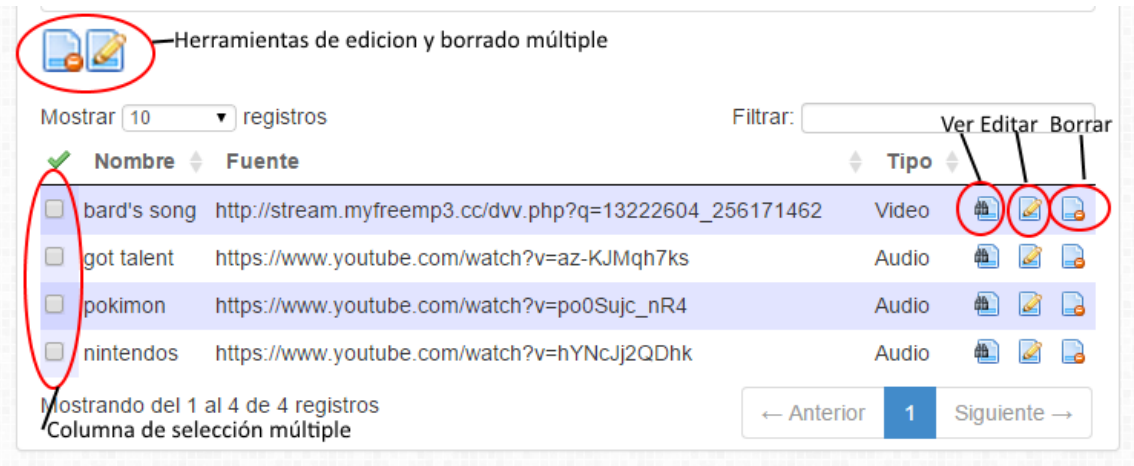


Figura 9.34 Detalle de la tabla de medios

Cuando se seleccionan filas de la tabla con su columna de selección, estas se resaltan en amarillo.

Figura 9.35 Tabla de medios con filas seleccionadas

Al hacer clic en el botón de edición múltiple situado sobre la tabla, con alguna fila seleccionada, los campos de estas filas pasan a ser editables para poder ajustar los valores de las mismas.

Además aparecen dos nuevos botones para aceptar y cancelar las ediciones realizadas.

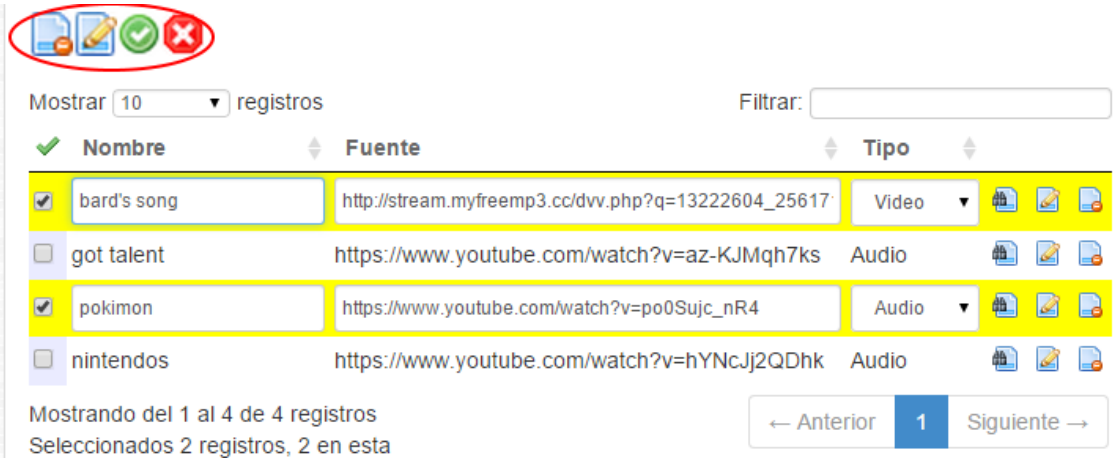


Figura 9.36 Edición de medios en la tabla

Si se selecciona borrar algún medio, se muestra un mensaje pidiendo confirmación para evitar borrados accidentales como se muestra en la siguiente imagen.

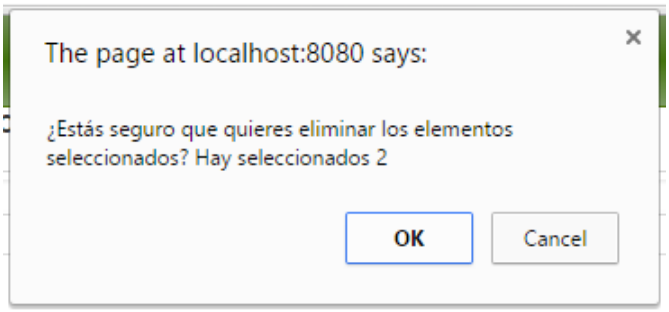


Figura 9.37 Confirmación de borrado

La opción de menú de buscar medios por tipo ofrece un formulario para seleccionar el tipo que queremos buscar, entre video o audio, y al hacer clic en buscar, muestra un listado de medios del usuario filtrado según el tipo elegido.

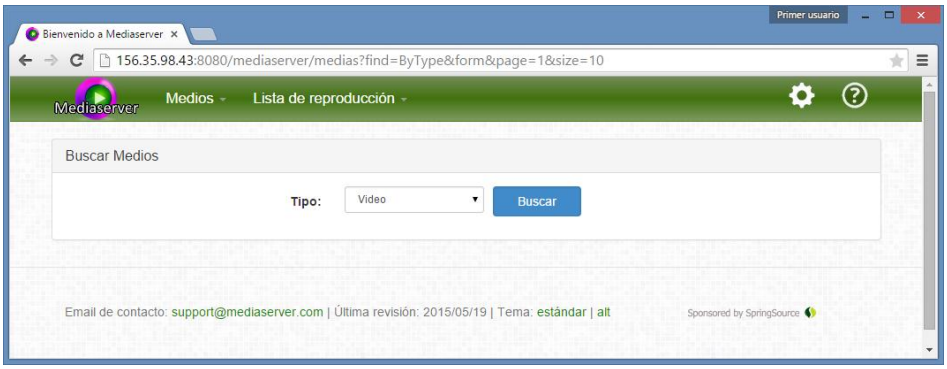


Figura 9.38 Página de búsqueda por tipo

Desde el menú principal podemos acceder a una página para crear nuevos medios.

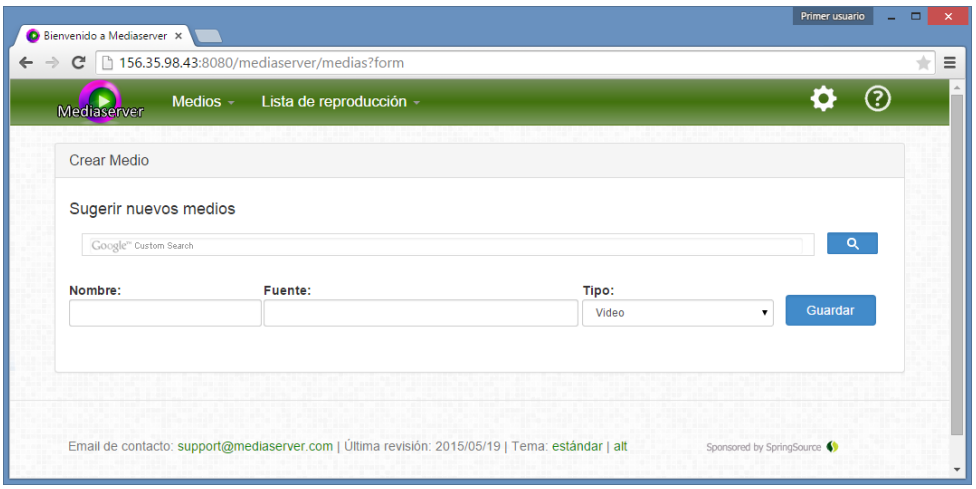


Figura 9.39 Página de creación de nuevo medio

El formulario para crear un nuevo medio accesible desde el menú es muy similar al ya explicado en el listado, la única diferencia es que tras completar la creación de este modo, se accede directamente a la reproducción del mismo.

Desde el listado también se puede acceder a la reproducción de cada medio concreto con su enlace correspondiente en la parte de la derecha.

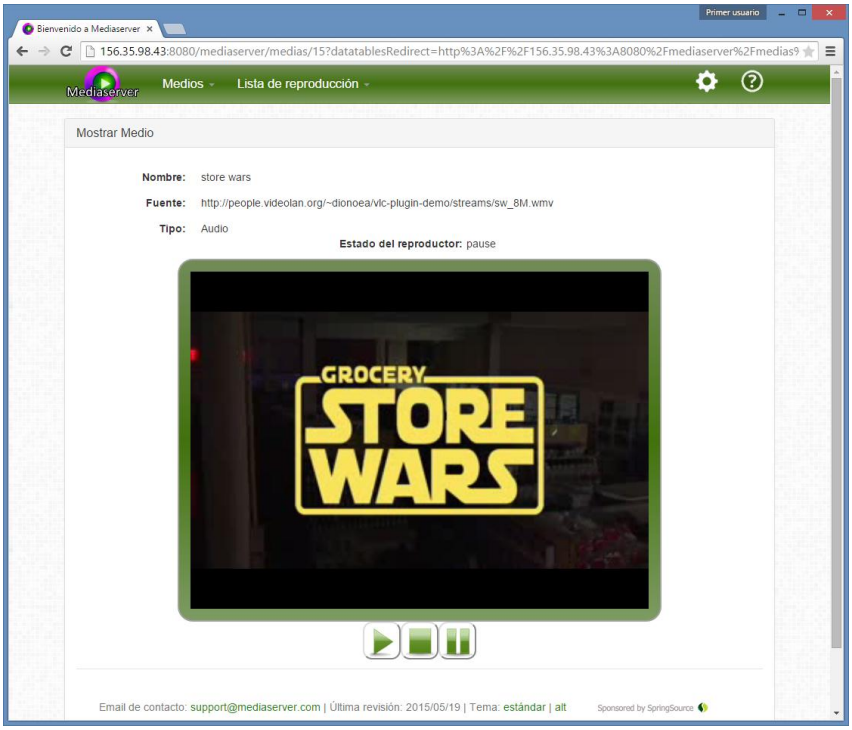


Figura 9.40 Página de ver medio

Al acceder a la visualización de un medio desde el listado, o tras haber creado uno en la página de creación accedemos a la página de reproducción. En ella además de la información del medio concreto disponemos de un reproductor con controles para visualizadlo e información sobre el estado de reproducción.

En esta página se reflejarán las órdenes recibidas del control remoto móvil. Que explicaremos más adelante.

Ahora que disponemos de medios, el siguiente punto interesante sería la posibilidad de crear listas de reproducción en las que ordenarlos y reproducirlos secuencialmente. Esto lo haremos a través de categoría de menú de listas de reproducción.

A través de la opción “Ver listas de reproducción” accederemos a la página que se muestra a continuación

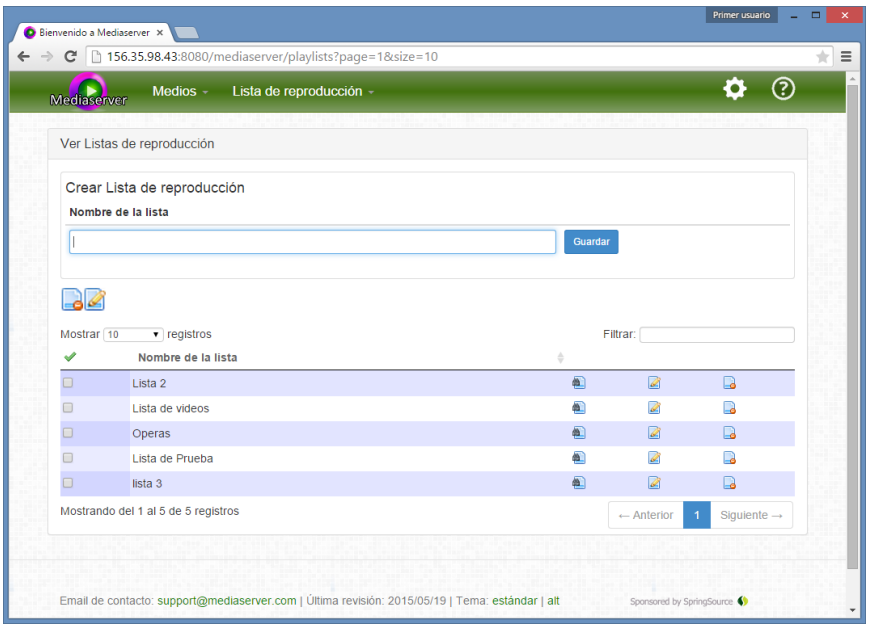


Figura 9.41 Página listado de listas de reproducción

En esta página se puede revisar y editar las listas de reproducción así como incorporar otras nuevas con solo especificar su nombre.

Además podremos acceder a la visualización de cada lista con el enlace correspondiente.

Como en el caso del listado de medios, la tabla de listas de reproducción puede filtrarse, reordenarse, y cambiar el número máximo de registros mostrados.

El manejo de esta página es muy similar al de la lista de medios.

También disponemos de una página de creación de nuevas listas de reproducción independiente del listado, y que como en el caso de su respectiva versión para la creación de medio, nos dirige tras completarla a la reproducción de la lista que acabamos de crear.

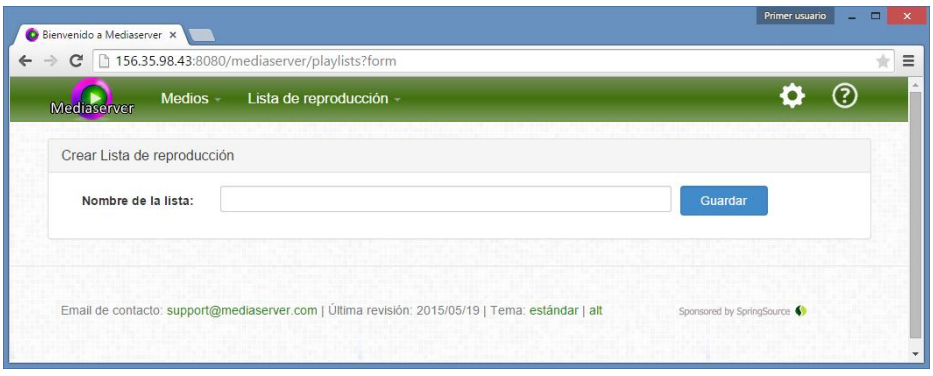


Figura 9.42 Página de crear lista de reproducción

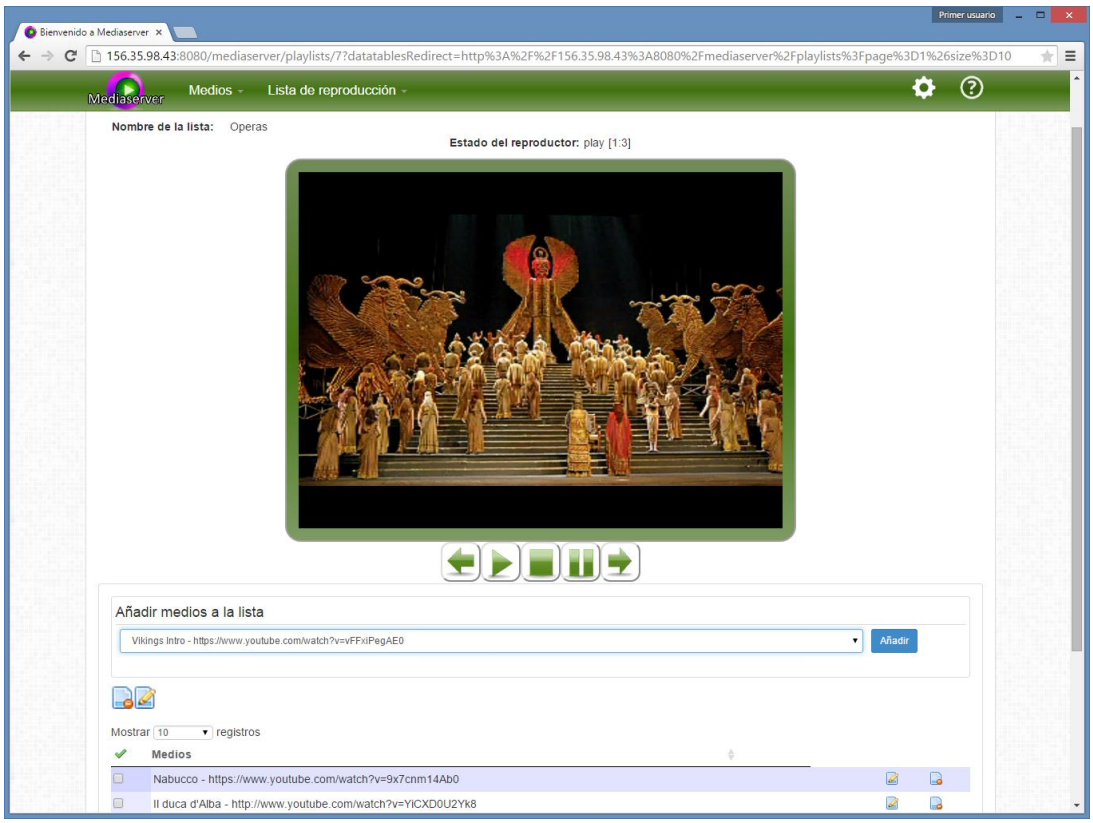


Figura 9.43 Página de ver lista de reproducción

Al acceder a la visualización de una lista de reproducción disponemos además de un reproductor y controles para la reproducción, de un formulario para añadir y editar qué medios forman parte de esa lista.

Además del estado de reproducción situado sobre el reproductor se informa del número de medio reproducido actualmente y el número total de medios de la lista.

Esta página de reproducción también es controlable desde la aplicación móvil.

Si no hemos añadido aún ningún medio a la lista de reproducción, no se mostrará el reproductor, sino solamente las herramientas para añadir nuevos medios a la misma

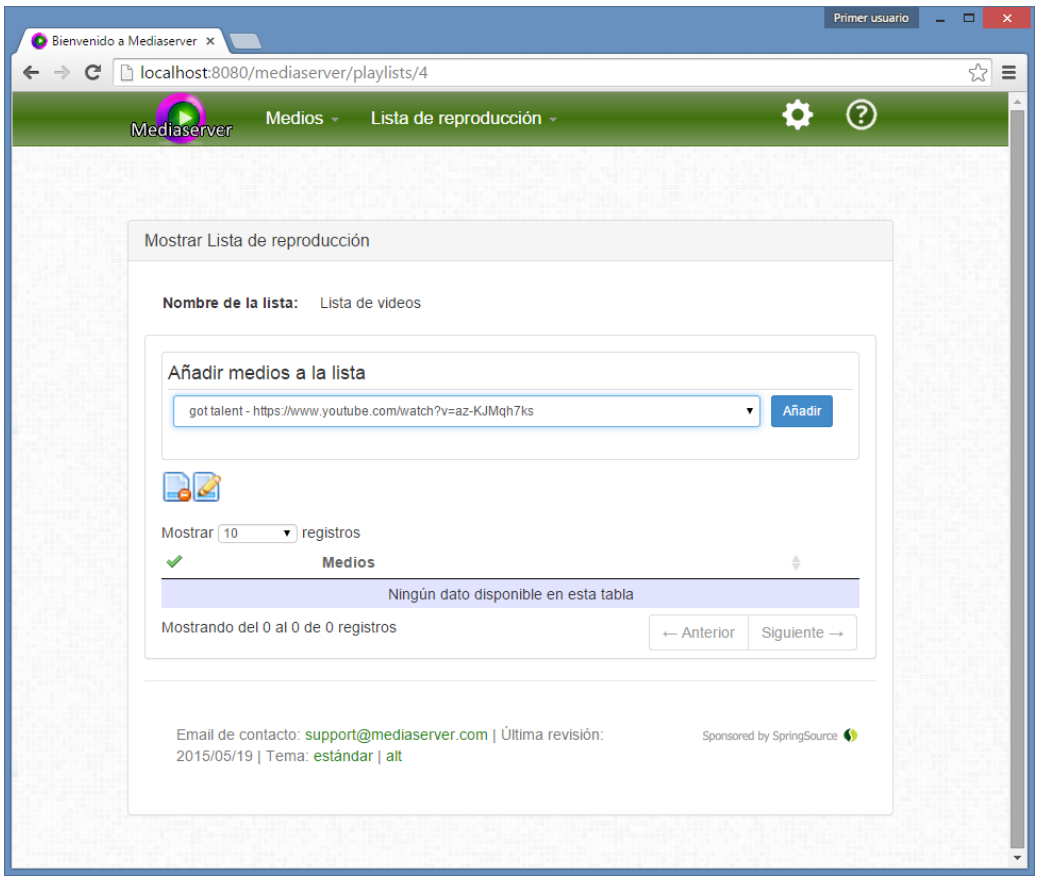


Figura 9.44 Página de ver lista de reproducción sin elementos añadidos

Para añadir nuevos elementos a la lista de preproducción, seleccionaremos uno del listado desplegable, donde aparecerán ordenados alfabéticamente por el nombre y haremos clic en el botón “Añadir”.

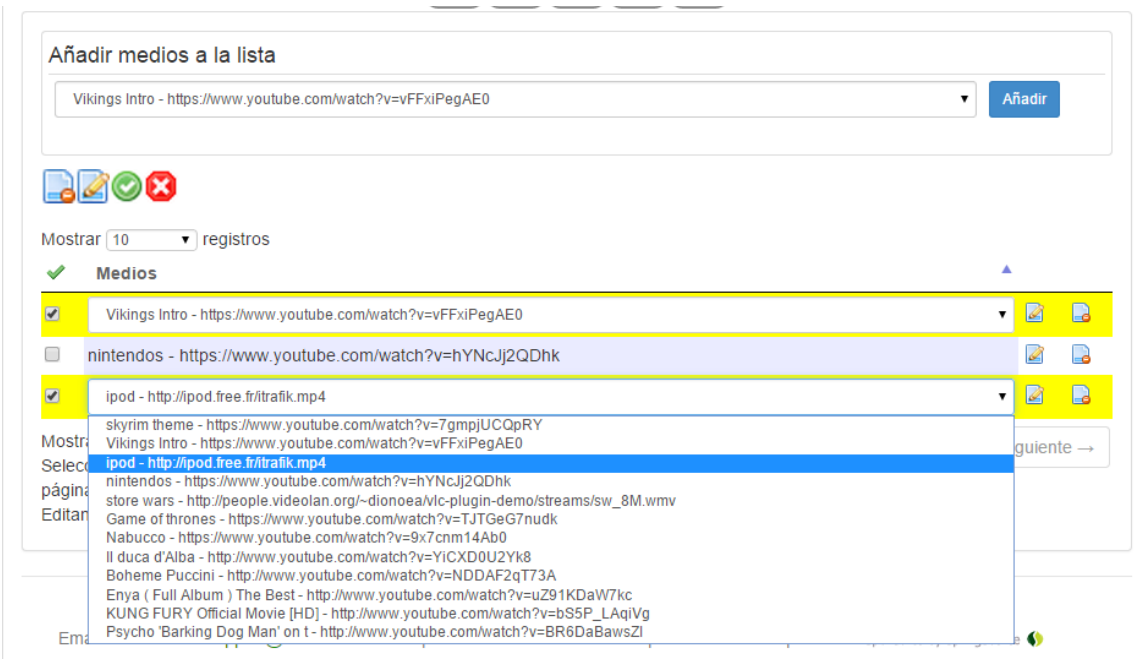


Figura 9.45 Edición de elementos de una lista de reproducción

Como en el resto de tablas, tenemos controles de edición y borrado para ajustar los datos según sea conveniente.

Para facilitar el uso de la aplicación se ofrece una opción de ayuda en el menú que da acceso a una página en la que se trata de resolver las dudas que el usuario pudiera tener para conseguir reproducir correctamente los videos en su ordenador, habilitando y desbloqueando el plugin de reproducción.

También se informa de donde conseguir la aplicación móvil de control remoto.

Figura 9.46 Página de ayuda

Si queremos cambiar la contraseña accederemos a página de cambio de contraseña a través del menú de configuración, y rellenaremos el siguiente formulario en el que se le solicita la contraseña actual y por duplicado la nueva.

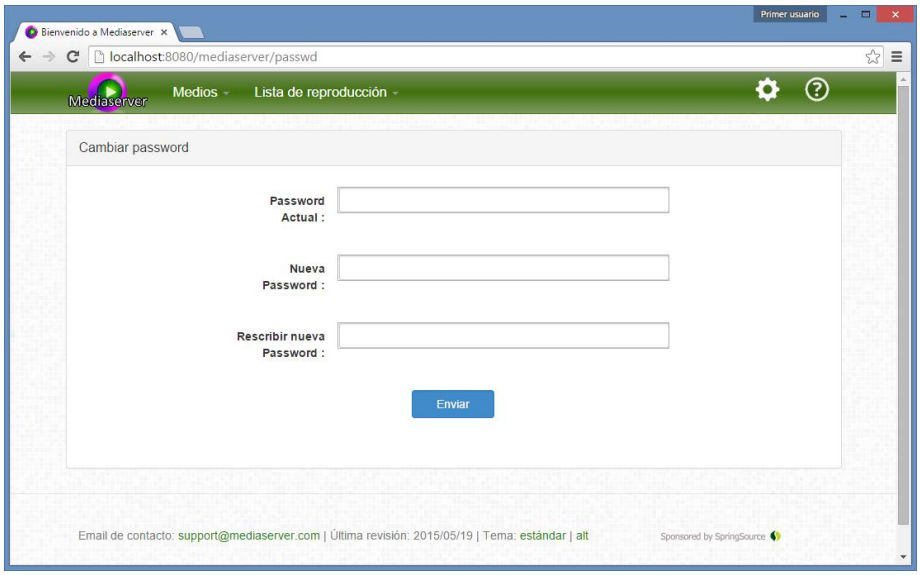


Figura 9.47 Página de cambio de contraseña

9.3.1.2 Parte Móvil

Ahora que sabemos cómo crear y editar medios y lista de reproducción, veamos cómo controlar la reproducción de las mismas a distancia.

Al acceder al sitio de la aplicación **con el navegador** de un dispositivo móvil, como por ejemplo un teléfono, se le recuerda al usuario la posibilidad de obtener la aplicación Android de control remoto simplemente haciendo clic en el llamativo cartel.

Figura 9.48 Página principal de la aplicación vista en un navegador móvil (pantalla estrecha)

Una vez instalada la aplicación móvil, podemos acceder a ella.

La interfaz de control remoto nos permitirá controlar la reproducción que se lleva a cabo en el navegador web, para ello hemos de tener **abierta la página de visualización de un medio o lista de reproducción**, y configurar correctamente la aplicación móvil para conectarse al servidor utilizando las mimas credenciales con las que hemos iniciado sesión en el cliente web.

Figura 9.49 Pantalla principal de la aplicación de control remoto

La pantalla principal de la aplicación de control remoto contiene controles para enviar órdenes de control de reproducción al servidor para:

- Iniciar la reproducción
- Pausar la reproducción
- Parar la reproducción
- Pasar al medio anterior y posterior
- Cambiar el volumen
- Cambiar el estado de progreso
- Pasar a pantalla completa o volver a pantalla parcial

Además disponemos de un campo en el que se muestra el último mensaje transmitido correctamente por el servidor. En caso de que se produzca algún error también se mostrará ahí.

La conexión a internet es imprescindible para enviar cualquier orden de control.

Desde esta pantalla se puede acceder la configuración de conexión a través del icono correspondiente en la esquina superior derecha de la barra de menú.

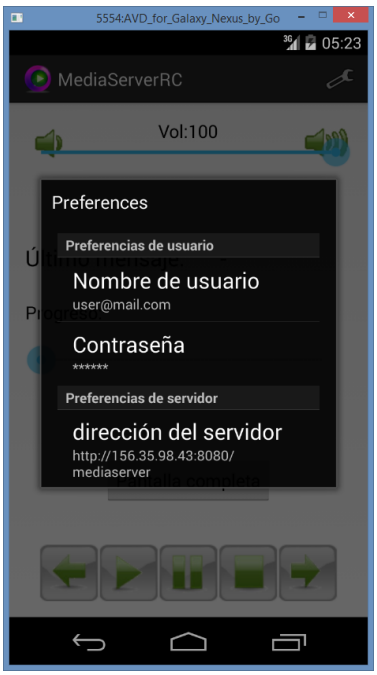


Figura 9.50 Pantalla de configuración de la aplicación de control remoto

En esta pantalla configuraremos los parámetros de conexión al servidor estableciendo las credenciales de usuario (email y contraseña) y la dirección del servidor al que conectarnos

9.3.2 Usuario administrador (web)

Al iniciar sesión como administrador se accede a la página de listado de usuarios, y el menú se amplía con todas las funciones de administración de usuarios, roles y asociaciones de medios y listas, además de las ya presentes como usuario standard.

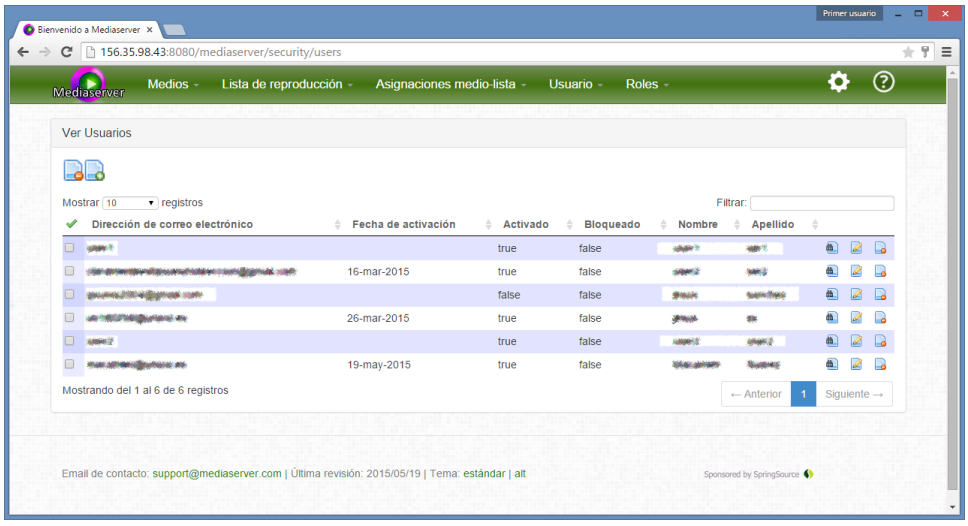


Figura 9.51 Página de listado de usuarios

En la página de listado de usuarios se puede ver la información relativa a todos los usuarios de la aplicación, comprobar su estado de bloqueo y fecha de activación.

También se puede acceder a ver uno de los usuarios en detalle, editarlo o borrarlo. En este caso como el número de campos es amplio que para los medios o listas de reproducción, la edición o creación se lleva siempre a cabo en una página independiente, aunque si que se puede realizar borrado múltiple como se vio en casos anteriores.

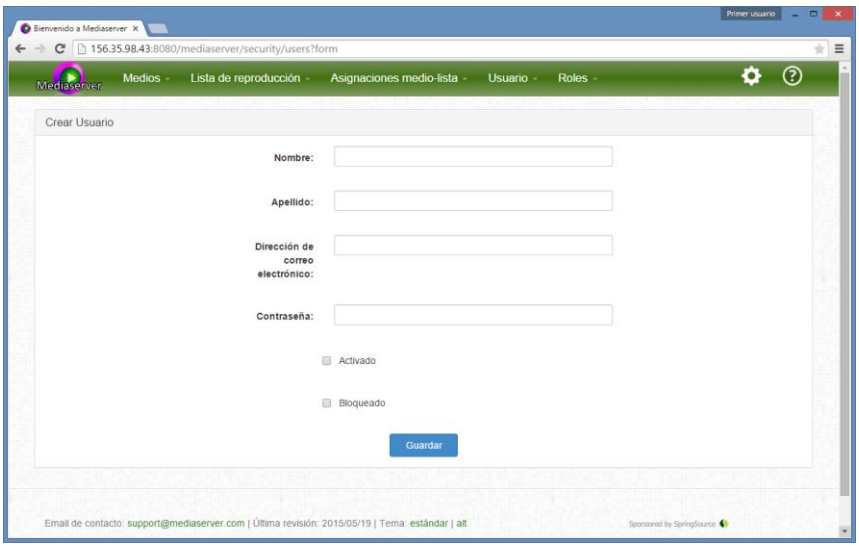


Figura 9.52 Página creación de usuario

Los usuarios pueden ser creados y activados manualmente por un administrador mediante el formulario de la figura precedente, aparte del registro de usuarios mediante el formulario de registro.

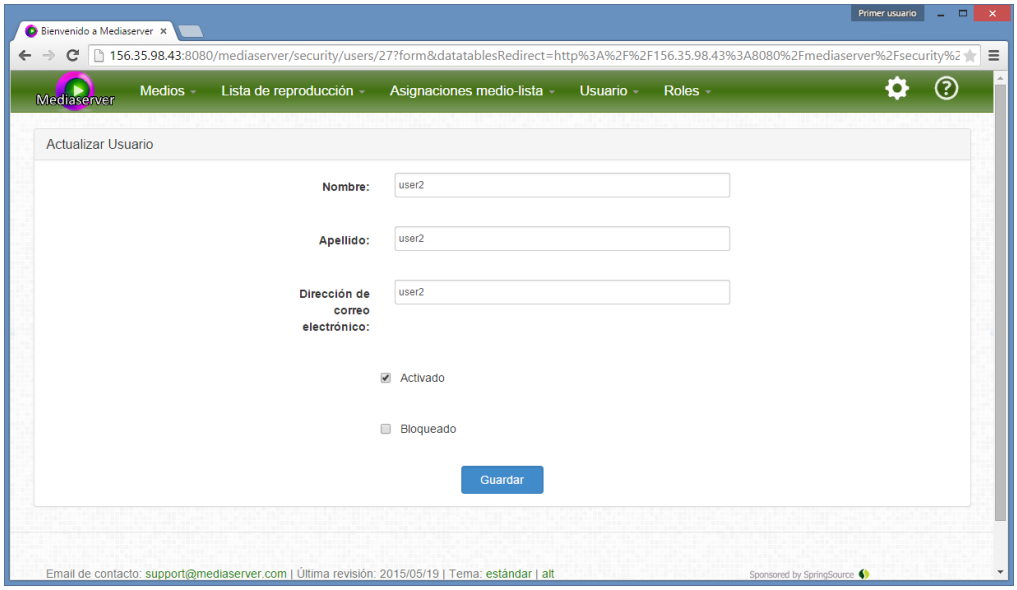


Figura 9.53 Página edición de usuario

Al acceder a la edición de un usuario se muestra un formulario con sus datos para poder actualizarlos.

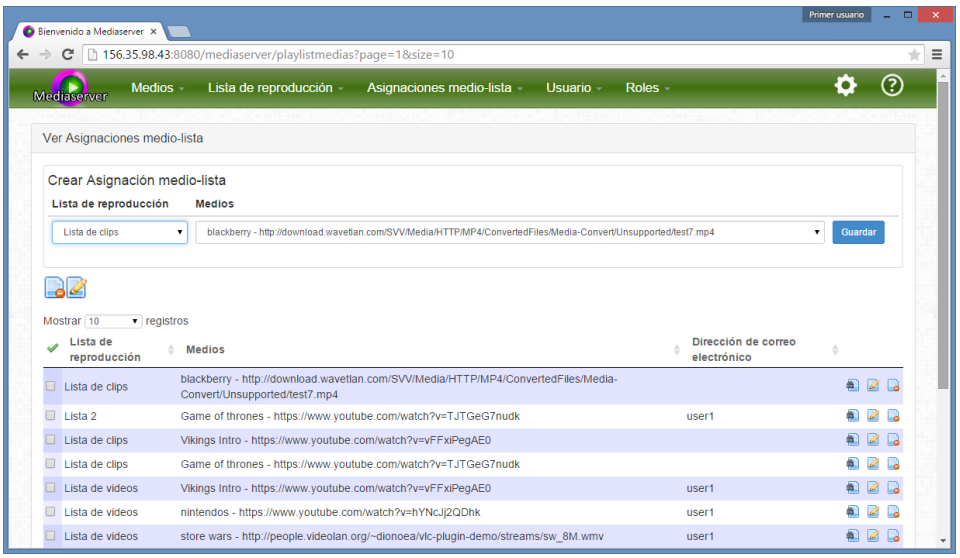


Figura 9.54 Página de listado de asignaciones medio-lista

A través de la opción de menú para listas las asignaciones medio-lista, el administrador puede ver qué medios pertenecen a qué listas de reproducción y a qué usuarios, ajustando o burrando estas asignaciones si es necesario.

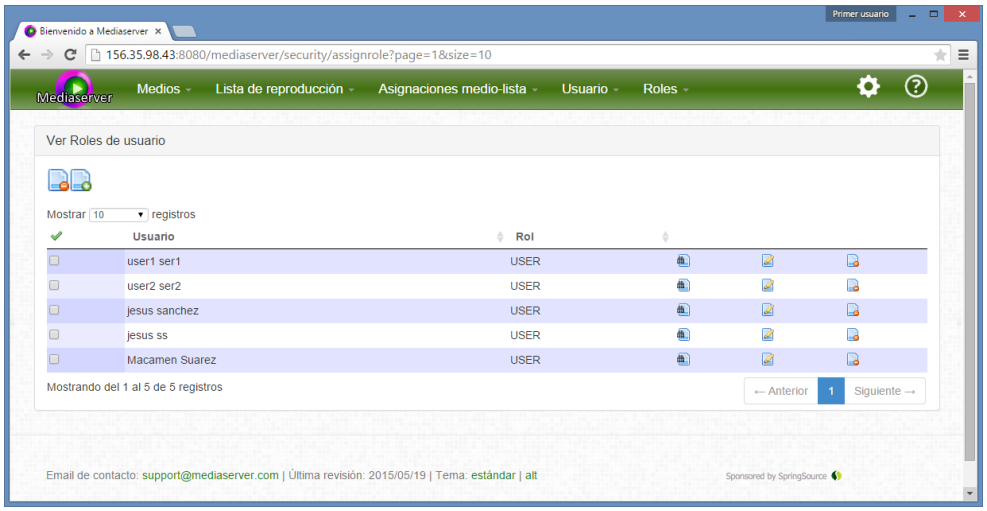


Figura 9.55 Página de listado de roles

En la página de listado de roles ell administrador podrá comprobar qué rol está asociado a cada usuario, acceder a su modificación o eliminar un rol de uno de ellos si es necesario.

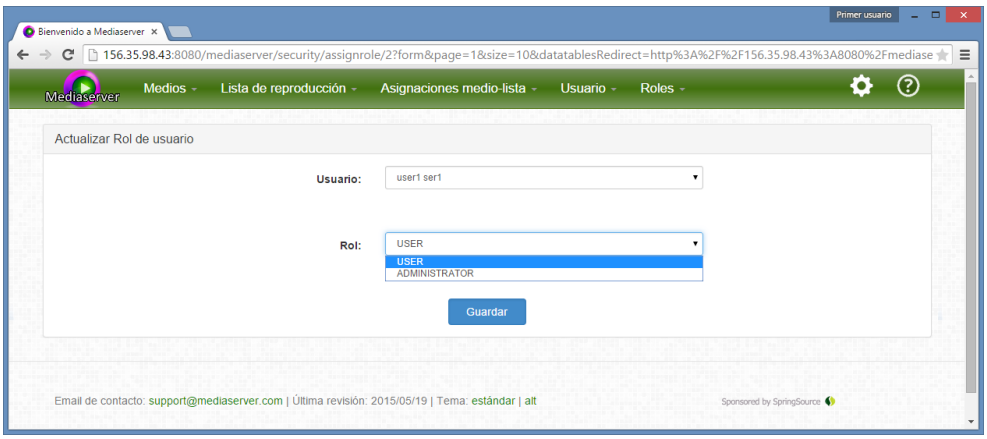


Figura 9.56 Página de editar rol de usuario

En la página de edición de rol el administrador podrá editar el rol de un usuario determinado.

A diferencia de un usuario standard, un administrador podrá ver todos los medios y listas de reproducción, no solo las de su propiedad.

9.4 Manual del Programador

En esta sección describiremos las consideraciones que han de tener los futuros desarrolladores a la hora de introducir mejoras de rendimiento o ampliar la funcionalidad de la aplicación.

9.4.1 Cambio de detalles de la base de datos

Para cambiar la base de datos de la aplicación es necesario modificar el archivo database.properties situado en el directorio src/main/resources/META-INT/spring y cuyo contenido sería similar al siguiente:

```
#Updated at Mon Sep 01 10:01:54 GMT+02:00 2014
#Mon Sep 01 10:01:54 GMT+02:00 2014
database.driverClassName=com.mysql.jdbc.Driver
database.url=jdbc\:mysql\://localhost\:3306/mediaserver
#database.url=jdbc\:mysql\://198.11.212.125\:3306/mediaserver
database.username=local
database.password=local
```

Para cambiar la base de datos habría que cambiar la referencia a la clase controladora, incluirla en las dependencias del proyecto, etc.

9.4.2 Cambio del proveedor de correo

Para modificar el proveedor de correo utilizado para enviar emails a los uusrios en necesario modificar el archivo email.properties situado en el directorio src/main/resources/META-INT/spring y cuyo contenido sería similar al siguiente:

```
#Updated at Mon Jul 21 09:40:49 GMT+02:00 2014
#Mon Jul 21 09:40:49 GMT+02:00 2014
email.from=mediaserver@mediaserver.com
email.host=smtp.mandrillapp.com
email.password=password
email.port=587
email.protocol=smtp
email.username=mediaserver@mediaserver.com
```

9.4.3 Cambio del plugin de reproducción

Actualmente, la reproducción de contenidos multimedia del proyecto depende del plugin web NPAPI VLC, que presenta ciertos problemas de compatibilidad y fiabilidad.

Para sustituir este plugin sería necesario modificar 3 archivos en los que se hace referencia a los elementos de VLC:

- mediaserver\src\main\webapp\WEB-INF\views\medias\show.jspx
- mediaserver\src\main\webapp\WEB-INF\views\playlists\show.jspx
- mediaserver\src\main\webapp\scripts\websockets\videoSocket.js

Los dos primeros se corresponden a las páginas de reproducción de medios y listas respectivamente y en ellos se hace referencia al objeto reproductor para insertarlo en la página web.

El último archivo corresponde al fichero javascript encargado de recibir las órdenes de la aplicación e interpretarlas, así como de enviar al servidor los eventos detectados.

Si fuera necesario incluir nuevos ficheros de script en la aplicación, que se utilicen en numerosas páginas del sitio, como puede ser en este caso, scripts de reproducción, se puede generalizar su inclusión añadiéndolos en el fichero

mediaserver\src\main\webapp\WEB-INF\tags\util\load-scripts.tagx

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Habiendo completado el desarrollo del proyecto se pueden extraer una serie de conclusiones que se expondrán a continuación.

Se han cumplido los objetivos planteados al comenzar el proyecto. Aunque esta primera versión sea susceptible de numerosas mejoras y ampliaciones, permite de manera correcta controlar remotamente la reproducción de contenidos multimedia en un reproductor web de manera remota desde un dispositivo móvil sin necesidad de emparejamiento previo de dispositivos, sincronizando incluso múltiples reproductores asociados. También se ha conseguido incorporar una completa gestión de usuarios, así como de medios y listas de reproducción asociados a ellos.

Para comparar las ventajas de la aplicación desarrollada con las de las alternativas estudiadas inicialmente, se ha agregado a la tabla inicial una columna con la nueva aplicación, como vemos a continuación.

Característica	Youtube TV	VLC Remote	Pogoplug PC	Microsoft Remote Desktop	Mediaserver
Fácil manejo	✓	✓	✓	✗	✓
Gratuito	✓	✓	✗	✓	✓
Gran cantidad de formatos	✗	✓	✓	✓	✓
Multitud de orígenes de archivos	✗	✓	✗	✓	✓
Reproducción remota	✓	✓	✗	✓	✓
Sistema web	✓	✗	✓	✗	✓
Recursos y listas disponibles globalmente	✓	✗	✓	✗	✓
Control y sincronización de varios reproductores	✗	✗	✗	✗	✓
Libre de proceso de emparejamiento	✗	✗	✓	✗	✓

De la tabla anterior podemos destacar:

- Mediaserver es una aplicación sencilla de manejar al igual que la mayoría de las anteriores.
- Es un sistema gratuito, lo cual hace que sea asequible a todos los públicos.
- Admite una gran cantidad de formatos y orígenes de archivos por lo que no está limitada en ese sentido y los recursos a reproducir son enormemente abundantes.

- Permite controlar la reproducción de manera remota sin tener que acercarse al equipo que actúa de reproductor y sin límite de distancia siempre que dispositivo móvil y reproductor estén conectados a internet.
- Es un sistema web por lo que está disponible en cualquier lugar sin necesidad de instalación, salvando la necesidad del plugin de reproducción, que actualmente está muy extendido dada la popularidad del reproductor multimedia VLC.
- También como consecuencia de ser un sistema web los recursos están siempre disponibles desde cualquier ubicación sin que el usuario tenga que preocuparse de llevarlos consigo.
- Al contrario que las alternativas estudiadas, Mediaserver permite controlar y sincronizar múltiples reproductores desde un solo dispositivo de control, dando opción a manejar la reproducción de contenidos multimedia de reproductores situados distantes geográficamente.
- La asociación entre máquina o máquinas reproductoras y dispositivos de control se realiza mediante las credenciales de cuenta, por lo que no es necesario emparejar ningún emparejamiento adicional entre reproductor y control remoto.

Una ventaja adicional de la aplicación que no se recoge en la tabla es la supresión de publicidad en algunos videos, que un su servidor original el usuario estaría obligado a ver antes de visualizar el vídeo en cuestión.

La elección del lenguaje y herramientas de desarrollo han cumplido las expectativas y posibilitado la consecución de los objetivos, no obstante, en el caso concreto de vg-Nix /Spring Roo da la sensación de que en ocasiones ha entorpecido el proceso de desarrollo y limitado la libertad de acción.

Si bien el comienzo del desarrollo con la herramienta de desarrollo rápido consigue pasar a disponer de manera muy rápida de una aplicación con las entidades de dominio conectada a una base de datos y una interfaz web para manejarlas, al tratar de variar mucho el esquema de funcionamiento y adaptarlo a las necesidades específicas, la herramienta y el framework pueden realizar modificaciones automáticas no deseadas que dificulten el trabajo por lo que para otros proyectos tal vez se considere la opción de utilizarla de inicio y luego desactivarla antes de que cause problemas.

Las tecnologías cambian rápidamente incluso durante el transcurso del proceso de desarrollo de un proyecto, apareciendo opciones mejores que las que se habían considerado inicialmente, o desapareciendo otras que se tenían por las más viables por lo que el desarrollador siempre ha de estar abierto a nuevas posibilidades, cambios modificaciones y ajustes para conseguir un mejor resultado final.

El empezar a documentar en fases tempranas del proyecto, facilita notablemente la posterior recopilación de la información y la consistencia y fidelidad de la misma.

Los servicios gratuitos de computación en la nube dan amplias posibilidades de experimentación, pero suponen riesgos adicionales el confiar en su disponibilidad y persistencia, ya que las máquinas virtuales suministradas pueden estar fuera de servicio

cuando más se necesitan o incluso ser reseteadas por lo que siempre es necesario tener un respaldo de sus funciones y datos en otro lugar.

El desarrollo en un entorno de empresa da la posibilita un frecuente contraste de opiniones y sugerencias respecto al desarrollo así como consulta de dudas, lo cual hace que el desarrollo sea más fácil y ameno, y el producto final sea de mejor calidad.

10.2 Ampliaciones

En este apartado se presentan las ampliaciones que han sido consideradas durante el desarrollo del proyecto, y que podrían ampliar y completar su funcionalidad pero que en esta versión primera del proyecto no han sido llevadas a cabo por riesgo de ampliar excesivamente las dimensiones del mismo dificultando llevarlo término y necesidad de completarlo en un plazo determinado.

1. En primer lugar como ya se ha mencionado en varios puntos, un cambio necesario a no mucho tardar sería la sustitución del plugin de reproducción por otro no basado en NPAPI, lo cual supondría mantener la compatibilidad con navegadores como Chrome que vayan a bloquear el plugin actual. También podría esto solucionar o mejorar problemas de fiabilidad y cuelgues ocasionales que se producen actualmente.

2. Otra ampliación interesante sería ampliar las funciones de la aplicación móvil, pasando de un mero control remoto a ofrecer funciones más completas de edición de los medios y listas de reproducción, poder incluir nuevos medios en las listas o cambiar el orden de los mismos.

3. La inclusión en el sistema de una herramienta para subida de archivos podría facilitar el que los usuarios pudieran disponer de sus propios contenidos multimedia. Aunque ahora mismo pueden hacerlo, han de almacenar estos archivos en servicios de almacenamiento como Dropbox, o google drive y ponerlos disponibles públicamente.

4. Android es una plataforma muy extendida en los dispositivos móviles pero no es la única, por lo que otras versiones para Apple o Windows pone ayudarían a disponer de un mercado más amplio para el sistema Mediaserver.

5. El desarrollo de un sistema de comunicación y sincronización entre servidores facilitaría la escalabilidad de la aplicación pudiendo desplegar multitud de servidores balanceados que atendieran simultáneamente peticiones de una cantidad de usuarios mucho mayor.

Capítulo 11. Presupuesto

En esta sección se hace un cálculo de presupuesto en el caso de que el cliente fuera un cliente real y no se tratara de una aplicación hecha para el ámbito académico.

El presupuesto engloba todo el proceso de desarrollo del proyecto y sus distintas partes, atendiendo a los costes temporales y el grado de esfuerzo requerido para llevarlas a cabo.

No se ha tenido en cuenta el coste de licencias de software ni bibliotecas pues todo el software base utilizado es de uso gratuito y no ha sido preciso contratar mano de obra extra.

Concepto	Cantidad (Horas)	Precio Unitario	Coste Total Concepto
Inicio del proyecto	-	500,00 €	500,00 €
Estudio de viabilidad	30,00	51,67 €	1.550,00 €
Análisis del sistema	25,00	55,00 €	1.375,00 €
Formación	50,00	35,00 €	1.750,00 €
Diseño del sistema	65,00	36,92 €	2.400,00 €
Implementación del sistema	100,00	17,60 €	1.760,00 €
Pruebas	105,00	20,45 €	2.147,00 €
Corrección de errores	40,00	17,50 €	700,00 €
Introducción de mejoras	45,00	32,00 €	1.440,00 €
Generación de documentación y manuales	65,00	20,00 €	1.300,00 €
Preparación, presentación e implantación	25,00	15,00 €	375,00 €
Reuniones de dirección	5,00	80,00 €	400,00 €
		Subtotal	15.697,00 €
		IVA (21%)	3.296,37 €
		TOTAL	**18.993,37 €**

Capítulo 12. Referencias Bibliográficas

12.1 Libros y Artículos

A continuación se presenta una recopilación de los principales libros y artículos consultados durante el desarrollo del proyecto o su documentación.

[Gironés14] Gironés, Jesús Tomás; Carbonell, Vicente; Vogt, Carsten; García Pineda, Miguel; Bataller Mscarell, Jordi; Ferri, Daniel. "El gran libro de Android avanzado". Marcombo. 2014. ISBN 978-84-267-2078-8

[Konda11] Konda, Madhusudhan. "Just Spring". O'Really. 2011. ISBN 978-1-449-31146-9

[Long11] Long, Josh; Mayzak, Steve. "Getting Started with Roo". O'Really. 2011. ISBN 978-1-449-30790-5

[Reid11] Reid, Jon. "jQeury Mobile" ". O'Really. E.E.U.U. 2011. ISBN: 978-1-449-30668-7

[Rimple12] Rimple, ken; Penchikala, Srini. "Spring Roo in Action". Manning. Shelter Island 2012 ISBN 978-19-351-8296-2

[Wargo11] Wargo, John M. "Apache Cordova 3 Programming". Addison-Wesley. 2013. ISBN-10: 0-321-95736-9 ISBN-13: 978-0-321-95736-8

12.2 Referencias en Internet

A continuación se presenta una recopilación de las principales páginas Web consultados durante el desarrollo del proyecto y documentación.

Accesibilidad herramienta de análisis.
http://www.sidar.org/hera/ 2015

Búsquedas personalizadas con el motor de búsqueda de Google.
https://cse.google.es/cse/all 2015

Creación de diagramas Entidad-relación.
https://www.gliffy.com 2015

Diagramas de actividad.
http://www.agilemodeling.com/artifacts/activityDiagram.htm 2015

Etiquetas de idioma en HTML.
http://www.w3.org/International/articles/language-tags/Overview.es.php 2015

Foro de programación.
http://stackoverflow.com/ 2015

gvNIX, herramienta de desarrollo rápido de aplicaciones Java.
http://www.gvnix.org 2015

Gv-Nix tutorial, de creación de proyectos.
https://github.com/DISID/gvnix-samples/tree/master/quickstart-app 2015

Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web".
http://www.nosolousabilidad.com/articulos/heuristica.htm 2015

HTML, javascript, CSS, utilización
http://www.w3schools.com/ 20155

JQuery, utilización
https://jquery.com/ 2015

Plugin de reproducción Web Chimera.
http://wiki.webchimera.org/ 2015

Servicio externo de correo para el envío de emails de registro y recuperación de password.
http://mandrill.com/ 2015

Herramienta de prueba de accesibilidad.
http://www.tawdis.net/ 2015

Test de rendimiento de carga de páginas web.
http://www.webpagetest.org/ 2015

Tutorial de UML.

http://www.sparxsystems.com.au/uml-tutorial.htmll 2015

Validación de CSS.
http://jigsaw.w3.org/css-validator/ 2015

VLC Documentación del plugin web.
https://wiki.videolan.org/Documentation:WebPlugin/ 2015

WebSockets, introducción.
http://www.arkaitzgarro.com/html5/capitulo-13.html 2015

Wikipedia.
http://es.wikipedia.org/wiki/Wikipedia:Portada 2015

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

Por orden alfabético, todos los términos considerados importantes en la aplicación con una descripción breve de su significado dentro de la aplicación.

- **Bases de datos**: Conjunto de información almacenada en memoria auxiliar que permite el acceso directo y un conjunto de programas que manipulan esos datos.

- **Contenido multimedia:** Medios utilizados para la comunicación que combinan texto, fotografías, imágenes de vídeo o sonido.

- **Cookie:** Es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador para ser luego recuperada por el servidor en posteriores visitas. En la aplicación este es el medio de guardar las sesiones de los usuarios así como otros datos como sus preferencias de idioma o tema.

- **Lista de reproducción:** Colección de medios ordenados para su reproducción secuencia.

- **Medio:** Cualquiera de los recursos multimedia de la aplicación, identificado por un nombre y disponible en una dirección o fuente. Se asocian al usuario que los introduce en el sistema.

- **Plugin de reproducción web:** Complemento para el navegador que permite reproducir archivos multimedia dentro de una página web. En el proyecto se utiliza el plugin web del reproductor multimedia VLC.

- **Usuario anónimo:** Es un usuario que todavía no se ha identificado mediante el proceso de login, y que solo puede acceder a la parte pública de la aplicación.

- **Usuario administrador:** Usuario que ha superado el proceso de login y que dispone del rol de administrador de la aplicación, lo que le permite acceder además de las opciones de un usuario standard a la gestión de usuarios roles y revisión de todos los recursos de los mismos.

- **Usuario móvil:** Usuario que interactúa con el sistema a distancia haciendo uso de su dispositivo móvil.

- **Usuario registrado:** Usuario standard de la aplicación, ha superado correctamente el proceso de login y puede acceder a las funciones de gestión y visualización de medios y listas de reproducción.

- **WebSocket** Es una tecnología que proporciona un canal de comunicación bidireccional y full-duplex sobre un único socket TCP. En el proyecto se utiliza para comunicar a los reproductores de los clientes web las órdenes de control emitidas por los clientes móviles de control remoto.

13.2 Contenidos Entregados

13.2.1 Contenidos

En la siguiente tabla se muestran los directorios de los que constan los anexos entregados adjuntos a este documento, explicando el contenido de cada directorio.

Primer anexo

Directorio	Contenido
./	Contiene un fichero leeme.txt explicando toda esta estructura.
./Mediaserver	Contiene toda la estructura de directorios de la parte web del proyecto.
./MediaserverRC	Contiene toda la estructura de directorios de la parte Android del proyecto.
./Mediaserver/doc	Contiene la la documentación javadoc generada relativa al proyecto web.
./Mediaserver/src	Contiene los ficheros fuetne del proyecto web.
./MediaserverRC/doc	Contiene la la documentación javadoc generada relativa al proyecto Android.
./MediaserverRC/src	Contiene los ficheros fuetne del proyecto Android.
MediaServerRC/bin	Contiene la aplicación MediaserverRC.apk

Anexos segundo y tercero.

Directorio	Contenido
./	Contiene la aplicación servidora desplegable mediaserver.war

13.2.2 Código Ejecutable e Instalación

La aplicación web requiere Java 1.7 instalado en el sistema, una base de datos MySQL configurada tal como se describe en el manual de instalación y un servidor de aplicaciones java como Tomcat 7.0.50.

Para iniciar la aplicación web solo es necesario depositar el archivo war de la aplicación en el directorio de despliegue e iniciar el servidor de aplicaciones web.

La aplicación móvil no precisa más que de ser descargada e instalada en el dispositivo para funcionar, además de ser configurada para comunicarse con el servidor con el fin de transmitir las órdenes.

13.2.3 Ficheros de Configuración

La aplicación web viene pre configurada para funcionar con una base de datos local y un servicio de correo externo, si se desean cambiar estas configuraciones, se pueden seguir las instrucciones detalladas en los manuales de instalación y programador.

13.3 Índice Alfabético

T

U

V

W

X

13.4 Código Fuente de la aplicación Web

En esta sección se incluye el código fuente de la aplicación. Se han seleccionado las clases más relevantes dejando algunas triviales, o importadas.

Las clases mostradas se organizan según los paquetes a los que pertenecen.

13.4.1 Paquete com.mediaserver.domain:

13.4.1.1 Fichero "Media.java":

```
package com.mediaserver.domain;
import javax.persistence.Enumerated;
import javax.persistence.ManyToOne;
import javax.validation.constraints.Size;

import org.gvnix.addon.jpa.query.GvNIXJpaQuery;
import org.springframework.roo.addon.javabean.RooJavaBean;
import org.springframework.roo.addon.jpa.activerecord.RooJpaActiveRecord;
import org.springframework.roo.addon.tostring.RooToString;

import com.mediaserver.domain.security.User;
import com.mediaserver.reference.MediaType;

/**
 * Esta clase representará cada uno de los archivos multimedia de la aplicación, compuestos por
un nombre y una ubicacion
 * @author UO180258
 *
 */
@RooJavaBean
@RooToString
@RooJpaActiveRecord(finders = { "findMediasByType","findMediasByOwner" })
public class Media {

    /**
     * Nombre del medio
     */
    @Size(min = 3, max = 30)
    private String name;

    /**
     * dirección en la que se ubica el medio
     */
    private String source;

    /**
     * Tipo de medio
     */
    @Enumerated
    private MediaType type;

    /**
     * Propietario o creador del medio
     */
    @ManyToOne
    @GvNIXJpaQuery(filterBy={"email_address"})
    private User owner;
}
```

13.4.1.2 Fichero "Media_Roo_configurable.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import org.springframework.beans.factory.annotation.Configurable;

privileged aspect Media_Roo_Configurable {

    declare @type: Media: @Configurable;

}
```

13.4.1.3 Fichero "Media_Roo_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.security.User;
import com.mediaserver.reference.MediaType;
import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

privileged aspect Media_Roo_Finder {

    public static Long Media.countFindMediasByOwner(User owner) {
        if (owner == null) throw new IllegalArgumentException("The owner argument is required");
        EntityManager em = Media.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM Media AS o WHERE o.owner = :owner",
Long.class);
        q.setParameter("owner", owner);
        return ((Long) q.getSingleResult());
    }

    public static Long Media.countFindMediasByType(MediaType type) {
        if (type == null) throw new IllegalArgumentException("The type argument is required");
        EntityManager em = Media.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM Media AS o WHERE o.type = :type",
Long.class);
        q.setParameter("type", type);
        return ((Long) q.getSingleResult());
    }

    public static TypedQuery<Media> Media.findMediasByOwner(User owner) {
        if (owner == null) throw new IllegalArgumentException("The owner argument is required");
        EntityManager em = Media.entityManager();
        TypedQuery<Media> q = em.createQuery("SELECT o FROM Media AS o WHERE o.owner = :owner",
Media.class);
        q.setParameter("owner", owner);
        return q;
    }

    public static TypedQuery<Media> Media.findMediasByOwner(User owner, String sortFieldName,
String sortOrder) {
        if (owner == null) throw new IllegalArgumentException("The owner argument is required");
        EntityManager em = Media.entityManager();
        StringBuilder queryBuilder = new StringBuilder("SELECT o FROM Media AS o WHERE o.owner =
:owner");
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            queryBuilder.append(" ORDER BY ").append(sortFieldName);
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                queryBuilder.append(" ").append(sortOrder);
            }
        }
        TypedQuery<Media> q = em.createQuery(queryBuilder.toString(), Media.class);
```

```
        q.setParameter("owner", owner);
        return q;
    }

    public static TypedQuery<Media> Media.findMediasByType(MediaType type) {
        if (type == null) throw new IllegalArgumentException("The type argument is required");
        EntityManager em = Media.entityManager();
        TypedQuery<Media> q = em.createQuery("SELECT o FROM Media AS o WHERE o.type = :type",
Media.class);
        q.setParameter("type", type);
        return q;
    }

    public static TypedQuery<Media> Media.findMediasByType(MediaType type, String sortFieldName,
String sortOrder) {
        if (type == null) throw new IllegalArgumentException("The type argument is required");
        EntityManager em = Media.entityManager();
        StringBuilder queryBuilder = new StringBuilder("SELECT o FROM Media AS o WHERE o.type =
:type");
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            queryBuilder.append(" ORDER BY ").append(sortFieldName);
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                queryBuilder.append(" ").append(sortOrder);
            }
        }
        TypedQuery<Media> q = em.createQuery(queryBuilder.toString(), Media.class);
        q.setParameter("type", type);
        return q;
    }

}
```

13.4.1.4 Fichero "Media_Roo_JavaBean.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.security.User;
import com.mediaserver.reference.MediaType;

privileged aspect Media_Roo_JavaBean {

    public String Media.getName() {
        return this.name;
    }

    public void Media.setName(String name) {
        this.name = name;
    }

    public String Media.getSource() {
        return this.source;
    }

    public void Media.setSource(String source) {
        this.source = source;
    }

    public MediaType Media.getType() {
        return this.type;
    }

    public void Media.setType(MediaType type) {
        this.type = type;
    }

    public User Media.getOwner() {
        return this.owner;
    }
```

```
    public void Media.setOwner(User owner) {
        this.owner = owner;
    }

}
```

13.4.1.5 Fichero "Media_Roo_Jpa_activeRecord":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.transaction.annotation.Transactional;

privileged aspect Media_Roo_Jpa_ActiveRecord {

    @PersistenceContext
    transient EntityManager Media.entityManager;

    public static final List<String> Media.fieldNames4OrderClauseFilter =
java.util.Arrays.asList("name", "source", "type");

    public static final EntityManager Media.entityManager() {
        EntityManager em = new Media().entityManager;
        if (em == null) throw new IllegalStateException("Entity manager has not been injected
(is the Spring Aspects JAR configured as an AJC/AJDT aspects library?)");
        return em;
    }

    public static long Media.countMedias() {
        return entityManager().createQuery("SELECT COUNT(o) FROM Media o",
Long.class).getSingleResult();
    }

    public static List<Media> Media.findAllMedias() {
        return entityManager().createQuery("SELECT o FROM Media o",
Media.class).getResultList();
    }

    public static List<Media> Media.findAllMedias(String sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM Media o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery, Media.class).getResultList();
    }

    public static Media Media.findMedia(Long id) {
        if (id == null) return null;
        return entityManager().find(Media.class, id);
    }

    public static List<Media> Media.findMediaEntries(int firstResult, int maxResults) {
        return entityManager().createQuery("SELECT o FROM Media o",
Media.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    public static List<Media> Media.findMediaEntries(int firstResult, int maxResults, String
sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM Media o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
```

```
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery,
Media.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    @Transactional
    public void Media.persist() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.persist(this);
    }

    @Transactional
    public void Media.remove() {
        if (this.entityManager == null) this.entityManager = entityManager();
        if (this.entityManager.contains(this)) {
            this.entityManager.remove(this);
        } else {
            Media attached = Media.findMedia(this.id);
            this.entityManager.remove(attached);
        }
    }

    @Transactional
    public void Media.flush() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.flush();
    }

    @Transactional
    public void Media.clear() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.clear();
    }

    @Transactional
    public Media Media.merge() {
        if (this.entityManager == null) this.entityManager = entityManager();
        Media merged = this.entityManager.merge(this);
        this.entityManager.flush();
        return merged;
    }

}
```

13.4.1.6 Fichero "Media_Roo_Jpa_entity.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Version;

privileged aspect Media_Roo_Jpa_Entity {

    declare @type: Media: @Entity;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long Media.id;
```

```
    @Version
    @Column(name = "version")
    private Integer Media.version;

    public Long Media.getId() {
        return this.id;
    }

    public void Media.setId(Long id) {
        this.id = id;
    }

    public Integer Media.getVersion() {
        return this.version;
    }

    public void Media.setVersion(Integer version) {
        this.version = version;
    }

}
```

13.4.1.7 Fichero "Media_Roo_ToString.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

privileged aspect Media_Roo_ToString {

    public String Media.toString() {
        return ReflectionToStringBuilder.toString(this, ToStringStyle.SHORT_PREFIX_STYLE);
    }

}
```

13.4.1.8 Fichero "MediaBatchService_Roo_GvNIXJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.MediaBatchService;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPADeleteClause;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.springframework.transaction.annotation.Transactional;

privileged aspect MediaBatchService_Roo_GvNIXJpaBatch {

    public Class MediaBatchService.getEntity() {
        return Media.class;
    }

    public EntityManager MediaBatchService.entityManager() {
```

```
        return Media.entityManager();
    }

    @Transactional
    public int MediaBatchService.deleteAll() {
        return entityManager().createQuery("DELETE FROM Media").executeUpdate();
    }

    @Transactional
    public int MediaBatchService.deleteIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM Media as m WHERE m.id IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public int MediaBatchService.deleteNotIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM Media as m WHERE m.id NOT IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public void MediaBatchService.create(List<Media> medias) {
        for( Media media : medias) {
            media.persist();
        }
    }

    @Transactional
    public List<Media> MediaBatchService.update(List<Media> medias) {
        List<Media> merged = new ArrayList<Media>();
        for( Media media : medias) {
            merged.add( media.merge() );
        }
        return merged;
    }

    public List<Media> MediaBatchService.findByValues(Map<String, Object> propertyValues) {

        // if there is a filter
        if (propertyValues != null && !propertyValues.isEmpty()) {
            // Prepare a predicate
            BooleanBuilder baseFilterPredicate = new BooleanBuilder();

            // Base filter. Using BooleanBuilder, a cascading builder for
            // Predicate expressions
            PathBuilder<Media> entity = new PathBuilder<Media>(Media.class, "entity");

            // Build base filter
            for (String key : propertyValues.keySet()) {
                baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
            }

            // Create a query with filter
            JPAQuery query = new JPAQuery(Media.entityManager());
            query = query.from(entity);

            // execute query
            return query.where(baseFilterPredicate).list(entity);
        }

        // no filter: return all elements
        return Media.findAllMedias();
    }

    @Transactional
    public long MediaBatchService.deleteByValues(Map<String, Object> propertyValues) {

        // if there no is a filter
        if (propertyValues == null || propertyValues.isEmpty()) {
            throw new IllegalArgumentException("Missing property values");
        }
```

```
        // Prepare a predicate
        BooleanBuilder baseFilterPredicate = new BooleanBuilder();

        // Base filter. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<Media> entity = new PathBuilder<Media>(Media.class, "entity");

        // Build base filter
        for (String key : propertyValues.keySet()) {
            baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
        }

        // Create a query with filter
        JPADeleteClause delete = new JPADeleteClause(Media.entityManager(),entity);

        // execute delete
        return delete.where(baseFilterPredicate).execute();
    }

    @Transactional
    public void MediaBatchService.delete(List<Media> medias) {
        for( Media media : medias) {
            media.remove();
        }
    }

}
```

13.4.1.9 Fichero "MediaBatchService.java":

```
package com.mediaserver.domain;
import org.gvnix.addon.jpa.batch.GvNIXJpaBatch;
import org.springframework.stereotype.Service;

@Service
@GvNIXJpaBatch(entity = Media.class)
public class MediaBatchService {
}
```

13.4.1.10 Fichero "PlayList.java":

```
package com.mediaserver.domain;
import javax.persistence.ManyToOne;
import javax.validation.constraints.Size;

import org.gvnix.addon.jpa.query.GvNIXJpaQuery;
import org.springframework.roo.addon.javabean.RooJavaBean;
import org.springframework.roo.addon.jpa.activerecord.RooJpaActiveRecord;
import org.springframework.roo.addon.tostring.RooToString;

import com.mediaserver.domain.security.User;
/**
 * Esta clase representa las listas de reproducción de la aplicación
 * Se identifican por un nombre
 * @author UO180258
 *
 */
@RooJavaBean
@RooToString
@RooJpaActiveRecord(finders = { "findPlayListsByPlayListName","findPlayListsByOwner" })
public class PlayList {

    /**Nombre de la lista de reproducción
     */
    @Size(min = 3, max = 30)
    private String playListName;

    /**
```

```
     * Creador o propietario de la lista de reproducción
     */
    @ManyToOne
    @GvNIXJpaQuery(filterBy={"email_address"})
    private User owner;
}
```

13.4.1.11 Fichero "PlayList_Roo_Configurable.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import org.springframework.beans.factory.annotation.Configurable;

privileged aspect PlayList_Roo_Configurable {

    declare @type: PlayList: @Configurable;

}
```

13.4.1.12 Fichero "PlayList_Roo_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.security.User;

import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

privileged aspect PlayList_Roo_Finder {

    public static Long PlayList.countFindPlayListsByPlayListName(String playListName) {
        if (playListName == null || playListName.length() == 0) throw new
IllegalArgumentException("The playListName argument is required");
        EntityManager em = PlayList.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM PlayList AS o WHERE o.playListName =
:playListName", Long.class);
        q.setParameter("playListName", playListName);
        return ((Long) q.getSingleResult());
    }

    public static TypedQuery<PlayList> PlayList.findPlayListsByPlayListName(String playListName)
{
        if (playListName == null || playListName.length() == 0) throw new
IllegalArgumentException("The playListName argument is required");
        EntityManager em = PlayList.entityManager();
        TypedQuery<PlayList> q = em.createQuery("SELECT o FROM PlayList AS o WHERE
o.playListName = :playListName", PlayList.class);
        q.setParameter("playListName", playListName);
        return q;
    }

    public static TypedQuery<PlayList> PlayList.findPlayListsByPlayListName(String playListName,
String sortFieldName, String sortOrder) {
        if (playListName == null || playListName.length() == 0) throw new
IllegalArgumentException("The playListName argument is required");
        EntityManager em = PlayList.entityManager();
        StringBuilder queryBuilder = new StringBuilder("SELECT o FROM PlayList AS o WHERE
o.playListName = :playListName");
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            queryBuilder.append(" ORDER BY ").append(sortFieldName);
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                queryBuilder.append(" ").append(sortOrder);
```

```
            }
        }
        TypedQuery<PlayList> q = em.createQuery(queryBuilder.toString(), PlayList.class);
        q.setParameter("playListName", playListName);
        return q;
    }

    public static Long PlayList.countFindPlayListsByOwner(User owner) {
        if (owner == null) throw new IllegalArgumentException("The owner argument is required");
        EntityManager em = PlayList.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM PlayList AS o WHERE o.owner =
:owner", Long.class);
        q.setParameter("owner", owner);
        return ((Long) q.getSingleResult());
    }

    public static TypedQuery<PlayList> PlayList.findPlayListsByOwner(User owner) {
        if (owner == null) throw new IllegalArgumentException("The owner argument is required");
        EntityManager em = PlayList.entityManager();
        TypedQuery<PlayList> q = em.createQuery("SELECT o FROM PlayList AS o WHERE o.owner =
:owner", PlayList.class);
        q.setParameter("owner", owner);
        return q;
    }

    public static TypedQuery<PlayList> PlayList.findPlayListsByOwner(User owner, String
sortFieldName, String sortOrder) {
        if (owner == null) throw new IllegalArgumentException("The owner argument is required");
        EntityManager em = PlayList.entityManager();
        StringBuilder queryBuilder = new StringBuilder("SELECT o FROM PlayList AS o WHERE
o.owner = :owner");
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            queryBuilder.append(" ORDER BY ").append(sortFieldName);
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                queryBuilder.append(" ").append(sortOrder);
            }
        }
        TypedQuery<PlayList> q = em.createQuery(queryBuilder.toString(), PlayList.class);
        q.setParameter("owner", owner);
        return q;
    }

}
```

13.4.1.13 Fichero "PlayList_Roo_JavaBean.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.security.User;

privileged aspect PlayList_Roo_JavaBean {

    public String PlayList.getPlayListName() {
        return this.playListName;
    }

    public void PlayList.setPlayListName(String playListName) {
        this.playListName = playListName;
    }

    public User PlayList.getOwner() {
        return this.owner;
    }

    public void PlayList.setOwner(User owner) {
        this.owner = owner;
    }

}
```

13.4.1.14 Fichero "PlayList_Roo_Jpa_ActiveRecord.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.transaction.annotation.Transactional;

privileged aspect PlayList_Roo_Jpa_ActiveRecord {

    @PersistenceContext
    transient EntityManager PlayList.entityManager;

    public static final List<String> PlayList.fieldNames4OrderClauseFilter =
java.util.Arrays.asList("playListName");

    public static final EntityManager PlayList.entityManager() {
        EntityManager em = new PlayList().entityManager;
        if (em == null) throw new IllegalStateException("Entity manager has not been injected
(is the Spring Aspects JAR configured as an AJC/AJDT aspects library?)");
        return em;
    }

    public static long PlayList.countPlayLists() {
        return entityManager().createQuery("SELECT COUNT(o) FROM PlayList o",
Long.class).getSingleResult();
    }

    public static List<PlayList> PlayList.findAllPlayLists() {
        return entityManager().createQuery("SELECT o FROM PlayList o",
PlayList.class).getResultList();
    }

    public static List<PlayList> PlayList.findAllPlayLists(String sortFieldName, String
sortOrder) {
        String jpaQuery = "SELECT o FROM PlayList o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery, PlayList.class).getResultList();
    }

    public static PlayList PlayList.findPlayList(Long id) {
        if (id == null) return null;
        return entityManager().find(PlayList.class, id);
    }

    public static List<PlayList> PlayList.findPlayListEntries(int firstResult, int maxResults) {
        return entityManager().createQuery("SELECT o FROM PlayList o",
PlayList.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    public static List<PlayList> PlayList.findPlayListEntries(int firstResult, int maxResults,
String sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM PlayList o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery,
PlayList.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    @Transactional
    public void PlayList.persist() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.persist(this);
```

```
    }

    @Transactional
    public void PlayList.remove() {
        if (this.entityManager == null) this.entityManager = entityManager();
        if (this.entityManager.contains(this)) {
            this.entityManager.remove(this);
        } else {
            PlayList attached = PlayList.findPlayList(this.id);
            this.entityManager.remove(attached);
        }
    }

    @Transactional
    public void PlayList.flush() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.flush();
    }

    @Transactional
    public void PlayList.clear() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.clear();
    }

    @Transactional
    public PlayList PlayList.merge() {
        if (this.entityManager == null) this.entityManager = entityManager();
        PlayList merged = this.entityManager.merge(this);
        this.entityManager.flush();
        return merged;
    }

}
```

13.4.1.15 Fichero "PlayList_Roo_Jpa_Entity.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Version;

privileged aspect PlayList_Roo_Jpa_Entity {

    declare @type: PlayList: @Entity;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long PlayList.id;

    @Version
    @Column(name = "version")
    private Integer PlayList.version;

    public Long PlayList.getId() {
        return this.id;
    }

    public void PlayList.setId(Long id) {
        this.id = id;
    }

    public Integer PlayList.getVersion() {
        return this.version;
    }
```

```
    public void PlayList.setVersion(Integer version) {
        this.version = version;
    }

}
```

13.4.1.16 Fichero "PlayList_RooToString.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

privileged aspect PlayList_Roo_ToString {

    public String PlayList.toString() {
        return ReflectionToStringBuilder.toString(this, ToStringStyle.SHORT_PREFIX_STYLE);
    }

}
```

13.4.1.17 Fichero "PlayListBatchService_Roo_GvNIXJpaBatch.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListBatchService;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPADeleteClause;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.springframework.transaction.annotation.Transactional;

privileged aspect PlayListBatchService_Roo_GvNIXJpaBatch {

    public Class PlayListBatchService.getEntity() {
        return PlayList.class;
    }

    public EntityManager PlayListBatchService.entityManager() {
        return PlayList.entityManager();
    }

    @Transactional
    public int PlayListBatchService.deleteAll() {
        return entityManager().createQuery("DELETE FROM PlayList").executeUpdate();
    }

    @Transactional
    public int PlayListBatchService.deleteIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM PlayList as p WHERE p.id IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public int PlayListBatchService.deleteNotIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM PlayList as p WHERE p.id NOT IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }
```

```
    @Transactional
    public void PlayListBatchService.create(List<PlayList> playLists) {
        for( PlayList playlist : playLists) {
            playlist.persist();
        }
    }

    @Transactional
    public List<PlayList> PlayListBatchService.update(List<PlayList> playlists) {
        List<PlayList> merged = new ArrayList<PlayList>();
        for( PlayList playlist : playlists) {
            merged.add( playlist.merge() );
        }
        return merged;
    }

    public List<PlayList> PlayListBatchService.findByValues(Map<String, Object> propertyValues)
{

        // if there is a filter
        if (propertyValues != null && !propertyValues.isEmpty()) {
            // Prepare a predicate
            BooleanBuilder baseFilterPredicate = new BooleanBuilder();

            // Base filter. Using BooleanBuilder, a cascading builder for
            // Predicate expressions
            PathBuilder<PlayList> entity = new PathBuilder<PlayList>(PlayList.class, "entity");

            // Build base filter
            for (String key : propertyValues.keySet()) {
                baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
            }

            // Create a query with filter
            JPAQuery query = new JPAQuery(PlayList.entityManager());
            query = query.from(entity);

            // execute query
            return query.where(baseFilterPredicate).list(entity);
        }

        // no filter: return all elements
        return PlayList.findAllPlayLists();
    }

    @Transactional
    public long PlayListBatchService.deleteByValues(Map<String, Object> propertyValues) {

        // if there no is a filter
        if (propertyValues == null || propertyValues.isEmpty()) {
            throw new IllegalArgumentException("Missing property values");
        }
        // Prepare a predicate
        BooleanBuilder baseFilterPredicate = new BooleanBuilder();

        // Base filter. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<PlayList> entity = new PathBuilder<PlayList>(PlayList.class, "entity");

        // Build base filter
        for (String key : propertyValues.keySet()) {
            baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
        }

        // Create a query with filter
        JPADeleteClause delete = new JPADeleteClause(PlayList.entityManager(),entity);

        // execute delete
        return delete.where(baseFilterPredicate).execute();
    }

    @Transactional
    public void PlayListBatchService.delete(List<PlayList> playlists) {
        for( PlayList playlist : playlists) {
            playlist.remove();
```

```
        }
    }

}
```

13.4.1.18 Fichero "PlayListBatchService.java":

```
package com.mediaserver.domain;
import org.gvnix.addon.jpa.batch.GvNIXJpaBatch;
import org.springframework.stereotype.Service;

@Service
@GvNIXJpaBatch(entity = PlayList.class)
public class PlayListBatchService {
}
```

13.4.1.19 Fichero "PlayListMedia.java":

```
package com.mediaserver.domain;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.validation.constraints.NotNull;

import org.springframework.roo.addon.javabean.RooJavaBean;
import org.springframework.roo.addon.jpa.activerecord.RooJpaActiveRecord;
import org.springframework.roo.addon.tostring.RooToString;

/**
 * Clase que relaciona una PlayList un elemento Media
 * para evitar una asociación múltiple por ambos extremos
 * @author UO180258
 *
 */
@RooJavaBean
@RooToString
@RooJpaActiveRecord(table = "playlist_meidas", finders = { "findPlayListMediasByPlayList",
"findPlayListMediasByMedia" })
public class PlayListMedia implements Comparable<PlayListMedia>{

    /**
     * Lista de reproducción
     */
    @NotNull
    @ManyToOne
    @JoinColumn(name = "playlist_id")
    private PlayList playList;

    /**
     * Medio
     */
    @NotNull
    @ManyToOne
    @JoinColumn(name = "media_id")
    private Media media;

    /**
     * Comparación para ordenación ascendente de elemenots de la clase según el nombre del medio
     * @param o
     * @return
     */
        @Override
        public int compareTo(PlayListMedia o) {
                int ret=o.getMedia().getName().compareTo(this.media.getName());
                return ret;
        }
}
```

13.4.1.20 Fichero "PlayListMedia_Roo_Configurable.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayListMedia;
import org.springframework.beans.factory.annotation.Configurable;

privileged aspect PlayListMedia_Roo_Configurable {

    declare @type: PlayListMedia: @Configurable;

}
```

13.4.1.21 Fichero "PlayListMedia_Roo_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

privileged aspect PlayListMedia_Roo_Finder {

    public static Long PlayListMedia.countFindPlayListMediasByMedia(Media media) {
        if (media == null) throw new IllegalArgumentException("The media argument is required");
        EntityManager em = PlayListMedia.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM PlayListMedia AS o WHERE o.media =
:media", Long.class);
        q.setParameter("media", media);
        return ((Long) q.getSingleResult());
    }

    public static Long PlayListMedia.countFindPlayListMediasByPlayList(PlayList playList) {
        if (playList == null) throw new IllegalArgumentException("The playList argument is
required");
        EntityManager em = PlayListMedia.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM PlayListMedia AS o WHERE o.playList
= :playList", Long.class);
        q.setParameter("playList", playList);
        return ((Long) q.getSingleResult());
    }

    public static TypedQuery<PlayListMedia> PlayListMedia.findPlayListMediasByMedia(Media media)
{
        if (media == null) throw new IllegalArgumentException("The media argument is required");
        EntityManager em = PlayListMedia.entityManager();
        TypedQuery<PlayListMedia> q = em.createQuery("SELECT o FROM PlayListMedia AS o WHERE
o.media = :media", PlayListMedia.class);
        q.setParameter("media", media);
        return q;
    }

    public static TypedQuery<PlayListMedia> PlayListMedia.findPlayListMediasByMedia(Media media,
String sortFieldName, String sortOrder) {
        if (media == null) throw new IllegalArgumentException("The media argument is required");
        EntityManager em = PlayListMedia.entityManager();
        StringBuilder queryBuilder = new StringBuilder("SELECT o FROM PlayListMedia AS o WHERE
o.media = :media");
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            queryBuilder.append(" ORDER BY ").append(sortFieldName);
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                queryBuilder.append(" ").append(sortOrder);
            }
        }
        TypedQuery<PlayListMedia> q = em.createQuery(queryBuilder.toString(),
PlayListMedia.class);
```

```
        q.setParameter("media", media);
        return q;
    }

    public static TypedQuery<PlayListMedia> PlayListMedia.findPlayListMediasByPlayList(PlayList
playList) {
        if (playList == null) throw new IllegalArgumentException("The playList argument is
required");
        EntityManager em = PlayListMedia.entityManager();
        TypedQuery<PlayListMedia> q = em.createQuery("SELECT o FROM PlayListMedia AS o WHERE
o.playList = :playList", PlayListMedia.class);
        q.setParameter("playList", playList);
        return q;
    }

    public static TypedQuery<PlayListMedia> PlayListMedia.findPlayListMediasByPlayList(PlayList
playList, String sortFieldName, String sortOrder) {
        if (playList == null) throw new IllegalArgumentException("The playList argument is
required");
        EntityManager em = PlayListMedia.entityManager();
        StringBuilder queryBuilder = new StringBuilder("SELECT o FROM PlayListMedia AS o WHERE
o.playList = :playList");
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            queryBuilder.append(" ORDER BY ").append(sortFieldName);
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                queryBuilder.append(" ").append(sortOrder);
            }
        }
        TypedQuery<PlayListMedia> q = em.createQuery(queryBuilder.toString(),
PlayListMedia.class);
        q.setParameter("playList", playList);
        return q;
    }

}
```

13.4.1.22 Fichero "PlayListMedia_Roo_JavaBean.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;

privileged aspect PlayListMedia_Roo_JavaBean {

    public PlayList PlayListMedia.getPlayList() {
        return this.playList;
    }

    public void PlayListMedia.setPlayList(PlayList playList) {
        this.playList = playList;
    }

    public Media PlayListMedia.getMedia() {
        return this.media;
    }

    public void PlayListMedia.setMedia(Media media) {
        this.media = media;
    }

}
```

13.4.1.23 Fichero "PlayListMedia_Roo_Jpa_ActiveRecord.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;
```

```
import com.mediaserver.domain.PlayListMedia;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.transaction.annotation.Transactional;

privileged aspect PlayListMedia_Roo_Jpa_ActiveRecord {

    @PersistenceContext
    transient EntityManager PlayListMedia.entityManager;

    public static final List<String> PlayListMedia.fieldNames4OrderClauseFilter =
java.util.Arrays.asList("playList", "media");

    public static final EntityManager PlayListMedia.entityManager() {
        EntityManager em = new PlayListMedia().entityManager;
        if (em == null) throw new IllegalStateException("Entity manager has not been injected
(is the Spring Aspects JAR configured as an AJC/AJDT aspects library?)");
        return em;
    }

    public static long PlayListMedia.countPlayListMedias() {
        return entityManager().createQuery("SELECT COUNT(o) FROM PlayListMedia o",
Long.class).getSingleResult();
    }

    public static List<PlayListMedia> PlayListMedia.findAllPlayListMedias() {
        return entityManager().createQuery("SELECT o FROM PlayListMedia o",
PlayListMedia.class).getResultList();
    }

    public static List<PlayListMedia> PlayListMedia.findAllPlayListMedias(String sortFieldName,
String sortOrder) {
        String jpaQuery = "SELECT o FROM PlayListMedia o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery, PlayListMedia.class).getResultList();
    }

    public static PlayListMedia PlayListMedia.findPlayListMedia(Long id) {
        if (id == null) return null;
        return entityManager().find(PlayListMedia.class, id);
    }

    public static List<PlayListMedia> PlayListMedia.findPlayListMediaEntries(int firstResult,
int maxResults) {
        return entityManager().createQuery("SELECT o FROM PlayListMedia o",
PlayListMedia.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    public static List<PlayListMedia> PlayListMedia.findPlayListMediaEntries(int firstResult,
int maxResults, String sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM PlayListMedia o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery,
PlayListMedia.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    @Transactional
    public void PlayListMedia.persist() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.persist(this);
    }

    @Transactional
```

```
    public void PlayListMedia.remove() {
        if (this.entityManager == null) this.entityManager = entityManager();
        if (this.entityManager.contains(this)) {
            this.entityManager.remove(this);
        } else {
            PlayListMedia attached = PlayListMedia.findPlayListMedia(this.id);
            this.entityManager.remove(attached);
        }
    }

    @Transactional
    public void PlayListMedia.flush() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.flush();
    }

    @Transactional
    public void PlayListMedia.clear() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.clear();
    }

    @Transactional
    public PlayListMedia PlayListMedia.merge() {
        if (this.entityManager == null) this.entityManager = entityManager();
        PlayListMedia merged = this.entityManager.merge(this);
        this.entityManager.flush();
        return merged;
    }

}
```

13.4.1.24 Fichero "PlayListMedia_Roo_Jpa_entity.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain;

import com.mediaserver.domain.PlayListMedia;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Version;

privileged aspect PlayListMedia_Roo_Jpa_Entity {

    declare @type: PlayListMedia: @Entity;

    declare @type: PlayListMedia: @Table(name = "playlist_meidas");

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long PlayListMedia.id;

    @Version
    @Column(name = "version")
    private Integer PlayListMedia.version;

    public Long PlayListMedia.getId() {
        return this.id;
    }

    public void PlayListMedia.setId(Long id) {
        this.id = id;
    }

    public Integer PlayListMedia.getVersion() {
```

```
        return this.version;
    }

    public void PlayListMedia.setVersion(Integer version) {
        this.version = version;
    }

}
```

13.4.1.25 Fichero "PlayListMedia_Roo_ToString.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayListMedia;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

privileged aspect PlayListMedia_Roo_ToString {

    public String PlayListMedia.toString() {
        return ReflectionToStringBuilder.toString(this, ToStringStyle.SHORT_PREFIX_STYLE);
    }

}
```

13.4.1.26 Fichero "PlayListMediaBatchService_Roo_GvNIXJpaBatch.aj":

```
package com.mediaserver.domain;

import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.PlayListMediaBatchService;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPADeleteClause;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.springframework.transaction.annotation.Transactional;

privileged aspect PlayListMediaBatchService_Roo_GvNIXJpaBatch {

    public Class PlayListMediaBatchService.getEntity() {
        return PlayListMedia.class;
    }

    public EntityManager PlayListMediaBatchService.entityManager() {
        return PlayListMedia.entityManager();
    }

    @Transactional
    public int PlayListMediaBatchService.deleteAll() {
        return entityManager().createQuery("DELETE FROM PlayListMedia").executeUpdate();
    }

    @Transactional
    public int PlayListMediaBatchService.deleteIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM PlayListMedia as p WHERE p.id IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public int PlayListMediaBatchService.deleteNotIn(List<Long> ids) {
```

```
        Query query = entityManager().createQuery("DELETE FROM PlayListMedia as p WHERE p.id NOT
IN (:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public void PlayListMediaBatchService.create(List<PlayListMedia> playListMedias) {
        for( PlayListMedia playlistmedia : playListMedias) {
            playlistmedia.persist();
        }
    }

    @Transactional
    public List<PlayListMedia> PlayListMediaBatchService.update(List<PlayListMedia>
playlistmedias) {
        List<PlayListMedia> merged = new ArrayList<PlayListMedia>();
        for( PlayListMedia playlistmedia : playlistmedias) {
            merged.add( playlistmedia.merge() );
        }
        return merged;
    }

    public List<PlayListMedia> PlayListMediaBatchService.findByValues(Map<String, Object>
propertyValues) {

        // if there is a filter
        if (propertyValues != null && !propertyValues.isEmpty()) {
            // Prepare a predicate
            BooleanBuilder baseFilterPredicate = new BooleanBuilder();

            // Base filter. Using BooleanBuilder, a cascading builder for
            // Predicate expressions
            PathBuilder<PlayListMedia> entity = new
PathBuilder<PlayListMedia>(PlayListMedia.class, "entity");

            // Build base filter
            for (String key : propertyValues.keySet()) {
                baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
            }

            // Create a query with filter
            JPAQuery query = new JPAQuery(PlayListMedia.entityManager());
            query = query.from(entity);

            // execute query
            return query.where(baseFilterPredicate).list(entity);
        }

        // no filter: return all elements
        return PlayListMedia.findAllPlayListMedias();
    }

    @Transactional
    public long PlayListMediaBatchService.deleteByValues(Map<String, Object> propertyValues) {

        // if there no is a filter
        if (propertyValues == null || propertyValues.isEmpty()) {
            throw new IllegalArgumentException("Missing property values");
        }
        // Prepare a predicate
        BooleanBuilder baseFilterPredicate = new BooleanBuilder();

        // Base filter. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<PlayListMedia> entity = new PathBuilder<PlayListMedia>(PlayListMedia.class,
"entity");

        // Build base filter
        for (String key : propertyValues.keySet()) {
            baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
        }

        // Create a query with filter
        JPADeleteClause delete = new JPADeleteClause(PlayListMedia.entityManager(),entity);
```

```
        // execute delete
        return delete.where(baseFilterPredicate).execute();
    }

    @Transactional
    public void PlayListMediaBatchService.delete(List<PlayListMedia> playlistmedias) {
        for( PlayListMedia playlistmedia : playlistmedias) {
            playlistmedia.remove();
        }
    }

}
```

13.4.1.27 Fichero "PlayListMediaBatchService.java":

```
package com.mediaserver.domain;
import org.gvnix.addon.jpa.batch.GvNIXJpaBatch;
import org.springframework.stereotype.Service;

@Service
@GvNIXJpaBatch(entity = PlayListMedia.class)
public class PlayListMediaBatchService {
}
```

13.4.2 Paquete com.mediaserver.domain.security:

13.4.2.1 Fichero "Role.java":

```
package com.mediaserver.domain.security;
import javax.persistence.Column;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import org.springframework.roo.addon.javabean.RooJavaBean;
import org.springframework.roo.addon.jpa.activerecord.RooJpaActiveRecord;
import org.springframework.roo.addon.tostring.RooToString;

/**
 * Esta clase representa los roles de los usuarios de la aplicación
 * Según los dinstintos roles los usuarios tendrán una serie de permisios y acciones posibles u
otras
 * @author UO180258
 *
 */
@RooJavaBean
@RooToString
@RooJpaActiveRecord(table = "roles", finders = { "findRolesByNameEquals" })
public class Role {

        public Role(){
        }
        /**
         * Constructor de la clase, a partir de su nombre y descripción
         * @param name nombre del rol
         * @param description descripción del rol
         */
    public Role(String name, String description) {
                this.name=name;
                this.description=description;
        }

        /**
         * Nombre del rol
     */
    @NotNull
    @Column(name = "name", unique = true)
```

```
    @Size(min = 1)
    //@Pattern(regexp = "^ROLE_[A-Z]*")
    private String name;

    /**
     * Descripción del rol
     */
    @NotNull
    @Column(name = "description")
    @Size(max = 200)
    private String description;
}
```

13.4.2.2 Fichero "Role_Roo_Configurable.aj":

```
package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;
import org.springframework.beans.factory.annotation.Configurable;

privileged aspect Role_Roo_Configurable {

    declare @type: Role: @Configurable;

}
```

13.4.2.3 Fichero "Role_Roo_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;
import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

privileged aspect Role_Roo_Finder {

    public static Long Role.countFindRolesByNameEquals(String name) {
        if (name == null || name.length() == 0) throw new IllegalArgumentException("The name
argument is required");
        EntityManager em = Role.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM Role AS o WHERE o.name = :name",
Long.class);
        q.setParameter("name", name);
        return ((Long) q.getSingleResult());
    }

    public static TypedQuery<Role> Role.findRolesByNameEquals(String name) {
        if (name == null || name.length() == 0) throw new IllegalArgumentException("The name
argument is required");
        EntityManager em = Role.entityManager();
        TypedQuery<Role> q = em.createQuery("SELECT o FROM Role AS o WHERE o.name = :name",
Role.class);
        q.setParameter("name", name);
        return q;
    }

    public static TypedQuery<Role> Role.findRolesByNameEquals(String name, String sortFieldName,
String sortOrder) {
        if (name == null || name.length() == 0) throw new IllegalArgumentException("The name
argument is required");
        EntityManager em = Role.entityManager();
        String jpaQuery = "SELECT o FROM Role AS o WHERE o.name = :name";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
```

```
        }
        TypedQuery<Role> q = em.createQuery(jpaQuery, Role.class);
        q.setParameter("name", name);
        return q;
    }

}
```

13.4.2.4 Fichero "Role_Roo_JavaBean.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;

privileged aspect Role_Roo_JavaBean {

    public String Role.getName() {
        return this.name;
    }

    public void Role.setName(String name) {
        this.name = name;
    }

    public String Role.getDescription() {
        return this.description;
    }

    public void Role.setDescription(String description) {
        this.description = description;
    }

}
```

13.4.2.5 Fichero "Role_Roo_Jpa_ActiveRecord.aj":

```
package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.transaction.annotation.Transactional;

privileged aspect Role_Roo_Jpa_ActiveRecord {

    @PersistenceContext
    transient EntityManager Role.entityManager;

    public static final List<String> Role.fieldNames4OrderClauseFilter =
java.util.Arrays.asList("name", "description");

    public static final EntityManager Role.entityManager() {
        EntityManager em = new Role().entityManager;
        if (em == null) throw new IllegalStateException("Entity manager has not been injected
(is the Spring Aspects JAR configured as an AJC/AJDT aspects library?)");
        return em;
    }

    public static long Role.countRoles() {
        return entityManager().createQuery("SELECT COUNT(o) FROM Role o",
Long.class).getSingleResult();
    }

    public static List<Role> Role.findAllRoles() {
        return entityManager().createQuery("SELECT o FROM Role o", Role.class).getResultList();
    }
```

```
    public static List<Role> Role.findAllRoles(String sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM Role o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery, Role.class).getResultList();
    }

    public static Role Role.findRole(Long id) {
        if (id == null) return null;
        return entityManager().find(Role.class, id);
    }

    public static List<Role> Role.findRoleEntries(int firstResult, int maxResults) {
        return entityManager().createQuery("SELECT o FROM Role o",
Role.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    public static List<Role> Role.findRoleEntries(int firstResult, int maxResults, String
sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM Role o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery,
Role.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    @Transactional
    public void Role.persist() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.persist(this);
    }

    @Transactional
    public void Role.remove() {
        if (this.entityManager == null) this.entityManager = entityManager();
        if (this.entityManager.contains(this)) {
            this.entityManager.remove(this);
        } else {
            Role attached = Role.findRole(this.id);
            this.entityManager.remove(attached);
        }
    }

    @Transactional
    public void Role.flush() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.flush();
    }

    @Transactional
    public void Role.clear() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.clear();
    }

    @Transactional
    public Role Role.merge() {
        if (this.entityManager == null) this.entityManager = entityManager();
        Role merged = this.entityManager.merge(this);
        this.entityManager.flush();
        return merged;
    }

}
```

13.4.2.6 Fichero "Role_Roo_Jpa_Entity.aj":

```
package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Version;

privileged aspect Role_Roo_Jpa_Entity {

    declare @type: Role: @Entity;

    declare @type: Role: @Table(name = "roles");

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long Role.id;

    @Version
    @Column(name = "version")
    private Integer Role.version;

    public Long Role.getId() {
        return this.id;
    }

    public void Role.setId(Long id) {
        this.id = id;
    }

    public Integer Role.getVersion() {
        return this.version;
    }

    public void Role.setVersion(Integer version) {
        this.version = version;
    }

}
```

13.4.2.7 Fichero "Role_Roo_ToString.aj":

```
package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

privileged aspect Role_Roo_ToString {

    public String Role.toString() {
        return ReflectionToStringBuilder.toString(this, ToStringStyle.SHORT_PREFIX_STYLE);
    }

}
```

13.4.2.8 Fichero "RoleBatchService_Roo_GvNIXJpaBatch.aj":

```
package com.mediaserver.domain.security;
```

```
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPADeleteClause;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.RoleBatchService;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.springframework.transaction.annotation.Transactional;

privileged aspect RoleBatchService_Roo_GvNIXJpaBatch {

    public Class RoleBatchService.getEntity() {
        return Role.class;
    }

    public EntityManager RoleBatchService.entityManager() {
        return Role.entityManager();
    }

    @Transactional
    public int RoleBatchService.deleteAll() {
        return entityManager().createQuery("DELETE FROM Role").executeUpdate();
    }

    @Transactional
    public int RoleBatchService.deleteIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM Role as r WHERE r.id IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public int RoleBatchService.deleteNotIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM Role as r WHERE r.id NOT IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public void RoleBatchService.create(List<Role> roles) {
        for( Role role : roles) {
            role.persist();
        }
    }

    @Transactional
    public List<Role> RoleBatchService.update(List<Role> roles) {
        List<Role> merged = new ArrayList<Role>();
        for( Role role : roles) {
            merged.add( role.merge() );
        }
        return merged;
    }

    public List<Role> RoleBatchService.findByValues(Map<String, Object> propertyValues) {

        // if there is a filter
        if (propertyValues != null && !propertyValues.isEmpty()) {
            // Prepare a predicate
            BooleanBuilder baseFilterPredicate = new BooleanBuilder();

            // Base filter. Using BooleanBuilder, a cascading builder for
            // Predicate expressions
            PathBuilder<Role> entity = new PathBuilder<Role>(Role.class, "entity");

            // Build base filter
            for (String key : propertyValues.keySet()) {
                baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
            }
```

```
            // Create a query with filter
            JPAQuery query = new JPAQuery(Role.entityManager());
            query = query.from(entity);

            // execute query
            return query.where(baseFilterPredicate).list(entity);
        }

        // no filter: return all elements
        return Role.findAllRoles();
    }

    @Transactional
    public long RoleBatchService.deleteByValues(Map<String, Object> propertyValues) {

        // if there no is a filter
        if (propertyValues == null || propertyValues.isEmpty()) {
            throw new IllegalArgumentException("Missing property values");
        }
        // Prepare a predicate
        BooleanBuilder baseFilterPredicate = new BooleanBuilder();

        // Base filter. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<Role> entity = new PathBuilder<Role>(Role.class, "entity");

        // Build base filter
        for (String key : propertyValues.keySet()) {
            baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
        }

        // Create a query with filter
        JPADeleteClause delete = new JPADeleteClause(Role.entityManager(),entity);

        // execute delete
        return delete.where(baseFilterPredicate).execute();
    }

    @Transactional
    public void RoleBatchService.delete(List<Role> roles) {
        for( Role role : roles) {
            role.remove();
        }
    }

}
```

13.4.2.9 Fichero "RoleBatchService.java":

```
package com.mediaserver.domain.security;
import org.gvnix.addon.jpa.batch.GvNIXJpaBatch;
import org.springframework.stereotype.Service;

@Service
@GvNIXJpaBatch(entity = Role.class)
public class RoleBatchService {
}
```

13.4.2.10 Fichero "User.java"

```
package com.mediaserver.domain.security;

import java.util.Date;
import java.util.List;

import javax.persistence.Column;
```

```
import javax.persistence.EntityManager;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.persistence.TypedQuery;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.roo.addon.javabean.RooJavaBean;
import org.springframework.roo.addon.jpa.activerecord.RooJpaActiveRecord;
import org.springframework.roo.addon.tostring.RooToString;
import org.springframework.security.core.context.SecurityContextHolder;

import com.mediaserver.domain.security.attribute.CreateAttributes;

/**
 * Esta clase representa a los usuarios de la aplicación
 * @author UO180258
 *
 */
@RooJavaBean
@RooToString
@RooJpaActiveRecord(table = "users", finders = { "findUsersByEmailAddress" })
public class User {

        /**
         * Nombre del usuario
         */
    @NotNull
    @Size(min = 1)
    @Column(name = "first_name")
    private String firstName;
    /**
     * Apellido del usuario
     */
    @NotNull
    @Size(min = 1)
    @Column(name = "last_name")
    private String lastName;
    /**
     * Dirección de email del usuario, será unica para cada usuario
     */
    @NotNull
    @Size(min = 1)
    @Column(name = "email_address", unique = true)
    private String emailAddress;

    /**
     * password de acceso del usuario
     */
    @NotNull(groups = CreateAttributes.class) // Submission is required only on create
    @Size(min = 1)
    @Column(name = "password")
    private String password;

    /**
     * Fecha de activación del usuario en el sistema
     */
    @Temporal(TemporalType.TIMESTAMP)
    @DateTimeFormat(style = "M-")
    @Column(name = "activation_date")
    private Date activationDate;

    /**
     * Estado de habilitación, permite activar a los usuarios para que puedan
     * acceder al sistema una vez se han registrado
     */
    @Column(name = "enabled")
    private Boolean enabled;

    /**
     * Estado de bloqueo, permite bloquar el acceso a los usuarios al sistema
     */
        @Column(name = "locked")
    private Boolean locked;
```

```
        /**
         * Este metodo devuelve el usuario que está utilizando el sistema en la sesión actual
         * obteniendolo sus datos del contexto y completando los datos del objeto
         *  con los de la base de datos
         * @return
         */
        public static User getContextUser (){
                User user = null;
                String username = SecurityContextHolder.getContext()
                                .getAuthentication().getName();
                List<User> userList = User.findUsersByEmailAddress(
                                username).getResultList();
                user = userList.get(0);// cojemos el pimer resultado
                return user;
        }
        /**
         * Encuenta un usuario identificado por su id a partir de su dirección de email
         * @param emailAddress
         * @return
         */
        public static Long countFindUsersByEmailAddress(String emailAddress) {
        if (emailAddress == null || emailAddress.length() == 0) throw new
IllegalArgumentException("The emailAddress argument is required");
        EntityManager em = User.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM User AS o WHERE o.emailAddress =
:emailAddress", Long.class);
        q.setParameter("emailAddress", emailAddress);
        Long ret=((Long) q.getSingleResult());
        return ret;
    }

        /**
         * Devuelve una TypeddQuery de usuarios a partir de su email
         * @param emailAddress
         * @return
         */
        public static TypedQuery<User> findUsersByEmailAddress(String emailAddress) {
        if (emailAddress == null || emailAddress.length() == 0) throw new
IllegalArgumentException("The emailAddress argument is required");
        EntityManager em = User.entityManager();
        TypedQuery<User> q = em.createQuery("SELECT o FROM User AS o WHERE o.emailAddress =
:emailAddress", User.class);
        q.setParameter("emailAddress", emailAddress);
        return q;
    }
        /**
         * Devuelve una TypeddQuery de usuarios a partir de su email ordenados según el
parametro sortOrder
         * @param emailAddress
         * @param sortFieldName
         * @param sortOrder puede tener los valores ASC para orden ascendente o DESC para orden
descendente
         * @return
         */
        public static TypedQuery<User> findUsersByEmailAddress(String emailAddress, String
sortFieldName, String sortOrder) {
        if (emailAddress == null || emailAddress.length() == 0) throw new
IllegalArgumentException("The emailAddress argument is required");
        EntityManager em = User.entityManager();
        String jpaQuery = "SELECT o FROM User AS o WHERE o.emailAddress = :emailAddress";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        TypedQuery<User> q = em.createQuery(jpaQuery, User.class);
        q.setParameter("emailAddress", emailAddress);
        return q;
    }
}
```

13.4.2.11 Fichero "User_Roo_Configurable":

```
package com.mediaserver.domain.security;

import com.mediaserver.domain.security.User;
import org.springframework.beans.factory.annotation.Configurable;

privileged aspect User_Roo_Configurable {

    declare @type: User: @Configurable;
}
```

13.4.2.12 Fichero "User_Roo_JavaBean.aj":

```
package com.mediaserver.domain.security;

import com.mediaserver.domain.security.User;
import java.util.Date;

privileged aspect User_Roo_JavaBean {

    public String User.getFirstName() {
        return this.firstName;
    }

    public void User.setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String User.getLastName() {
        return this.lastName;
    }

    public void User.setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String User.getEmailAddress() {
        return this.emailAddress;
    }

    public void User.setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public String User.getPassword() {
        return this.password;
    }

    public void User.setPassword(String password) {
        this.password = password;
    }

    public Date User.getActivationDate() {
        return this.activationDate;
    }

    public void User.setActivationDate(Date activationDate) {
        this.activationDate = activationDate;
    }

    public Boolean User.getEnabled() {
        return this.enabled;
    }

    public void User.setEnabled(Boolean enabled) {
        this.enabled = enabled;
    }

    public Boolean User.getLocked() {
        return this.locked;
```

```
    }

    public void User.setLocked(Boolean locked) {
        this.locked = locked;
    }

}
```

13.4.2.13 Fichero "User_Roo_Jpa_ActiveRecord.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.User;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.transaction.annotation.Transactional;

privileged aspect User_Roo_Jpa_ActiveRecord {

    @PersistenceContext
    transient EntityManager User.entityManager;

    public static final List<String> User.fieldNames4OrderClauseFilter =
java.util.Arrays.asList("firstName", "lastName", "emailAddress", "password", "activationDate",
"enabled", "locked");

    public static final EntityManager User.entityManager() {
        EntityManager em = new User().entityManager;
        if (em == null) throw new IllegalStateException("Entity manager has not been injected
(is the Spring Aspects JAR configured as an AJC/AJDT aspects library?)");
        return em;
    }

    public static long User.countUsers() {
        return entityManager().createQuery("SELECT COUNT(o) FROM User o",
Long.class).getSingleResult();
    }

    public static List<User> User.findAllUsers() {
        return entityManager().createQuery("SELECT o FROM User o", User.class).getResultList();
    }

    public static List<User> User.findAllUsers(String sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM User o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery, User.class).getResultList();
    }

    public static User User.findUser(Long id) {
        if (id == null) return null;
        return entityManager().find(User.class, id);
    }

    public static List<User> User.findUserEntries(int firstResult, int maxResults) {
        return entityManager().createQuery("SELECT o FROM User o",
User.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    public static List<User> User.findUserEntries(int firstResult, int maxResults, String
sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM User o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
```

```
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery,
User.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    @Transactional
    public void User.persist() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.persist(this);
    }

    @Transactional
    public void User.remove() {
        if (this.entityManager == null) this.entityManager = entityManager();
        if (this.entityManager.contains(this)) {
            this.entityManager.remove(this);
        } else {
            User attached = User.findUser(this.id);
            this.entityManager.remove(attached);
        }
    }

    @Transactional
    public void User.flush() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.flush();
    }

    @Transactional
    public void User.clear() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.clear();
    }

    @Transactional
    public User User.merge() {
        if (this.entityManager == null) this.entityManager = entityManager();
        User merged = this.entityManager.merge(this);
        this.entityManager.flush();
        return merged;
    }

}
```

13.4.2.14 Fichero "User_Roo_Jpa_Entity.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.User;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Version;

privileged aspect User_Roo_Jpa_Entity {

    declare @type: User: @Entity;

    declare @type: User: @Table(name = "users");

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Column(name = "id")
    private Long User.id;

    @Version
    @Column(name = "version")
    private Integer User.version;

    public Long User.getId() {
        return this.id;
    }

    public void User.setId(Long id) {
        this.id = id;
    }

    public Integer User.getVersion() {
        return this.version;
    }

    public void User.setVersion(Integer version) {
        this.version = version;
    }

}
```

13.4.2.15 Fichero "User_Roo_ToString.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.User;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;

privileged aspect User_Roo_ToString {

    public String User.toString() {
        return ReflectionToStringBuilder.toString(this, ToStringStyle.SHORT_PREFIX_STYLE);
    }

}
```

13.4.2.16 Fichero "UserBatchService_Roo_GvNIXJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPADeleteClause;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserBatchService;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.springframework.transaction.annotation.Transactional;

privileged aspect UserBatchService_Roo_GvNIXJpaBatch {

    public Class UserBatchService.getEntity() {
        return User.class;
    }
```

```
    public EntityManager UserBatchService.entityManager() {
        return User.entityManager();
    }

    @Transactional
    public int UserBatchService.deleteAll() {
        return entityManager().createQuery("DELETE FROM User").executeUpdate();
    }

    @Transactional
    public int UserBatchService.deleteIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM User as u WHERE u.id IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public int UserBatchService.deleteNotIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM User as u WHERE u.id NOT IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public void UserBatchService.create(List<User> users) {
        for( User user : users) {
            user.persist();
        }
    }

    @Transactional
    public List<User> UserBatchService.update(List<User> users) {
        List<User> merged = new ArrayList<User>();
        for( User user : users) {
            merged.add( user.merge() );
        }
        return merged;
    }

    public List<User> UserBatchService.findByValues(Map<String, Object> propertyValues) {

        // if there is a filter
        if (propertyValues != null && !propertyValues.isEmpty()) {
            // Prepare a predicate
            BooleanBuilder baseFilterPredicate = new BooleanBuilder();

            // Base filter. Using BooleanBuilder, a cascading builder for
            // Predicate expressions
            PathBuilder<User> entity = new PathBuilder<User>(User.class, "entity");

            // Build base filter
            for (String key : propertyValues.keySet()) {
                baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
            }

            // Create a query with filter
            JPAQuery query = new JPAQuery(User.entityManager());
            query = query.from(entity);

            // execute query
            return query.where(baseFilterPredicate).list(entity);
        }

        // no filter: return all elements
        return User.findAllUsers();
    }

    @Transactional
    public long UserBatchService.deleteByValues(Map<String, Object> propertyValues) {

        // if there no is a filter
        if (propertyValues == null || propertyValues.isEmpty()) {
            throw new IllegalArgumentException("Missing property values");
```

```
        }
        // Prepare a predicate
        BooleanBuilder baseFilterPredicate = new BooleanBuilder();

        // Base filter. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<User> entity = new PathBuilder<User>(User.class, "entity");

        // Build base filter
        for (String key : propertyValues.keySet()) {
            baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
        }

        // Create a query with filter
        JPADeleteClause delete = new JPADeleteClause(User.entityManager(),entity);

        // execute delete
        return delete.where(baseFilterPredicate).execute();
    }

    @Transactional
    public void UserBatchService.delete(List<User> users) {
        for( User user : users) {
            user.remove();
        }
    }

}
```

13.4.2.17 Fichero "UserBatchService.java":

```
package com.mediaserver.domain.security;
import org.gvnix.addon.jpa.batch.GvNIXJpaBatch;
import org.springframework.stereotype.Service;

@Service
@GvNIXJpaBatch(entity = User.class)
public class UserBatchService {
}
```

13.4.2.18 Fichero "UserRole.java":

```
package com.mediaserver.domain.security;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.validation.constraints.NotNull;

import org.springframework.roo.addon.javabean.RooJavaBean;
import org.springframework.roo.addon.jpa.activerecord.RooJpaActiveRecord;
import org.springframework.roo.addon.tostring.RooToString;

/**
 * Clase que relaciona un usuario con un rol
 * para evitar la relacion múltiple por ambos extremos
 * @author UO180258
 *
 */
@RooJavaBean
@RooToString
@RooJpaActiveRecord(table = "users_roles", finders = { "findUserRolesByUser" })
public class UserRole {

    /**
     * Usuario objetivo de la relación
     */
    @NotNull
    @ManyToOne
    @JoinColumn(name = "user_id")
```

```
    private User user;

    /**
     * Rol objetivo de la relación
     */
    @NotNull
    @ManyToOne
    @JoinColumn(name = "role_id")
    private Role role;
}
```

13.4.2.19 Fichero "UserRole_Roo_Configurable.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.UserRole;
import org.springframework.beans.factory.annotation.Configurable;

privileged aspect UserRole_Roo_Configurable {

    declare @type: UserRole: @Configurable;

}
```

13.4.2.20 Fichero "UserRole_Roo_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;
import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;

privileged aspect UserRole_Roo_Finder {

    public static Long UserRole.countFindUserRolesByUser(User user) {
        if (user == null) throw new IllegalArgumentException("The user argument is required");
        EntityManager em = UserRole.entityManager();
        TypedQuery q = em.createQuery("SELECT COUNT(o) FROM UserRole AS o WHERE o.user = :user",
Long.class);
        q.setParameter("user", user);
        return ((Long) q.getSingleResult());
    }

    public static TypedQuery<UserRole> UserRole.findUserRolesByUser(User user) {
        if (user == null) throw new IllegalArgumentException("The user argument is required");
        EntityManager em = UserRole.entityManager();
        TypedQuery<UserRole> q = em.createQuery("SELECT o FROM UserRole AS o WHERE o.user =
:user", UserRole.class);
        q.setParameter("user", user);
        return q;
    }

    public static TypedQuery<UserRole> UserRole.findUserRolesByUser(User user, String
sortFieldName, String sortOrder) {
        if (user == null) throw new IllegalArgumentException("The user argument is required");
        EntityManager em = UserRole.entityManager();
        String jpaQuery = "SELECT o FROM UserRole AS o WHERE o.user = :user";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
```

```
        TypedQuery<UserRole> q = em.createQuery(jpaQuery, UserRole.class);
        q.setParameter("user", user);
        return q;
    }

}
```

13.4.2.21 Fichero "UserRole_Roo_JavaBean.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;

privileged aspect UserRole_Roo_JavaBean {

    public User UserRole.getUser() {
        return this.user;
    }

    public void UserRole.setUser(User user) {
        this.user = user;
    }

    public Role UserRole.getRole() {
        return this.role;
    }

    public void UserRole.setRole(Role role) {
        this.role = role;
    }

}
```

13.4.2.22 Fichero "UserRole_Roo_Jpa_ActiveRecord.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.UserRole;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;
import org.springframework.transaction.annotation.Transactional;

privileged aspect UserRole_Roo_Jpa_ActiveRecord {

    @PersistenceContext
    transient EntityManager UserRole.entityManager;

    public static final List<String> UserRole.fieldNames4OrderClauseFilter =
java.util.Arrays.asList("user", "role");

    public static final EntityManager UserRole.entityManager() {
        EntityManager em = new UserRole().entityManager;
        if (em == null) throw new IllegalStateException("Entity manager has not been injected
(is the Spring Aspects JAR configured as an AJC/AJDT aspects library?)");
        return em;
    }

    public static long UserRole.countUserRoles() {
```

```
        return entityManager().createQuery("SELECT COUNT(o) FROM UserRole o",
Long.class).getSingleResult();
    }

    public static List<UserRole> UserRole.findAllUserRoles() {
        return entityManager().createQuery("SELECT o FROM UserRole o",
UserRole.class).getResultList();
    }

    public static List<UserRole> UserRole.findAllUserRoles(String sortFieldName, String
sortOrder) {
        String jpaQuery = "SELECT o FROM UserRole o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery, UserRole.class).getResultList();
    }

    public static UserRole UserRole.findUserRole(Long id) {
        if (id == null) return null;
        return entityManager().find(UserRole.class, id);
    }

    public static List<UserRole> UserRole.findUserRoleEntries(int firstResult, int maxResults) {
        return entityManager().createQuery("SELECT o FROM UserRole o",
UserRole.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    public static List<UserRole> UserRole.findUserRoleEntries(int firstResult, int maxResults,
String sortFieldName, String sortOrder) {
        String jpaQuery = "SELECT o FROM UserRole o";
        if (fieldNames4OrderClauseFilter.contains(sortFieldName)) {
            jpaQuery = jpaQuery + " ORDER BY " + sortFieldName;
            if ("ASC".equalsIgnoreCase(sortOrder) || "DESC".equalsIgnoreCase(sortOrder)) {
                jpaQuery = jpaQuery + " " + sortOrder;
            }
        }
        return entityManager().createQuery(jpaQuery,
UserRole.class).setFirstResult(firstResult).setMaxResults(maxResults).getResultList();
    }

    @Transactional
    public void UserRole.persist() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.persist(this);
    }

    @Transactional
    public void UserRole.remove() {
        if (this.entityManager == null) this.entityManager = entityManager();
        if (this.entityManager.contains(this)) {
            this.entityManager.remove(this);
        } else {
            UserRole attached = UserRole.findUserRole(this.id);
            this.entityManager.remove(attached);
        }
    }

    @Transactional
    public void UserRole.flush() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.flush();
    }

    @Transactional
    public void UserRole.clear() {
        if (this.entityManager == null) this.entityManager = entityManager();
        this.entityManager.clear();
    }

    @Transactional
    public UserRole UserRole.merge() {
```

```
        if (this.entityManager == null) this.entityManager = entityManager();
        UserRole merged = this.entityManager.merge(this);
        this.entityManager.flush();
        return merged;
    }

}
```

13.4.2.23 Fichero "UserRole_Roo_Jpa_Entity.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.UserRole;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Version;

privileged aspect UserRole_Roo_Jpa_Entity {

    declare @type: UserRole: @Entity;

    declare @type: UserRole: @Table(name = "users_roles");

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long UserRole.id;

    @Version
    @Column(name = "version")
    private Integer UserRole.version;

    public Long UserRole.getId() {
        return this.id;
    }

    public void UserRole.setId(Long id) {
        this.id = id;
    }

    public Integer UserRole.getVersion() {
        return this.version;
    }

    public void UserRole.setVersion(Integer version) {
        this.version = version;
    }

}
```

13.4.2.24 Fichero "UserRole_Roo_ToString.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mediaserver.domain.security.UserRole;
import org.apache.commons.lang3.builder.ReflectionToStringBuilder;
import org.apache.commons.lang3.builder.ToStringStyle;
```

```
privileged aspect UserRole_Roo_ToString {

    public String UserRole.toString() {
        return ReflectionToStringBuilder.toString(this, ToStringStyle.SHORT_PREFIX_STYLE);
    }

}
```

13.4.2.25 Fichero "UserRoleBatchService_Roo_GvNIXJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.domain.security;

import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPADeleteClause;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import com.mediaserver.domain.security.UserRole;
import com.mediaserver.domain.security.UserRoleBatchService;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import org.springframework.transaction.annotation.Transactional;

privileged aspect UserRoleBatchService_Roo_GvNIXJpaBatch {

    public Class UserRoleBatchService.getEntity() {
        return UserRole.class;
    }

    public EntityManager UserRoleBatchService.entityManager() {
        return UserRole.entityManager();
    }

    @Transactional
    public int UserRoleBatchService.deleteAll() {
        return entityManager().createQuery("DELETE FROM UserRole").executeUpdate();
    }

    @Transactional
    public int UserRoleBatchService.deleteIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM UserRole as u WHERE u.id IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public int UserRoleBatchService.deleteNotIn(List<Long> ids) {
        Query query = entityManager().createQuery("DELETE FROM UserRole as u WHERE u.id NOT IN
(:idList)");
        query.setParameter("idList", ids);
        return query.executeUpdate();
    }

    @Transactional
    public void UserRoleBatchService.create(List<UserRole> userRoles) {
        for( UserRole userrole : userRoles) {
            userrole.persist();
        }
    }

    @Transactional
    public List<UserRole> UserRoleBatchService.update(List<UserRole> userroles) {
        List<UserRole> merged = new ArrayList<UserRole>();
        for( UserRole userrole : userroles) {
            merged.add( userrole.merge() );
        }
```

```
        return merged;
    }

    public List<UserRole> UserRoleBatchService.findByValues(Map<String, Object> propertyValues)
{

        // if there is a filter
        if (propertyValues != null && !propertyValues.isEmpty()) {
            // Prepare a predicate
            BooleanBuilder baseFilterPredicate = new BooleanBuilder();

            // Base filter. Using BooleanBuilder, a cascading builder for
            // Predicate expressions
            PathBuilder<UserRole> entity = new PathBuilder<UserRole>(UserRole.class, "entity");

            // Build base filter
            for (String key : propertyValues.keySet()) {
                baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
            }

            // Create a query with filter
            JPAQuery query = new JPAQuery(UserRole.entityManager());
            query = query.from(entity);

            // execute query
            return query.where(baseFilterPredicate).list(entity);
        }

        // no filter: return all elements
        return UserRole.findAllUserRoles();
    }

    @Transactional
    public long UserRoleBatchService.deleteByValues(Map<String, Object> propertyValues) {

        // if there no is a filter
        if (propertyValues == null || propertyValues.isEmpty()) {
            throw new IllegalArgumentException("Missing property values");
        }
        // Prepare a predicate
        BooleanBuilder baseFilterPredicate = new BooleanBuilder();

        // Base filter. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<UserRole> entity = new PathBuilder<UserRole>(UserRole.class, "entity");

        // Build base filter
        for (String key : propertyValues.keySet()) {
            baseFilterPredicate.and(entity.get(key).eq(propertyValues.get(key)));
        }

        // Create a query with filter
        JPADeleteClause delete = new JPADeleteClause(UserRole.entityManager(),entity);

        // execute delete
        return delete.where(baseFilterPredicate).execute();
    }

    @Transactional
    public void UserRoleBatchService.delete(List<UserRole> userroles) {
        for( UserRole userrole : userroles) {
            userrole.remove();
        }
    }

}
```

13.4.2.26 Fichero "UserRoleBatchService.java":

```
package com.mediaserver.domain.security;
import org.gvnix.addon.jpa.batch.GvNIXJpaBatch;
import org.springframework.stereotype.Service;
```

```
@Service
@GvNIXJpaBatch(entity = UserRole.class)
public class UserRoleBatchService {
}
```

13.4.3 Paquete com.mediaserver.reference

13.4.3.1 Fichero "MediaType.java":

```
package com.mediaserver.reference;

public enum MediaType {

    Video, Audio
}
```

13.4.4 Paquete com.mediaserver.security

13.4.4.1 Fichero "JpaUserDetailService.java":

```
/*
 * Copyright 2013 Muhammad Ichsan
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package com.mediaserver.security;

import java.util.ArrayList;
import java.util.List;

import javax.persistence.NonUniqueResultException;
import javax.persistence.TypedQuery;

import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;

import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;

/**
 * Load user and his/her roles using JPA
 */
public class JpaUserDetailService implements UserDetailsService {

        @Override
        public UserDetails loadUserByUsername(String username)
                        throws UsernameNotFoundException {
                try {
                        TypedQuery<User> userQuery = User.findUsersByEmailAddress(username);
                        User targetUser = userQuery.getSingleResult();
```

```
                        List<GrantedAuthority> authorities = loadGrantedAuthorities(targetUser);

                        return new org.springframework.security.core.userdetails.User(
                                        targetUser.getEmailAddress(),
                                        targetUser.getPassword(),
                                        targetUser.getEnabled(),
                                        true, // account not expired
                                        true, // credentials not expired
                                        !targetUser.getLocked(),
                                        authorities);

                } catch (EmptyResultDataAccessException e) {
                        throw new UsernameNotFoundException("User " + username
                                        + " is not found");
                } catch (NonUniqueResultException e) {
                        throw new IllegalStateException("Non-unique user" + username
                                        + ", contact administrator");
                }
        }

        private List<GrantedAuthority> loadGrantedAuthorities(User targetUser) {
                List<GrantedAuthority> authorities = new ArrayList<GrantedAuthority>();

                TypedQuery<UserRole> roleQuery = UserRole
                                .findUserRolesByUser(targetUser);
                List<UserRole> userRoles = roleQuery.getResultList();

                for (UserRole userRole : userRoles) {
                        authorities.add(new SimpleGrantedAuthority(userRole.getRole()
                                        .getName()));
                }

                return authorities;
        }

}
```

13.4.4.2 Fichero "RoleBasedAuthenticationSuccessHandler.java":

```
package com.mediaserver.security;

import java.io.IOException;
import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;

public class RoleBasedAuthenticationSuccessHandler implements
                AuthenticationSuccessHandler {
        private Map<String, String> roleUrlMap;

        @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
                                        HttpServletResponse response,
                                        Authentication authentication) throws IOException,
ServletException {

        if (authentication.getPrincipal() instanceof UserDetails) {
            UserDetails userDetails = (UserDetails) authentication.getPrincipal();
            String role = userDetails.getAuthorities().isEmpty() ? null :
userDetails.getAuthorities().toArray()[0].toString();
            response.sendRedirect(request.getContextPath() + roleUrlMap.get(role));
        }
    }
```

```
    public void setRoleUrlMap(Map<String, String> roleUrlMap) {
        this.roleUrlMap = roleUrlMap;
    }

}
```

13.4.5 Paquete com.mediaserver.util

13.4.5.1 Fichero "EncriptionUtil.java":

```
package com.mediaserver.security;

import java.io.IOException;
import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.AuthenticationSuccessHandler;

public class RoleBasedAuthenticationSuccessHandler implements
                AuthenticationSuccessHandler {
        private Map<String, String> roleUrlMap;

        @Override
    public void onAuthenticationSuccess(HttpServletRequest request,
                                        HttpServletResponse response,
                                        Authentication authentication) throws IOException,
ServletException {

        if (authentication.getPrincipal() instanceof UserDetails) {
            UserDetails userDetails = (UserDetails) authentication.getPrincipal();
            String role = userDetails.getAuthorities().isEmpty() ? null :
userDetails.getAuthorities().toArray()[0].toString();
            response.sendRedirect(request.getContextPath() + roleUrlMap.get(role));
        }
    }

    public void setRoleUrlMap(Map<String, String> roleUrlMap) {
        this.roleUrlMap = roleUrlMap;
    }

}
```

13.4.6 Paquete com.mediaserver.web

13.4.6.1 Fichero "ApplicationConversionServiceFactoryBean.java":

```
package com.mediaserver.web;

import org.springframework.core.convert.converter.Converter;
import org.springframework.format.FormatterRegistry;
import org.springframework.format.support.FormattingConversionServiceFactoryBean;
import org.springframework.roo.addon.web.mvc.controller.converter.RooConversionService;

import com.itextpdf.text.pdf.PdfStructTreeController.returnType;
import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.User;
```

```
/**
 * A central place to register application converters and formatters.
 */
@RooConversionService
public class ApplicationConversionServiceFactoryBean extends
FormattingConversionServiceFactoryBean {

	@SuppressWarnings("deprecation")
	@Override
	protected void installFormatters(FormatterRegistry registry) {
		super.installFormatters(registry);
		// Register application converters and formatters
		registry.addConverter(getUserToStringConverter());
		registry.addConverter(getRoleToStringConverter());
		registry.addConverter(getMediaToStringConverter());
		registry.addConverter(getPlayListToStringConverter());
	}
	/**
	 * Convierte un usuario en una cadea de texo mostrable por la interfaz
	 * @return
	 */
	public Converter<User, String> getUserToStringConverter() {
		return new org.springframework.core.convert.converter.Converter<User,
String>() {
		  public String convert(User user) {
		    return new StringBuilder().append(user.getFirstName()+ "
"+user.getLastName()).toString();
		  }
		};
	}
	/**
	 * Convierte un rol  en una cadea de texo mostrable por la interfaz
	 * @return
	 */
	public Converter<Role, String> getRoleToStringConverter() {
		return new org.springframework.core.convert.converter.Converter<Role,
String>() {
		  public String convert(Role role) {
		      return role.getName();
		  }
		};
	}
	/**
	 * Convierte un medio en una cadea de texo mostrable por la interfaz
	 * @return
	 */
	public Converter<Media, String> getMediaToStringConverter() {
		return new org.springframework.core.convert.converter.Converter<Media,
String>() {
		  public String convert(Media media) {
		    //return "<b>"+media.getName()+"</b> - "+media.getSource();
		    return media.getName()+" - "+media.getSource();
		  }
		};
	}
	/**
	 * Convierte una lista de reproducción  en una cadea de texo mostrable por la interfaz
	 * @return
	 */
	public Converter<PlayList, String> getPlayListToStringConverter() {
		return new org.springframework.core.convert.converter.Converter<PlayList,
String>() {
		  public String convert(PlayList playList) {
		    return playList.getPlayListName();
		  }
		};
	}
}
```

13.4.6.2 Fichero "ApplicationConversionServiceFactoryBean_Roo_ConversionService.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;
import com.mediaserver.web.ApplicationConversionServiceFactoryBean;

import org.springframework.beans.factory.annotation.Configurable;
import org.springframework.core.convert.converter.Converter;
import org.springframework.format.FormatterRegistry;

privileged aspect ApplicationConversionServiceFactoryBean_Roo_ConversionService {

    declare @type: ApplicationConversionServiceFactoryBean: @Configurable;


    public Converter<Long, Media>
ApplicationConversionServiceFactoryBean.getIdToMediaConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.Long,
com.mediaserver.domain.Media>() {
            public com.mediaserver.domain.Media convert(java.lang.Long id) {
                return Media.findMedia(id);
            }
        };
    }

    public Converter<String, Media>
ApplicationConversionServiceFactoryBean.getStringToMediaConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.String,
com.mediaserver.domain.Media>() {
            public com.mediaserver.domain.Media convert(String id) {
                return getObject().convert(getObject().convert(id, Long.class), Media.class);
            }
        };
    }


    public Converter<Long, PlayList>
ApplicationConversionServiceFactoryBean.getIdToPlayListConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.Long,
com.mediaserver.domain.PlayList>() {
            public com.mediaserver.domain.PlayList convert(java.lang.Long id) {
                return PlayList.findPlayList(id);
            }
        };
    }

    public Converter<String, PlayList>
ApplicationConversionServiceFactoryBean.getStringToPlayListConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.String,
com.mediaserver.domain.PlayList>() {
            public com.mediaserver.domain.PlayList convert(String id) {
                return getObject().convert(getObject().convert(id, Long.class), PlayList.class);
            }
        };
    }

    public Converter<PlayListMedia, String>
ApplicationConversionServiceFactoryBean.getPlayListMediaToStringConverter() {
        return new
org.springframework.core.convert.converter.Converter<com.mediaserver.domain.PlayListMedia,
java.lang.String>() {
```

```
            public String convert(PlayListMedia playListMedia) {
                return "(no displayable fields)";
            }
        };
    }

    public Converter<Long, PlayListMedia>
ApplicationConversionServiceFactoryBean.getIdToPlayListMediaConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.Long,
com.mediaserver.domain.PlayListMedia>() {
            public com.mediaserver.domain.PlayListMedia convert(java.lang.Long id) {
                return PlayListMedia.findPlayListMedia(id);
            }
        };
    }

    public Converter<String, PlayListMedia>
ApplicationConversionServiceFactoryBean.getStringToPlayListMediaConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.String,
com.mediaserver.domain.PlayListMedia>() {
            public com.mediaserver.domain.PlayListMedia convert(String id) {
                return getObject().convert(getObject().convert(id, Long.class),
PlayListMedia.class);
            }
        };
    }


    public Converter<Long, Role> ApplicationConversionServiceFactoryBean.getIdToRoleConverter()
{
        return new org.springframework.core.convert.converter.Converter<java.lang.Long,
com.mediaserver.domain.security.Role>() {
            public com.mediaserver.domain.security.Role convert(java.lang.Long id) {
                return Role.findRole(id);
            }
        };
    }

    public Converter<String, Role>
ApplicationConversionServiceFactoryBean.getStringToRoleConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.String,
com.mediaserver.domain.security.Role>() {
            public com.mediaserver.domain.security.Role convert(String id) {
                return getObject().convert(getObject().convert(id, Long.class), Role.class);
            }
        };
    }



    public Converter<Long, User> ApplicationConversionServiceFactoryBean.getIdToUserConverter()
{
        return new org.springframework.core.convert.converter.Converter<java.lang.Long,
com.mediaserver.domain.security.User>() {
            public com.mediaserver.domain.security.User convert(java.lang.Long id) {
                return User.findUser(id);
            }
        };
    }

    public Converter<String, User>
ApplicationConversionServiceFactoryBean.getStringToUserConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.String,
com.mediaserver.domain.security.User>() {
            public com.mediaserver.domain.security.User convert(String id) {
                return getObject().convert(getObject().convert(id, Long.class), User.class);
            }
        };
    }

    public Converter<UserRole, String>
ApplicationConversionServiceFactoryBean.getUserRoleToStringConverter() {
        return new
org.springframework.core.convert.converter.Converter<com.mediaserver.domain.security.UserRole,
java.lang.String>() {
```

```
            public String convert(UserRole userRole) {
                return "(no displayable fields)";
            }
        };
    }

    public Converter<Long, UserRole>
ApplicationConversionServiceFactoryBean.getIdToUserRoleConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.Long,
com.mediaserver.domain.security.UserRole>() {
            public com.mediaserver.domain.security.UserRole convert(java.lang.Long id) {
                return UserRole.findUserRole(id);
            }
        };
    }

    public Converter<String, UserRole>
ApplicationConversionServiceFactoryBean.getStringToUserRoleConverter() {
        return new org.springframework.core.convert.converter.Converter<java.lang.String,
com.mediaserver.domain.security.UserRole>() {
            public com.mediaserver.domain.security.UserRole convert(String id) {
                return getObject().convert(getObject().convert(id, Long.class), UserRole.class);
            }
        };
    }

    public void ApplicationConversionServiceFactoryBean.installLabelConverters(FormatterRegistry
registry) {
        registry.addConverter(getIdToMediaConverter());
        registry.addConverter(getStringToMediaConverter());
        registry.addConverter(getIdToPlayListConverter());
        registry.addConverter(getStringToPlayListConverter());
        registry.addConverter(getPlayListMediaToStringConverter());
        registry.addConverter(getIdToPlayListMediaConverter());
        registry.addConverter(getStringToPlayListMediaConverter());
        registry.addConverter(getRoleToStringConverter());
        registry.addConverter(getIdToRoleConverter());
        registry.addConverter(getStringToRoleConverter());
        registry.addConverter(getIdToUserConverter());
        registry.addConverter(getStringToUserConverter());
        registry.addConverter(getIdToUserRoleConverter());
        registry.addConverter(getStringToUserRoleConverter());
    }

    public void ApplicationConversionServiceFactoryBean.afterPropertiesSet() {
        super.afterPropertiesSet();
        installLabelConverters(getObject());
    }

}
```

13.4.6.3 Fichero "MediaController.java":

```
package com.mediaserver.web;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.gvnix.addon.datatables.GvNIXDatatables;
import org.gvnix.addon.web.mvc.batch.GvNIXWebJpaBatch;
import org.gvnix.addon.web.mvc.jquery.GvNIXWebJQuery;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.json.JsonResponse;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.roo.addon.web.mvc.controller.finder.RooWebFinder;
import org.springframework.roo.addon.web.mvc.controller.scaffold.RooWebScaffold;
```

```
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.Media;
import com.mediaserver.domain.MediaBatchService;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.security.User;
import com.mediaserver.reference.MediaType;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.types.path.PathBuilder;

/**
 * Controlador de la clase Media
 * @author UO180258
 *
 */
@RequestMapping("/medias")
@Controller
@RooWebScaffold(path = "medias", formBackingObject = Media.class)
@RooWebFinder
@GvNIXWebJQuery
@GvNIXWebJpaBatch(service = MediaBatchService.class)
@GvNIXDatatables(ajax = true, inlineEditing = true)
public class MediaController {

        public Map<String, Object> getPropertyMap(Media media, HttpServletRequest request) {
        //        método generado incorrectamente
        return null;
    }

        /**
         * Produce una vista con el listado de medios
         * @param page número de pagnia si es necesario paginar los resultados
         * @param sizetamaño
         * @param sortFieldName Nombre del campo por el que se ordenan las filas mostradas
         * @param sortOrder Orden ascenente o descendente
         * @param uiModel       modelos de interfaz
         * @return
         */
        @RequestMapping(produces = "text/html")
    public String list(
                @RequestParam(value = "page", required = false) Integer page,
                @RequestParam(value = "size", required = false) Integer size,
                @RequestParam(value = "sortFieldName", required = false) String sortFieldName,
                @RequestParam(value = "sortOrder", required = false) String sortOrder,
                Model uiModel) {
                        //Obtenemos el nombre de su rol
                        String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                        // si usuario administrador
                        if (roleString.equals("ADMINISTRATOR")) {
                                if (page != null || size != null) {
                            int sizeNo = size == null ? 10 : size.intValue();
                            final int firstResult = page == null ? 0 : (page.intValue() - 1) *
sizeNo;

                                uiModel.addAttribute("medias", Media.findMediaEntries(firstResult,
sizeNo, sortFieldName, sortOrder));
                                float nrOfPages = (float) Media.countMedias() / sizeNo;
```

```
                            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages
|| nrOfPages == 0.0) ? nrOfPages + 1 : nrOfPages));
                        } else {
                            uiModel.addAttribute("medias", Media.findAllMedias(sortFieldName,
sortOrder));
                        }
                                return "medias/list";
                        }
                        //mostramos los medios de los que el usuario es owner
                        User user=User.getContextUser();
                        if (user!=null) {// si no está vacía
                                List<Media> resultList =
Media.findMediasByOwner(user).getResultList();// buscamos los medios del usuario
                                uiModel.addAttribute("medias", resultList);
                        }
                        return "medias/list";

    }

        /**
         * Crea un nuevo medio en el sistema y devuelve la vista del elemento creado
         * @param media
         * @param bindingResult
         * @param uiModel
         * @param httpServletRequest
         * @return
         */
        @RequestMapping(method = RequestMethod.POST, produces = "text/html")
    public String create(@Valid Media media, BindingResult bindingResult, Model uiModel,
HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, media);
            return "medias/create";
        }
        uiModel.asMap().clear();
        media.setOwner(User.getContextUser());
        media.persist();
        return "redirect:/medias/" + encodeUrlPathSegment(media.getId().toString(),
httpServletRequest);
    }

        /**
         * Encuentra los medios de la aplicación, todos en caso de que el usuario del contexto
sea administrador y solo los del
         * usuario concreto en otro caso
         * @param criterias
         * @param media
         * @param request
         * @return
         */
        @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax",
produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>> findAllMedias(@DatatablesParams
DatatablesCriterias criterias, @ModelAttribute Media media, HttpServletRequest request) {
        // URL parameters are used as base search filters
        Map<String, Object> baseSearchValuesMap = getPropertyMap(media, request);

        setDatatablesBaseFilter(baseSearchValuesMap);
      //Obtenemos el nombre de su rol
                String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                long totalRecords;
                long recordsFound;
                String pkFieldName = "id";
                DataSet<Map<String, String>> dataSet;
                // si usuario administrador
                if (roleString.equals("ADMINISTRATOR")) {
                SearchResults<Media> searchResult = DatatablesUtils.findByCriteria(Media.class,
Media.entityManager(), criterias, baseSearchValuesMap, conversionService_dtt,
messageSource_dtt);
                // Get datatables required counts
                totalRecords = searchResult.getTotalCount();
                recordsFound = searchResult.getResultsCount();
```

```
                // Entity pk field name
                dataSet = DatatablesUtils.populateDataSet(searchResult.getResults(),
pkFieldName, totalRecords, recordsFound, criterias.getColumnDefs(), null,
conversionService_dtt);
                return DatatablesResponse.build(dataSet,criterias);
                }else{
                        BooleanBuilder baseSearch = new BooleanBuilder();
                // Base Search. Using BooleanBuilder, a cascading builder for
                // Predicate expressions
                PathBuilder<Media> entity = new PathBuilder<Media>(Media.class, "entity");
                baseSearch.and(entity.get("owner").eq(User.getContextUser()));
                SearchResults<Media> searchResult = DatatablesUtils.findByCriteria(entity,
Media.entityManager(), criterias, baseSearch);

                // Get datatables required counts
                totalRecords = searchResult.getTotalCount();
                recordsFound = searchResult.getResultsCount();
                dataSet = DatatablesUtils.populateDataSet(searchResult.getResults(),
pkFieldName, totalRecords, recordsFound, criterias.getColumnDefs(), null,
conversionService_dtt);
                return DatatablesResponse.build(dataSet,criterias);

                }

    }


        /**
         * Creción AJAX de nuevos medios. Se añaden a la lista de medios y se les asigna el
usuario
         * que solicitó su creación como propietario. Devuelve una respuesta jon para mostrar
los resultados sin necesidad de
         * recargar la página.
         * @param medias
         * @param bindingResult
         * @param request
         * @return
         */
        @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.POST, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<Media>> createBatch(@RequestBody @Valid List<Media> medias,
BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<Media>> jsonResponse = new JsonResponse<List<Media>>();
        jsonResponse.setValue(medias);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        for(Media m:medias){
        m.setOwner(User.getContextUser());
        }
        try {
            batchService.create(medias);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setOid(getOIDList(medias));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

        /**
         * Modifica un medio del sistema
         * @param media
         * @param bindingResult
         * @param uiModel
         * @param httpServletRequest
         * @return
         */
    @RequestMapping(method = RequestMethod.PUT, produces = "text/html")
```

```
    public String update(@Valid Media media, BindingResult bindingResult, Model uiModel,
HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, media);
            return "medias/update";
        }
        uiModel.asMap().clear();
        media.setOwner(User.getContextUser());
        media.merge();
        return "redirect:/medias/" + encodeUrlPathSegment(media.getId().toString(),
httpServletRequest);
    }

    /**
     * Actualiza mediante petición AJAx un medio y devuelve el resultado para mostrar sin
necesidad de actualizar la página
     * @param medias
     * @param bindingResult
     * @param request
     * @return
     */
    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.PUT, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<Media>> updateBatch(@RequestBody @Valid List<Media> medias,
BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<Media>> jsonResponse = new JsonResponse<List<Media>>();
        jsonResponse.setValue(medias);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
        for(Media m:medias){
                User user=Media.findMedia(m.getId()).getOwner();
                m.setOwner(user);
        }
            medias = batchService.update(medias);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setValue(medias);
        jsonResponse.setOid(getOIDList(medias));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }
    /**
     * Elimina un medio mediante una petición AJAX y devuelve el resultado para mostrarlo sin
necisidad de recargar la página
     * @param all
     * @param deleteIn
     * @param idList
     * @param media
     * @param request
     * @return
     */
          @RequestMapping(value = "/delete", produces = "application/json", method =
RequestMethod.POST)
            public ResponseEntity<Object> deleteBatch(@RequestParam(value = "all", required =
false) boolean all, @RequestParam(value = "deleteIn", required = false) Boolean deleteIn,
@RequestParam(value = "idList[]", required = false) List<Long> idList, @ModelAttribute Media
media, WebRequest request) {
                HttpHeaders headers = new HttpHeaders();
                headers.add("Content-Type", "application/json");
                long count = 0;
                try {
                    if (all) {
                        Map<String, Object> baseFilter =
getRequestPropertyValues(media,request.getParameterNames());
                        if (baseFilter == null || baseFilter.isEmpty()) {
                            count = batchService.deleteAll();
```

```
                        } else {
                            count = batchService.deleteByValues(baseFilter);
                        }
                    } else {
                        if (idList == null) {
                            throw new IllegalArgumentException("Missing request parameter
'idList[]'");
                        }
                        if (!idList.isEmpty()) {
                            if (deleteIn) {
                                for(Long id:idList){
                                Media m=Media.findMedia(id);
                                unBind(m);
                                }
                                count = batchService.deleteIn(idList);
                            } else {
                                count = batchService.deleteNotIn(idList);
                            }
                        }
                    }
                } catch (RuntimeException e) {
                    LOGGER_BATCH.error("error deleting selection", e);
                    return new ResponseEntity<Object>(e, headers,
HttpStatus.INTERNAL_SERVER_ERROR);
                }
                LOGGER_BATCH.debug("deleted: " + count);
                return new ResponseEntity<Object>(count, headers, HttpStatus.OK);
            }


        private void unBind(Media media) {
                List<PlayListMedia>
list=PlayListMedia.findPlayListMediasByMedia(media).getResultList();
                for (PlayListMedia plm:list){
                        plm.remove();
                }
        }

        /**
         * Encuenta los medios del sistema por tipo, todas para administradores, solo las del
usuario en otro caso
         * @param criterias
         * @param type
         * @return
         */
        @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", params
= "ajax_find=ByType", produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>> findMediasByType(@DatatablesParams
DatatablesCriterias criterias, @RequestParam("type") MediaType type) {
        BooleanBuilder baseSearch = new BooleanBuilder();

        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<Media> entity = new PathBuilder<Media>(Media.class, "entity");

        baseSearch.and(entity.get("type").eq(type));
        String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                if (!(roleString.equals("ADMINISTRATOR"))){
                        baseSearch.and(entity.get("owner").eq(User.getContextUser()));
                }
        SearchResults<Media> searchResult = DatatablesUtils.~~findByCriteria~~(entity,
Media.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";
```

```
        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }
}
```

13.4.6.4 Fichero "MediaController_Roo_Controller_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.Media;
import com.mediaserver.reference.MediaType;
import com.mediaserver.web.MediaController;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

privileged aspect MediaController_Roo_Controller_Finder {

    @RequestMapping(params = { "find=ByType", "form" }, method = RequestMethod.GET)
    public String MediaController.findMediasByTypeForm(Model uiModel) {
        uiModel.addAttribute("mediatypes",
java.util.Arrays.asList(MediaType.class.getEnumConstants()));
        return "medias/findMediasByType";
    }

    @RequestMapping(params = "find=ByType", method = RequestMethod.GET)
    public String MediaController.findMediasByType(@RequestParam("type") MediaType type,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, @RequestParam(value = "sortFieldName", required = false) String
sortFieldName, @RequestParam(value = "sortOrder", required = false) String sortOrder, Model
uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("medias", Media.findMediasByType(type, sortFieldName,
sortOrder).setFirstResult(firstResult).setMaxResults(sizeNo).getResultList());
            float nrOfPages = (float) Media.countFindMediasByType(type) / sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("medias", Media.findMediasByType(type, sortFieldName,
sortOrder).getResultList());
        }
        return "medias/list";
    }

}
```

13.4.6.5 Fichero "MediaController_Roo_Controller":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.security.User;
import com.mediaserver.reference.MediaType;
import com.mediaserver.web.MediaController;

import java.io.UnsupportedEncodingException;
```

```
import java.util.Arrays;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

privileged aspect MediaController_Roo_Controller {



    @RequestMapping(params = "form", produces = "text/html")
    public String MediaController.createForm(Model uiModel) {
        populateEditForm(uiModel, new Media());
        return "medias/create";
    }

    @RequestMapping(value = "/{id}", produces = "text/html")
    public String MediaController.show(@PathVariable("id") Long id, Model uiModel) {
        uiModel.addAttribute("media", Media.findMedia(id));
        uiModel.addAttribute("itemId", id);
        return "medias/show";
    }




    @RequestMapping(value = "/{id}", params = "form", produces = "text/html")
    public String MediaController.updateForm(@PathVariable("id") Long id, Model uiModel) {
        populateEditForm(uiModel, Media.findMedia(id));
        return "medias/update";
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE, produces = "text/html")
    public String MediaController.delete(@PathVariable("id") Long id, @RequestParam(value =
"page", required = false) Integer page, @RequestParam(value = "size", required = false) Integer
size, Model uiModel) {
        Media media = Media.findMedia(id);
        media.remove();
        uiModel.asMap().clear();
        uiModel.addAttribute("page", (page == null) ? "1" : page.toString());
        uiModel.addAttribute("size", (size == null) ? "10" : size.toString());
        return "redirect:/medias";
    }

    void MediaController.populateEditForm(Model uiModel, Media media) {
        uiModel.addAttribute("media", media);
        uiModel.addAttribute("mediatypes", Arrays.asList(MediaType.values()));
    }

    String MediaController.encodeUrlPathSegment(String pathSegment, HttpServletRequest
httpServletRequest) {
        String enc = httpServletRequest.getCharacterEncoding();
        if (enc == null) {
            enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
        }
        try {
            pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
        } catch (UnsupportedEncodingException uee) {}
        return pathSegment;
    }

}
```

13.4.6.6 Fichero "MediaController_Roo_GvNIXDatatables.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.core.exception.ExportException;
import com.github.dandelion.datatables.core.export.CsvExport;
import com.github.dandelion.datatables.core.export.DatatablesExport;
import com.github.dandelion.datatables.core.export.ExportConf;
import com.github.dandelion.datatables.core.export.ExportType;
import com.github.dandelion.datatables.core.export.ExportUtils;
import com.github.dandelion.datatables.core.export.XmlExport;
import com.github.dandelion.datatables.core.html.HtmlTable;
import com.github.dandelion.datatables.extras.export.itext.PdfExport;
import com.github.dandelion.datatables.extras.export.poi.XlsExport;
import com.github.dandelion.datatables.extras.export.poi.XlsxExport;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.Media;
import com.mediaserver.domain.security.User;
import com.mediaserver.reference.MediaType;
import com.mediaserver.web.MediaController;
import com.mediaserver.web.MediaController_Roo_Controller;
import com.mediaserver.web.MediaController_Roo_GvNIXDatatables;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;

import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletResponseWrapper;
import javax.validation.Valid;

import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.ArrayUtils;
import org.apache.commons.lang3.StringUtils;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.validation.BeanPropertyBindingResult;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
```

```
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

privileged aspect MediaController_Roo_GvNIXDatatables {

    declare precedence: MediaController_Roo_GvNIXDatatables, MediaController_Roo_Controller;

    @Autowired
    public ConversionService MediaController.conversionService_dtt;

    @Autowired
    public MessageSource MediaController.messageSource_dtt;

    public BeanWrapper MediaController.beanWrapper;

    @RequestMapping(method = RequestMethod.GET, produces = "text/html")
    public String MediaController.listDatatables(Model uiModel, HttpServletRequest request) {
        Map<String, String> params = populateParametersMap(request);
        // Get parentId information for details render
        String parentId = params.remove("_dt_parentId");
        if (StringUtils.isNotBlank(parentId)) {
            uiModel.addAttribute("parentId", parentId);
        }
        String rowOnTopIds = params.remove("dtt_row_on_top_ids");
        if (StringUtils.isNotBlank(rowOnTopIds)) {
            uiModel.addAttribute("dtt_row_on_top_ids", rowOnTopIds);
        }
        String tableHashId = params.remove("dtt_parent_table_id_hash");
        if (StringUtils.isNotBlank(tableHashId)) {
            uiModel.addAttribute("dtt_parent_table_id_hash", tableHashId);
        }
        if (!params.isEmpty()) {
            uiModel.addAttribute("baseFilter", params);
        }
        return "medias/list";
    }

    @ModelAttribute
    public void MediaController.populateDatatablesConfig(Model uiModel) {
        uiModel.addAttribute("datatablesHasBatchSupport", true);
        uiModel.addAttribute("datatablesUseAjax",true);
        uiModel.addAttribute("datatablesInlineEditing",true);
        uiModel.addAttribute("datatablesInlineCreating",true);
        uiModel.addAttribute("datatablesSecurityApplied",false);
        uiModel.addAttribute("datatablesStandardMode",true);
        uiModel.addAttribute("finderNameParam","ajax_find");
    }



    public Map<String, String> MediaController.populateParametersMap(HttpServletRequest request)
{
        Map<String, Object> params;
        if (request == null) {
            params = Collections.emptyMap();
        } else {
            params = new HashMap<String, Object>(request.getParameterMap());
        }

        Map<String, String> allParams = new HashMap<String, String>(params.size());

        String value;
        Object objValue;
        for (String key : params.keySet()) {
            objValue = params.get(key);
            if (objValue instanceof String[]) {
                value = ((String[]) objValue)[0];
            } else {
                value = (String) objValue;
            }
            allParams.put(key, value);
        }
        return allParams;
```

```
    }

    public Map<String, Object> MediaController.getPropertyMap(Media media,
Enumeration<Map<String, String>> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(media == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(media);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }


    public void MediaController.setDatatablesBaseFilter(Map<String, Object> propertyMap) {
        // Add here your baseFilters to propertyMap.
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "getColumnType")
    public String MediaController.getColumnType(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_columnName_", required = false) String columnName) {
        // Getting all declared fields
        boolean fieldExists = false;
        Field attr = null;
        for(Field field : Media.class.getDeclaredFields()){
            if(field.getName().equals(columnName)){
                attr = field;
                fieldExists = true;
                break;
            }
        }
        // If current field not exists on entity, find on superclass
        if(!fieldExists){
            if(Media.class.getSuperclass() != null){
                for(Field field : Media.class.getSuperclass().getDeclaredFields()){
                    if(field.getName().equals(columnName)){
                        attr = field;
                        fieldExists = true;
                        break;
                    }
                }
            }
        }
        if(fieldExists){
            // Getting field type
            Object fieldType = null;
            if (attr != null) {
                fieldType = attr.getType();
                String type = fieldType.toString();
                // Returning value based on type
                if ("String".equals(type)){
```

```
                    return "{\"columnType\": \"string\"}";
                } else if ("float".equals(type) || type.contains("Float")){
                    return "{\"columnType\": \"number\"}";
                } else if ("short".equals(type) || type.contains("Short")){
                    return "{\"columnType\": \"number\"}";
                } else if ("long".equals(type) || type.contains("Long")){
                    return "{\"columnType\": \"number\"}";
                } else if ("double".equals(type) || type.contains("Double")){
                    return "{\"columnType\": \"number\"}";
                } else if ("int".equals(type) || type.contains("Integer")){
                    return "{\"columnType\": \"number\"}";
                } else if ("Date".equals(type)){
                    return "{\"columnType\": \"date\"}";
                } else if ("boolean".equals(type) || type.contains("Boolean")){
                    return "{\"columnType\": \"boolean\"}";
                } else {
                    // Returning by default
                    return "{\"columnType\": \"undefined\"}";
                }
            }
        }
        // Returning by default
        return "{\"columnType\": \"undefined\"}";
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "geti18nText")
    public String MediaController.geti18nText(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_locale_", required = false) String locale) {
        // Getting current locale
        Locale defaultLocale = new Locale(locale);
        // Building JSON response
        StringBuilder json = new StringBuilder();
        json.append("\"all_isnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.isnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"all_notnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.notnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_false\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.false", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_true\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.true", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_date\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.date", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_day\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.day", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"date_month\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.month", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_year\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.year", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"number_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.number.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_contains\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.contains", null, defaultLocale) +
"\"");
```

```
        json.append(",");
        json.append("\"string_ends\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.ends", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isempty", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isnotempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isnotempty", null, defaultLocale)
+ "\"");
        json.append(",");
        json.append("\"string_starts\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.starts", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"button_find\": \"" + messageSource_dtt.getMessage("button_find", null,
defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_isnull\": \"" + messageSource_dtt.getMessage("help.all.isnull",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_notnull\": \"" +
messageSource_dtt.getMessage("help.all.notnull", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_false\": \"" +
messageSource_dtt.getMessage("help.boolean.false", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_true\": \"" +
messageSource_dtt.getMessage("help.boolean.true", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_between\": \"" +
messageSource_dtt.getMessage("help.date.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_date\": \"" + messageSource_dtt.getMessage("help.date.date",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_day\": \"" + messageSource_dtt.getMessage("help.date.day",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_month\": \"" + messageSource_dtt.getMessage("help.date.month",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_year\": \"" + messageSource_dtt.getMessage("help.date.year",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_between\": \"" +
messageSource_dtt.getMessage("help.number.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_eq\": \"" + messageSource_dtt.getMessage("help.number.eq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_neq\": \"" + messageSource_dtt.getMessage("help.number.neq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_gt\": \"" + messageSource_dtt.getMessage("help.number.gt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_lt\": \"" + messageSource_dtt.getMessage("help.number.lt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_goe\": \"" + messageSource_dtt.getMessage("help.number.goe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_loe\": \"" + messageSource_dtt.getMessage("help.number.loe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_contains\": \"" +
messageSource_dtt.getMessage("help.string.contains", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_ends\": \"" +
messageSource_dtt.getMessage("help.string.ends", null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"help_string_isempty\": \"" +
messageSource_dtt.getMessage("help.string.isempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isnotempty\": \"" +
messageSource_dtt.getMessage("help.string.isnotempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_starts\": \"" +
messageSource_dtt.getMessage("help.string.starts", null, defaultLocale) + "\"");
        json.append("}");
        // return JSON with locale strings
        return json.toString();
    }

    @RequestMapping(produces = "text/html", value = "/list")
    public String MediaController.listDatatablesDetail(Model uiModel, HttpServletRequest
request, @ModelAttribute Media media) {
        // Do common datatables operations: get entity list filtered by request parameters
        listDatatables(uiModel, request);
        // Show only the list fragment (without footer, header, menu, etc.)
        return "forward:/WEB-INF/views/medias/list.jspx";
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.POST, params =
"datatablesRedirect")
    public String MediaController.createDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid Media media, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common create operations (check errors, populate, persist, ...)
        String view = create(media, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common create error view (remain in create
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM, media.getId());
        // If create success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.PUT, params =
"datatablesRedirect")
    public String MediaController.updateDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid Media media, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common update operations (check errors, populate, merge, ...)
        String view = update(media, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common update error view (remain in update
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM, media.getId());
        // If update success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.DELETE, params =
"datatablesRedirect", value = "/{id}")
    public String MediaController.deleteDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @PathVariable("id") Long id,
```

```
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        // Do common delete operations (find, remove, add pagination attributes, ...)
        String view = delete(id, page, size, uiModel);
        // If no redirect, return common list view
        if (redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        // Redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }



    @RequestMapping(headers = "Accept=application/json", params = "checkFilters")
    @ResponseBody
    public ResponseEntity<String> MediaController.checkFilterExpressions(WebRequest request,
@RequestParam(value = "property", required = false) String property, @RequestParam(value =
"expression", required = false) String expression) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json; charset=utf-8");
        if(beanWrapper == null){
            beanWrapper = new BeanWrapperImpl(Media.class);
        }
        Class type = beanWrapper.getPropertyType(property);
        boolean response = DatatablesUtils.checkFilterExpressions(type,expression,
messageSource_dtt);
        return new ResponseEntity<String>(String.format("{ \"response\": %s, \"property\":
\"%s\"}",response, property), headers, org.springframework.http.HttpStatus.OK);
    }



    @RequestMapping(value = "/exportcsv", produces = "text/csv")
    public void MediaController.exportCsv(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Media media, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, media, ExportType.CSV, new CsvExport(), request, response);
    }

    @RequestMapping(value = "/exportpdf", produces = "text/pdf")
    public void MediaController.exportPdf(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Media media, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, media, ExportType.PDF, new PdfExport(), request, response);
    }

    @RequestMapping(value = "/exportxls", produces = "text/xls")
    public void MediaController.exportXls(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Media media, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, media, ExportType.XLS, new XlsExport(), request, response);
    }

    @RequestMapping(value = "/exportxlsx", produces = "text/xlsx")
    public void MediaController.exportXlsx(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Media media, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, media, ExportType.XLSX, new XlsxExport(), request, response);
    }

    @RequestMapping(value = "/exportxml", produces = "text/xml")
    public void MediaController.exportXml(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Media media, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, media, ExportType.XML, new XmlExport(), request, response);
    }

    private void MediaController.export(DatatablesCriterias criterias, Media media, ExportType
exportType, DatatablesExport datatablesExport, HttpServletRequest request, HttpServletResponse
response) throws ExportException {
        // Does the export process as is explained in
http://dandelion.github.io/datatables/tutorials/export/controller-based-exports.html
        // 1. Retrieve the data
        List<Map<String, String>> data = retrieveData(criterias, media, request);
```

```
        // 2. Build an instance of "ExportConf"
        ExportConf exportConf = new
ExportConf.Builder(exportType).header(true).exportClass(datatablesExport).autoSize(true).fileNam
e(media.getClass().getSimpleName()).build();
        // 3. Build an instance of "HtmlTable"
        HtmlTable table = DatatablesUtils.makeHtmlTable(data, criterias, exportConf, request);
        // 4. Render the generated export file
        ExportUtils.renderExport(table, exportConf, response);
    }

    private List<Map<String, String>> MediaController.retrieveData(DatatablesCriterias
criterias, Media media, HttpServletRequest request) {
        // Cloned criteria in order to not paginate the results
        DatatablesCriterias noPaginationCriteria = new
DatatablesCriterias(criterias.getSearch(), 0, null, criterias.getColumnDefs(),
criterias.getSortingColumnDefs(), criterias.getInternalCounter());
        // Do the search to obtain the data
        Map<String, Object> baseSearchValuesMap = getPropertyMap(media, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        org.gvnix.web.datatables.query.SearchResults<com.mediaserver.domain.Media> searchResult
= DatatablesUtils.findByCriteria(Media.class, Media.entityManager(), noPaginationCriteria,
baseSearchValuesMap);
        // Use ConversionService with the obtained data
        return DatatablesUtils.populateDataSet(searchResult.getResults(), "id",
searchResult.getTotalCount(), searchResult.getResultsCount(), criterias.getColumnDefs(), null,
conversionService_dtt).getRows();
    }

    @RequestMapping(value = "/datatables/createform", produces = "application/json", headers =
"Accept=application/json")
    @ResponseBody
    public List<Map<String, String>> MediaController.createJsonForm(HttpServletRequest request,
HttpServletResponse response, Model uiModel) throws ServletException, IOException {

        // Prepare result
        List<Map<String, String>> result = new ArrayList<Map<String, String>>();
        String controllerPath = "medias";
        String pageToUse = "create";
        String renderUrl = String.format("/WEB-INF/views/%s/%s.jspx", controllerPath,
pageToUse);

        Map<String, String> item = new HashMap<String, String>();
        final StringWriter buffer = new StringWriter();

        // Call JSP to render update form
        RequestDispatcher dispatcher = request.getRequestDispatcher(renderUrl);

        // spring from:input tag uses BindingResult to locate property editors
        // for each bean property. So, we add a request attribute (required key
        // id BindingResult.MODEL_KEY_PREFIX + object name) with a correctly
        // initialized bindingResult.
        Media media = new Media();
        BeanPropertyBindingResult bindingResult = new BeanPropertyBindingResult(media, "media");
        bindingResult.initConversion(conversionService_dtt);
        request.setAttribute(BindingResult.MODEL_KEY_PREFIX + "media", bindingResult);

        populateItemForRender(request, media, true);

        dispatcher.include(request, new HttpServletResponseWrapper(response) {

            private PrintWriter writer = new PrintWriter(buffer);

            @Override
            public PrintWriter getWriter() throws IOException {
                return writer;
            }
        });
        String render = buffer.toString();

        // Put rendered content into first column
        item.put("form", render);
        result.add(item);

        return result;
    }
```

```
    @RequestMapping(value = "/datatables/updateforms", produces = "application/json", headers =
"Accept=application/json")
    @ResponseBody
    public List<Map<String, String>> MediaController.updateJsonForms(@RequestParam("id") Long[]
ids, HttpServletRequest request, HttpServletResponse response, Model uiModel) throws
ServletException, IOException {
        if (ArrayUtils.isEmpty(ids)) {
            return new ArrayList<Map<String, String>>();
        }

        // Using PathBuilder, a cascading builder for Predicate expressions
        PathBuilder entity = new PathBuilder(Media.class, "entity");
        // URL parameters are used as base search filters
        Set set = new HashSet();
        set.addAll(Arrays.asList(ids));
        BooleanBuilder filterBy = QuerydslUtils.createPredicateByIn(entity, "id", set);
        // Create a query with filter
        JPAQuery query = new JPAQuery(Media.entityManager());
        query = query.from(entity);
        // execute query
        List<Media> medias = query.where(filterBy).list(entity);
        List<Map<String, String>> udpateForms = renderUpdateForm(medias, request, response);
        return udpateForms;
    }

    public List<Map<String, String>> MediaController.renderUpdateForm(List<Media> medias,
HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // Prepare result
        List<Map<String, String>> result = new ArrayList<Map<String, String>>(medias.size());
        String controllerPath = "medias";
        String pageToUse = "update";
        String renderUrl = String.format("/WEB-INF/views/%s/%s.jspx", controllerPath,
pageToUse);
        // For every element
        for (Media media : medias) {
            Map<String, String> item = new HashMap<String, String>();
            final StringWriter buffer = new StringWriter();
            // Call JSP to render update form
            RequestDispatcher dispatcher = request.getRequestDispatcher(renderUrl);
            populateItemForRender(request, media, true);
            dispatcher.include(request, new HttpServletResponseWrapper(response) {

                private PrintWriter writer = new PrintWriter(buffer);

                @Override
                public PrintWriter getWriter() throws IOException {
                    return writer;
                }
            });
            String render = buffer.toString();
            // Load item id
            item.put("DT_RowId", conversionService_dtt.convert(media.getId(), String.class));
            // Put rendered content into first column
            item.put("form", render);
            result.add(item);
        }
        return result;
    }

    public void MediaController.populateItemForRender(HttpServletRequest request, Media media,
boolean editing) {
        org.springframework.ui.Model uiModel = new org.springframework.ui.ExtendedModelMap();

        request.setAttribute("media", media);
        request.setAttribute("itemId",
conversionService_dtt.convert(media.getId(),String.class));

        if (editing) {
            // spring from:input tag uses BindingResult to locate property editors for each bean
            // property. So, we add a request attribute (required key id
BindingResult.MODEL_KEY_PREFIX + object name)
            // with a correctly initialized bindingResult.
            BeanPropertyBindingResult bindindResult = new BeanPropertyBindingResult(media,
"media");
```

```
            bindindResult.initConversion(conversionService_dtt);
            request.setAttribute(BindingResult.MODEL_KEY_PREFIX + "media",bindindResult);
            // Add date time patterns and enums to populate inputs
            populateEditForm(uiModel, media);
        }

        // Load uiModel attributes into request
        Map<String, Object> modelMap = uiModel.asMap();
        for (String key : modelMap.keySet()){
            request.setAttribute(key, modelMap.get(key));
        }
    }

}
```

13.4.6.7 Fichero "MediaController_Roo_GvNIXWebJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.MediaBatchService;
import com.mediaserver.web.MediaController;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.gvnix.web.json.JsonResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

privileged aspect MediaController_Roo_GvNIXWebJpaBatch {

    @Autowired
    public ConversionService MediaController.conversionService_batch;

    public static Logger MediaController.LOGGER_BATCH =
LoggerFactory.getLogger(MediaController.class);;

    @Autowired
    public MediaBatchService MediaController.batchService;




    public List<String> MediaController.getOIDList(List<Media> medias) {
        List<String> result = new ArrayList<String>(medias.size());
        for (Media media : medias) {
            result.add(conversionService_batch.convert(media.getId(), String.class));
        }
```

```
        return result;
    }

    public Map<String, Object> MediaController.getRequestPropertyValues(Media media,
Iterator<String> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(media == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(media);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }

}
```

13.4.6.8 Fichero "PlayListController.java":

```
package com.mediaserver.web;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.gvnix.addon.datatables.GvNIXDatatables;
import org.gvnix.addon.web.mvc.batch.GvNIXWebJpaBatch;
import org.gvnix.addon.web.mvc.jquery.GvNIXWebJQuery;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.json.JsonResponse;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.roo.addon.web.mvc.controller.finder.RooWebFinder;
import org.springframework.roo.addon.web.mvc.controller.scaffold.RooWebScaffold;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
```

```
import org.springframework.web.context.request.WebRequest;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListBatchService;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.security.User;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.types.path.PathBuilder;

/**
 * Controlador de la clase PlayList
 */
@RequestMapping("/playlists")
@Controller
@RooWebScaffold(path = "playlists", formBackingObject = PlayList.class)
@RooWebFinder
@GvNIXWebJQuery
@GvNIXWebJpaBatch(service = PlayListBatchService.class)
@GvNIXDatatables(ajax = true, inlineEditing = true)
public class PlayListController {

    public Map<String, Object> getPropertyMap(PlayList playList, HttpServletRequest request) {
//        método generado incorrectamente
        return null;
    }

    /**
     * Lista las PlayList de la aplicación, todas para administrador, y solo las del usuario en
otro caso
     * @param page
     * @param size
     * @param sortFieldName
     * @param sortOrder
     * @param uiModel
     * @return
     */
    @RequestMapping(produces = "text/html")
    public String list(
                @RequestParam(value = "page", required = false) Integer page,
                @RequestParam(value = "size", required = false) Integer size,
                @RequestParam(value = "sortFieldName", required = false) String sortFieldName,
                @RequestParam(value = "sortOrder", required = false) String sortOrder,
                Model uiModel) {
                        //Obtenemos el nombre de su rol
                        String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                        // si usuario administrador
                        if (roleString.equals("ADMINISTRATOR")) {
                                if (page != null || size != null) {
                            int sizeNo = size == null ? 10 : size.intValue();
                            final int firstResult = page == null ? 0 : (page.intValue() - 1) *
sizeNo;

                            uiModel.addAttribute("playLists",
PlayList.findPlayListEntries(firstResult, sizeNo, sortFieldName, sortOrder));
                            float nrOfPages = (float) PlayList.countPlayLists() / sizeNo;
                            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages
|| nrOfPages == 0.0) ? nrOfPages + 1 : nrOfPages));
                        } else {
                            uiModel.addAttribute("playLists",
PlayList.findAllPlayLists(sortFieldName, sortOrder));
                        }
                                return "playlists/list";
                        }
                        //mostramos los medios de los que el usuario es owner
                        User user=User.getContextUser();
                        if (user!=null) {// si no está vacía
```

```
                                List<PlayList> resultList =
PlayList.findPlayListsByOwner(user).getResultList();// buscamos los medios del usuario
                                uiModel.addAttribute("playLists", resultList);
                            }
                            return "playlists/list";


    }

    /**
     * crea una nueva PlayList en el sistema
     * @param playList
     * @param bindingResult
     * @param uiModel
     * @param httpServletRequest
     * @return
     */
        @RequestMapping(method = RequestMethod.POST, produces = "text/html")
    public String create(@Valid PlayList playList, BindingResult bindingResult, Model uiModel,
HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, playList);
            return "playlists/create";
        }
        uiModel.asMap().clear();
        playList.setOwner(User.getContextUser());
        playList.persist();
        return "redirect:/playlists/" + encodeUrlPathSegment(playList.getId().toString(),
httpServletRequest);
    }


        /**
         * Busca las PlayLsit de la aplicación y las devuelve mediante AJAX para que no sea
necesario recargar la página
         * Mostrará todas para los administradores, y solo las del usuario en otro caso
         * @param criterias
         * @param playList
         * @param request
         * @return
         */
        @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax",
produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>> findAllPlayLists(@DatatablesParams
DatatablesCriterias criterias, @ModelAttribute PlayList playList, HttpServletRequest request) {
        // URL parameters are used as base search filters
        Map<String, Object> baseSearchValuesMap = getPropertyMap(playList, request);

        setDatatablesBaseFilter(baseSearchValuesMap);
      //Obtenemos el nombre de su rol
                String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                long totalRecords;
                long recordsFound;
                String pkFieldName = "id";
                DataSet<Map<String, String>> dataSet;
                // si usuario administrador
                if (roleString.equals("ADMINISTRATOR")) {
                SearchResults<PlayList> searchResult =
DatatablesUtils.findByCriteria(PlayList.class, PlayList.entityManager(), criterias,
baseSearchValuesMap, conversionService_dtt, messageSource_dtt);
                // Get datatables required counts
                totalRecords = searchResult.getTotalCount();
                recordsFound = searchResult.getResultsCount();
                // Entity pk field name
                dataSet = DatatablesUtils.populateDataSet(searchResult.getResults(),
pkFieldName, totalRecords, recordsFound, criterias.getColumnDefs(), null,
conversionService_dtt);
                return DatatablesResponse.build(dataSet,criterias);
                }else{//USER
                        BooleanBuilder baseSearch = new BooleanBuilder();
                // Base Search. Using BooleanBuilder, a cascading builder for
                // Predicate expressions
```

```
                PathBuilder<PlayList> entity = new PathBuilder<PlayList>(PlayList.class,
"entity");
                baseSearch.and(entity.get("owner").eq(User.getContextUser()));
                SearchResults<PlayList> searchResult = DatatablesUtils.findByCriteria(entity,
PlayList.entityManager(), criterias, baseSearch);

                // Get datatables required counts
                totalRecords = searchResult.getTotalCount();
                recordsFound = searchResult.getResultsCount();
                dataSet = DatatablesUtils.populateDataSet(searchResult.getResults(),
pkFieldName, totalRecords, recordsFound, criterias.getColumnDefs(), null,
conversionService_dtt);
                return DatatablesResponse.build(dataSet,criterias);

                }

    }


        /**
         * Crea una nueva PlayList en el sistema mediante petición AJAX para incorporarla al
listado completo y que se muestre sin
         * necesidad de recargar la página. asigna el usuario actual como su propietario
         * @param playLists
         * @param bindingResult
         * @param request
         * @return
         */
        @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.POST, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<PlayList>> createBatch(@RequestBody @Valid List<PlayList>
playLists, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<PlayList>> jsonResponse = new JsonResponse<List<PlayList>>();
        jsonResponse.setValue(playLists);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        for(PlayList m:playLists){
        m.setOwner(User.getContextUser());
        }
        try {
            batchService.create(playLists);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setOid(getOIDList(playLists));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }
        /**
         * Muestra una PlayList se le especifica su id como parámetro
         * @param id
         * @param uiModel
         * @return
         */

    @RequestMapping(value = "/{id}", produces = "text/html")
    public String show(@PathVariable("id") Long id, Model uiModel) {
        PlayList pl=PlayList.findPlayList(id);
        uiModel.addAttribute("playlist", pl);
        uiModel.addAttribute("itemId", id);
        List<PlayListMedia>
plMedias=PlayListMedia.findPlayListMediasByPlayList(pl).getResultList();
        uiModel.addAttribute("playlistmedias", plMedias);

        List<String> playlistURLS= new ArrayList<String>();
        for(PlayListMedia plm : plMedias){
        playlistURLS.add(plm.getMedia().getSource());
        }
```

```
        ObjectMapper mapper = new ObjectMapper();
        String jsonlist = "";
        try {
            jsonlist = mapper.writeValueAsString(playlistURLS);
        } catch (Exception e) {
            e.printStackTrace();
        }
        uiModel.addAttribute("playlistURLS",jsonlist);
        return "playlists/show";
    }

    /**
     * Elimina una PlayList o conjunto de ellas mediante petición AJAx para poder actualizar
     * la vista sin necesidad de recargar la página
     * @param all
     * @param deleteIn
     * @param idList
     * @param playList
     * @param request
     * @return
     */
    @RequestMapping(value = "/delete", produces = "application/json", method =
RequestMethod.POST)
    public ResponseEntity<Object> deleteBatch(@RequestParam(value = "all", required = false)
boolean all, @RequestParam(value = "deleteIn", required = false) Boolean deleteIn,
@RequestParam(value = "idList[]", required = false) List<Long> idList, @ModelAttribute PlayList
playList, WebRequest request) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json");
        long count = 0;
        try {
            if (all) {
                Map<String, Object> baseFilter =
getRequestPropertyValues(playList,request.getParameterNames());
                if (baseFilter == null || baseFilter.isEmpty()) {
                    count = batchService.deleteAll();
                } else {
                    count = batchService.deleteByValues(baseFilter);
                }
            } else {
                if (idList == null) {
                    throw new IllegalArgumentException("Missing request parameter 'idList[]'");
                }
                if (!idList.isEmpty()) {
                    if (deleteIn) {
                        for(Long id:idList){
                        PlayList pl=PlayList.findPlayList(id);
                        unBind(pl);
                        }
                        count = batchService.deleteIn(idList);
                    } else {
                        count = batchService.deleteNotIn(idList);
                    }
                }
            }
        } catch (RuntimeException e) {
            LOGGER_BATCH.error("error deleting selection", e);
            return new ResponseEntity<Object>(e, headers, HttpStatus.INTERNAL_SERVER_ERROR);
        }
        LOGGER_BATCH.debug("deleted: " + count);
        return new ResponseEntity<Object>(count, headers, HttpStatus.OK);
    }

    /**
     * Elimina los vinculos de la Playlist, todos los objetos PlayListMedia relacionados
     * @param playlist
     */
    private void unBind(PlayList playlist) {
                List<PlayListMedia>
list=PlayListMedia.findPlayListMediasByPlayList(playlist).getResultList();
                for (PlayListMedia plm:list){
                        plm.remove();
                }
        }
```

```
    /**
     * Actualiza una PlayList
     * @param playList
     * @param bindingResult
     * @param uiModel
     * @param httpServletRequest
     * @return
     */
    @RequestMapping(method = RequestMethod.PUT, produces = "text/html")
    public String update(@Valid PlayList playList, BindingResult bindingResult, Model uiModel,
HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, playList);
            return "playlists/update";
        }
        uiModel.asMap().clear();
        playList.setOwner(User.getContextUser());
        playList.merge();
        return "redirect:/playlists/" + encodeUrlPathSegment(playList.getId().toString(),
httpServletRequest);
    }

    /**
     * Actualiza una PlayList mediante petición AJAx para poder mostrar los resultados sin
necesidad de actualizar la página
     * @param playLists
     * @param bindingResult
     * @param request
     * @return
     */
    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.PUT, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<PlayList>> updateBatch(@RequestBody @Valid List<PlayList>
playLists, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<PlayList>> jsonResponse = new JsonResponse<List<PlayList>>();
        jsonResponse.setValue(playLists);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
        for(PlayList m:playLists){//findLista.owner
                User user=PlayList.findPlayList(m.getId()).getOwner();
                m.setOwner(user);
        }
            playLists = batchService.update(playLists);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setValue(playLists);
        jsonResponse.setOid(getOIDList(playLists));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }
}
```

13.4.6.9 Fichero "PlayListController_Roo_Controller_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.PlayList;
import com.mediaserver.web.PlayListController;
```

```
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

privileged aspect PlayListController_Roo_Controller_Finder {

    @RequestMapping(params = { "find=ByPlayListName", "form" }, method = RequestMethod.GET)
    public String PlayListController.findPlayListsByPlayListNameForm(Model uiModel) {
        return "playlists/findPlayListsByPlayListName";
    }

    @RequestMapping(params = "find=ByPlayListName", method = RequestMethod.GET)
    public String PlayListController.findPlayListsByPlayListName(@RequestParam("playListName")
String playListName, @RequestParam(value = "page", required = false) Integer page,
@RequestParam(value = "size", required = false) Integer size, @RequestParam(value =
"sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("playlists", PlayList.findPlayListsByPlayListName(playListName,
sortFieldName, sortOrder).setFirstResult(firstResult).setMaxResults(sizeNo).getResultList());
            float nrOfPages = (float) PlayList.countFindPlayListsByPlayListName(playListName) /
sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("playlists", PlayList.findPlayListsByPlayListName(playListName,
sortFieldName, sortOrder).getResultList());
        }
        return "playlists/list";
    }

}
```

13.4.6.10 Fichero "PlayListController_Roo_Controller.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.PlayList;
import com.mediaserver.web.PlayListController;
import java.io.UnsupportedEncodingException;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

privileged aspect PlayListController_Roo_Controller {


    @RequestMapping(params = "form", produces = "text/html")
    public String PlayListController.createForm(Model uiModel) {
        populateEditForm(uiModel, new PlayList());
        return "playlists/create";
    }


    @RequestMapping(value = "/{id}", params = "form", produces = "text/html")
    public String PlayListController.updateForm(@PathVariable("id") Long id, Model uiModel) {
        populateEditForm(uiModel, PlayList.findPlayList(id));
        return "playlists/update";
    }
```

```
    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE, produces = "text/html")
    public String PlayListController.delete(@PathVariable("id") Long id, @RequestParam(value =
"page", required = false) Integer page, @RequestParam(value = "size", required = false) Integer
size, Model uiModel) {
        PlayList playList = PlayList.findPlayList(id);
        playList.remove();
        uiModel.asMap().clear();
        uiModel.addAttribute("page", (page == null) ? "1" : page.toString());
        uiModel.addAttribute("size", (size == null) ? "10" : size.toString());
        return "redirect:/playlists";
    }

    void PlayListController.populateEditForm(Model uiModel, PlayList playList) {
        uiModel.addAttribute("playList", playList);
    }

    String PlayListController.encodeUrlPathSegment(String pathSegment, HttpServletRequest
httpServletRequest) {
        String enc = httpServletRequest.getCharacterEncoding();
        if (enc == null) {
            enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
        }
        try {
            pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
        } catch (UnsupportedEncodingException uee) {}
        return pathSegment;
    }

}
```

13.4.6.11 Fichero "PlayListController_Roo_GvNIXDatatables.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.core.exception.ExportException;
import com.github.dandelion.datatables.core.export.CsvExport;
import com.github.dandelion.datatables.core.export.DatatablesExport;
import com.github.dandelion.datatables.core.export.ExportConf;
import com.github.dandelion.datatables.core.export.ExportType;
import com.github.dandelion.datatables.core.export.ExportUtils;
import com.github.dandelion.datatables.core.export.XmlExport;
import com.github.dandelion.datatables.core.html.HtmlTable;
import com.github.dandelion.datatables.extras.export.itext.PdfExport;
import com.github.dandelion.datatables.extras.export.poi.XlsExport;
import com.github.dandelion.datatables.extras.export.poi.XlsxExport;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.PlayList;
import com.mediaserver.web.PlayListController;
import com.mediaserver.web.PlayListController_Roo_Controller;
import com.mediaserver.web.PlayListController_Roo_GvNIXDatatables;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Locale;
```

```
import java.util.Map;
import java.util.Set;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletResponseWrapper;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.ArrayUtils;
import org.apache.commons.lang3.StringUtils;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.validation.BeanPropertyBindingResult;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

privileged aspect PlayListController_Roo_GvNIXDatatables {

    declare precedence: PlayListController_Roo_GvNIXDatatables,
PlayListController_Roo_Controller;

    @Autowired
    public ConversionService PlayListController.conversionService_dtt;

    @Autowired
    public MessageSource PlayListController.messageSource_dtt;

    public BeanWrapper PlayListController.beanWrapper;

    @RequestMapping(method = RequestMethod.GET, produces = "text/html")
    public String PlayListController.listDatatables(Model uiModel, HttpServletRequest request) {
        Map<String, String> params = populateParametersMap(request);
        // Get parentId information for details render
        String parentId = params.remove("_dt_parentId");
        if (StringUtils.isNotBlank(parentId)) {
            uiModel.addAttribute("parentId", parentId);
        }
        String rowOnTopIds = params.remove("dtt_row_on_top_ids");
        if (StringUtils.isNotBlank(rowOnTopIds)) {
            uiModel.addAttribute("dtt_row_on_top_ids", rowOnTopIds);
        }
        String tableHashId = params.remove("dtt_parent_table_id_hash");
        if (StringUtils.isNotBlank(tableHashId)) {
            uiModel.addAttribute("dtt_parent_table_id_hash", tableHashId);
        }
        if (!params.isEmpty()) {
            uiModel.addAttribute("baseFilter", params);
        }
        return "playlists/list";
    }

    @ModelAttribute
    public void PlayListController.populateDatatablesConfig(Model uiModel) {
        uiModel.addAttribute("datatablesHasBatchSupport", true);
        uiModel.addAttribute("datatablesUseAjax",true);
        uiModel.addAttribute("datatablesInlineEditing",true);
        uiModel.addAttribute("datatablesInlineCreating",true);
        uiModel.addAttribute("datatablesSecurityApplied",false);
```

```
        uiModel.addAttribute("datatablesStandardMode",true);
        uiModel.addAttribute("finderNameParam","ajax_find");
    }

    public Map<String, String> PlayListController.populateParametersMap(HttpServletRequest
request) {
        Map<String, Object> params;
        if (request == null) {
            params = Collections.emptyMap();
        } else {
            params = new HashMap<String, Object>(request.getParameterMap());
        }

        Map<String, String> allParams = new HashMap<String, String>(params.size());

        String value;
        Object objValue;
        for (String key : params.keySet()) {
            objValue = params.get(key);
            if (objValue instanceof String[]) {
                value = ((String[]) objValue)[0];
            } else {
                value = (String) objValue;
            }
            allParams.put(key, value);
        }
        return allParams;
    }

    public Map<String, Object> PlayListController.getPropertyMap(PlayList playList,
Enumeration<Map<String, String>> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(playList == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(playList);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }



    public void PlayListController.setDatatablesBaseFilter(Map<String, Object> propertyMap) {
        // Add here your baseFilters to propertyMap.
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "getColumnType")
    public String PlayListController.getColumnType(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_columnName_", required = false) String columnName) {
        // Getting all declared fields
        boolean fieldExists = false;
```

```
        Field attr = null;
        for(Field field : PlayList.class.getDeclaredFields()){
            if(field.getName().equals(columnName)){
                attr = field;
                fieldExists = true;
                break;
            }
        }
        // If current field not exists on entity, find on superclass
        if(!fieldExists){
            if(PlayList.class.getSuperclass() != null){
                for(Field field : PlayList.class.getSuperclass().getDeclaredFields()){
                    if(field.getName().equals(columnName)){
                        attr = field;
                        fieldExists = true;
                        break;
                    }
                }
            }
        }
        if(fieldExists){
            // Getting field type
            Object fieldType = null;
            if (attr != null) {
                fieldType = attr.getType();
                String type = fieldType.toString();
                // Returning value based on type
                if ("String".equals(type)){
                    return "{\"columnType\": \"string\"}";
                } else if ("float".equals(type) || type.contains("Float")){
                    return "{\"columnType\": \"number\"}";
                } else if ("short".equals(type) || type.contains("Short")){
                    return "{\"columnType\": \"number\"}";
                } else if ("long".equals(type) || type.contains("Long")){
                    return "{\"columnType\": \"number\"}";
                } else if ("double".equals(type) || type.contains("Double")){
                    return "{\"columnType\": \"number\"}";
                } else if ("int".equals(type) || type.contains("Integer")){
                    return "{\"columnType\": \"number\"}";
                } else if ("Date".equals(type)){
                    return "{\"columnType\": \"date\"}";
                } else if ("boolean".equals(type) || type.contains("Boolean")){
                    return "{\"columnType\": \"boolean\"}";
                } else {
                    // Returning by default
                    return "{\"columnType\": \"undefined\"}";
                }
            }
        }
        // Returning by default
        return "{\"columnType\": \"undefined\"}";
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "geti18nText")
    public String PlayListController.geti18nText(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_locale_", required = false) String locale) {
        // Getting current locale
        Locale defaultLocale = new Locale(locale);
        // Building JSON response
        StringBuilder json = new StringBuilder();
        json.append("\"all_isnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.isnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"all_notnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.notnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_false\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.false", null, defaultLocale) +
"\"");
        json.append(",");
```

```
        json.append("\"boolean_true\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.true", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_date\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.date", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_day\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.day", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"date_month\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.month", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_year\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.year", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"number_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.number.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_contains\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.contains", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_ends\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.ends", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isempty", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isnotempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isnotempty", null, defaultLocale)
+ "\"");
        json.append(",");
        json.append("\"string_starts\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.starts", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"button_find\": \"" + messageSource_dtt.getMessage("button_find", null,
defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_isnull\": \"" + messageSource_dtt.getMessage("help.all.isnull",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_notnull\": \"" +
messageSource_dtt.getMessage("help.all.notnull", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_false\": \"" +
messageSource_dtt.getMessage("help.boolean.false", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_true\": \"" +
messageSource_dtt.getMessage("help.boolean.true", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_between\": \"" +
messageSource_dtt.getMessage("help.date.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_date\": \"" + messageSource_dtt.getMessage("help.date.date",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_day\": \"" + messageSource_dtt.getMessage("help.date.day",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_month\": \"" + messageSource_dtt.getMessage("help.date.month",
null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"help_date_year\": \"" + messageSource_dtt.getMessage("help.date.year",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_between\": \"" +
messageSource_dtt.getMessage("help.number.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_eq\": \"" + messageSource_dtt.getMessage("help.number.eq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_neq\": \"" + messageSource_dtt.getMessage("help.number.neq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_gt\": \"" + messageSource_dtt.getMessage("help.number.gt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_lt\": \"" + messageSource_dtt.getMessage("help.number.lt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_goe\": \"" + messageSource_dtt.getMessage("help.number.goe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_loe\": \"" + messageSource_dtt.getMessage("help.number.loe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_contains\": \"" +
messageSource_dtt.getMessage("help.string.contains", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_ends\": \"" +
messageSource_dtt.getMessage("help.string.ends", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isempty\": \"" +
messageSource_dtt.getMessage("help.string.isempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isnotempty\": \"" +
messageSource_dtt.getMessage("help.string.isnotempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_starts\": \"" +
messageSource_dtt.getMessage("help.string.starts", null, defaultLocale) + "\"");
        json.append("}");
        // return JSON with locale strings
        return json.toString();
    }

    @RequestMapping(produces = "text/html", value = "/list")
    public String PlayListController.listDatatablesDetail(Model uiModel, HttpServletRequest
request, @ModelAttribute PlayList playList) {
        // Do common datatables operations: get entity list filtered by request parameters
        listDatatables(uiModel, request);
        // Show only the list fragment (without footer, header, menu, etc.)
        return "forward:/WEB-INF/views/playlists/list.jspx";
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.POST, params =
"datatablesRedirect")
    public String PlayListController.createDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid PlayList playlist, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common create operations (check errors, populate, persist, ...)
        String view = create(playlist, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common create error view (remain in create
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM,
playlist.getId());
        // If create success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
```

```
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.PUT, params =
"datatablesRedirect")
    public String PlayListController.updateDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid PlayList playlist, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common update operations (check errors, populate, merge, ...)
        String view = update(playlist, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common update error view (remain in update
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM,
playlist.getId());
        // If update success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.DELETE, params =
"datatablesRedirect", value = "/{id}")
    public String PlayListController.deleteDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @PathVariable("id") Long id,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        // Do common delete operations (find, remove, add pagination attributes, ...)
        String view = delete(id, page, size, uiModel);
        // If no redirect, return common list view
        if (redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        // Redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }


    @RequestMapping(headers = "Accept=application/json", params = "checkFilters")
    @ResponseBody
    public ResponseEntity<String> PlayListController.checkFilterExpressions(WebRequest request,
@RequestParam(value = "property", required = false) String property, @RequestParam(value =
"expression", required = false) String expression) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json; charset=utf-8");
        if(beanWrapper == null){
            beanWrapper = new BeanWrapperImpl(PlayList.class);
        }
        Class type = beanWrapper.getPropertyType(property);
        boolean response = DatatablesUtils.checkFilterExpressions(type,expression,
messageSource_dtt);
        return new ResponseEntity<String>(String.format("{ \"response\": %s, \"property\":
\"%s\"}",response, property), headers, org.springframework.http.HttpStatus.OK);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", params =
"ajax_find=ByPlayListName", produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>>
PlayListController.findPlayListsByPlayListName(@DatatablesParams DatatablesCriterias criterias,
@RequestParam("playListName") String playListName) {
        BooleanBuilder baseSearch = new BooleanBuilder();

        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<PlayList> entity = new PathBuilder<PlayList>(PlayList.class, "entity");

        baseSearch.and(entity.getString("playListName").eq(playListName));
```

```
        SearchResults<PlayList> searchResult = DatatablesUtils.findByCriteria(entity,
PlayList.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(value = "/exportcsv", produces = "text/csv")
    public void PlayListController.exportCsv(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute PlayList playList, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, playList, ExportType.CSV, new CsvExport(), request, response);
    }

    @RequestMapping(value = "/exportpdf", produces = "text/pdf")
    public void PlayListController.exportPdf(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute PlayList playList, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, playList, ExportType.PDF, new PdfExport(), request, response);
    }

    @RequestMapping(value = "/exportxls", produces = "text/xls")
    public void PlayListController.exportXls(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute PlayList playList, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, playList, ExportType.XLS, new XlsExport(), request, response);
    }

    @RequestMapping(value = "/exportxlsx", produces = "text/xlsx")
    public void PlayListController.exportXlsx(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute PlayList playList, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, playList, ExportType.XLSX, new XlsxExport(), request, response);
    }

    @RequestMapping(value = "/exportxml", produces = "text/xml")
    public void PlayListController.exportXml(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute PlayList playList, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, playList, ExportType.XML, new XmlExport(), request, response);
    }

    private void PlayListController.export(DatatablesCriterias criterias, PlayList playList,
ExportType exportType, DatatablesExport datatablesExport, HttpServletRequest request,
HttpServletResponse response) throws ExportException {
        // Does the export process as is explained in
http://dandelion.github.io/datatables/tutorials/export/controller-based-exports.html
        // 1. Retrieve the data
        List<Map<String, String>> data = retrieveData(criterias, playList, request);
        // 2. Build an instance of "ExportConf"
        ExportConf exportConf = new
ExportConf.Builder(exportType).header(true).exportClass(datatablesExport).autoSize(true).fileNam
e(playList.getClass().getSimpleName()).build();
        // 3. Build an instance of "HtmlTable"
        HtmlTable table = DatatablesUtils.makeHtmlTable(data, criterias, exportConf, request);
        // 4. Render the generated export file
        ExportUtils.renderExport(table, exportConf, response);
    }

    private List<Map<String, String>> PlayListController.retrieveData(DatatablesCriterias
criterias, PlayList playList, HttpServletRequest request) {
        // Cloned criteria in order to not paginate the results
        DatatablesCriterias noPaginationCriteria = new
DatatablesCriterias(criterias.getSearch(), 0, null, criterias.getColumnDefs(),
criterias.getSortingColumnDefs(), criterias.getInternalCounter());
        // Do the search to obtain the data
```

```
        Map<String, Object> baseSearchValuesMap = getPropertyMap(playList, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        org.gvnix.web.datatables.query.SearchResults<com.mediaserver.domain.PlayList>
searchResult = DatatablesUtils.findByCriteria(PlayList.class, PlayList.entityManager(),
noPaginationCriteria, baseSearchValuesMap);
        // Use ConversionService with the obtained data
        return DatatablesUtils.populateDataSet(searchResult.getResults(), "id",
searchResult.getTotalCount(), searchResult.getResultsCount(), criterias.getColumnDefs(), null,
conversionService_dtt).getRows();
    }

    @RequestMapping(value = "/datatables/createform", produces = "application/json", headers =
"Accept=application/json")
    @ResponseBody
    public List<Map<String, String>> PlayListController.createJsonForm(HttpServletRequest
request, HttpServletResponse response, Model uiModel) throws ServletException, IOException {

        // Prepare result
        List<Map<String, String>> result = new ArrayList<Map<String, String>>();
        String controllerPath = "playlists";
        String pageToUse = "create";
        String renderUrl = String.format("/WEB-INF/views/%s/%s.jspx", controllerPath,
pageToUse);

        Map<String, String> item = new HashMap<String, String>();
        final StringWriter buffer = new StringWriter();

        // Call JSP to render update form
        RequestDispatcher dispatcher = request.getRequestDispatcher(renderUrl);

        // spring from:input tag uses BindingResult to locate property editors
        // for each bean property. So, we add a request attribute (required key
        // id BindingResult.MODEL_KEY_PREFIX + object name) with a correctly
        // initialized bindingResult.
        PlayList playList = new PlayList();
        BeanPropertyBindingResult bindingResult = new BeanPropertyBindingResult(playList,
"playList");
        bindingResult.initConversion(conversionService_dtt);
        request.setAttribute(BindingResult.MODEL_KEY_PREFIX + "playList", bindingResult);

        populateItemForRender(request, playList, true);

        dispatcher.include(request, new HttpServletResponseWrapper(response) {

            private PrintWriter writer = new PrintWriter(buffer);

            @Override
            public PrintWriter getWriter() throws IOException {
                return writer;
            }
        });
        String render = buffer.toString();

        // Put rendered content into first column
        item.put("form", render);
        result.add(item);

        return result;
    }

    @RequestMapping(value = "/datatables/updateforms", produces = "application/json", headers =
"Accept=application/json")
    @ResponseBody
    public List<Map<String, String>> PlayListController.updateJsonForms(@RequestParam("id")
Long[] ids, HttpServletRequest request, HttpServletResponse response, Model uiModel) throws
ServletException, IOException {
        if (ArrayUtils.isEmpty(ids)) {
            return new ArrayList<Map<String, String>>();
        }

        // Using PathBuilder, a cascading builder for Predicate expressions
        PathBuilder entity = new PathBuilder(PlayList.class, "entity");
        // URL parameters are used as base search filters
        Set set = new HashSet();
        set.addAll(Arrays.asList(ids));
```

```
        BooleanBuilder filterBy = QuerydslUtils.createPredicateByIn(entity, "id", set);
        // Create a query with filter
        JPAQuery query = new JPAQuery(PlayList.entityManager());
        query = query.from(entity);
        // execute query
        List<PlayList> playLists = query.where(filterBy).list(entity);
        List<Map<String, String>> udpateForms = renderUpdateForm(playLists, request, response);
        return udpateForms;
    }

    public List<Map<String, String>> PlayListController.renderUpdateForm(List<PlayList>
playLists, HttpServletRequest request, HttpServletResponse response) throws ServletException,
IOException {
        // Prepare result
        List<Map<String, String>> result = new ArrayList<Map<String, String>>(playLists.size());
        String controllerPath = "playlists";
        String pageToUse = "update";
        String renderUrl = String.format("/WEB-INF/views/%s/%s.jspx", controllerPath,
pageToUse);
        // For every element
        for (PlayList playList : playLists) {
            Map<String, String> item = new HashMap<String, String>();
            final StringWriter buffer = new StringWriter();
            // Call JSP to render update form
            RequestDispatcher dispatcher = request.getRequestDispatcher(renderUrl);
            populateItemForRender(request, playList, true);
            dispatcher.include(request, new HttpServletResponseWrapper(response) {

                private PrintWriter writer = new PrintWriter(buffer);

                @Override
                public PrintWriter getWriter() throws IOException {
                    return writer;
                }
            });
            String render = buffer.toString();
            // Load item id
            item.put("DT_RowId", conversionService_dtt.convert(playList.getId(), String.class));
            // Put rendered content into first column
            item.put("form", render);
            result.add(item);
        }
        return result;
    }

    public void PlayListController.populateItemForRender(HttpServletRequest request, PlayList
playList, boolean editing) {
        org.springframework.ui.Model uiModel = new org.springframework.ui.ExtendedModelMap();

        request.setAttribute("playList", playList);
        request.setAttribute("itemId",
conversionService_dtt.convert(playList.getId(),String.class));

        if (editing) {
            // spring from:input tag uses BindingResult to locate property editors for each bean
            // property. So, we add a request attribute (required key id
BindingResult.MODEL_KEY_PREFIX + object name)
            // with a correctly initialized bindingResult.
            BeanPropertyBindingResult bindindResult = new BeanPropertyBindingResult(playList,
"playList");
            bindindResult.initConversion(conversionService_dtt);
            request.setAttribute(BindingResult.MODEL_KEY_PREFIX + "playList",bindindResult);
            // Add date time patterns and enums to populate inputs
            populateEditForm(uiModel, playList);
        }

        // Load uiModel attributes into request
        Map<String, Object> modelMap = uiModel.asMap();
        for (String key : modelMap.keySet()){
            request.setAttribute(key, modelMap.get(key));
        }
    }

}
```

13.4.6.12 Fichero "PlayListController_Roo_GvNIXWebJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListBatchService;
import com.mediaserver.web.PlayListController;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.gvnix.web.json.JsonResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

privileged aspect PlayListController_Roo_GvNIXWebJpaBatch {

    @Autowired
    public ConversionService PlayListController.conversionService_batch;

    public static Logger PlayListController.LOGGER_BATCH =
LoggerFactory.getLogger(PlayListController.class);;

    @Autowired
    public PlayListBatchService PlayListController.batchService;



    public List<String> PlayListController.getOIDList(List<PlayList> playLists) {
        List<String> result = new ArrayList<String>(playLists.size());
        for (PlayList playList : playLists) {
            result.add(conversionService_batch.convert(playList.getId(), String.class));
        }
        return result;
    }

    public Map<String, Object> PlayListController.getRequestPropertyValues(PlayList playList,
Iterator<String> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(playList == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
```

```
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(playList);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }

}
```

13.4.6.13 Fichero "PlayListMediaController.java":

```
package com.mediaserver.web;
import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;

import org.gvnix.addon.datatables.GvNIXDatatables;
import org.gvnix.addon.web.mvc.batch.GvNIXWebJpaBatch;
import org.gvnix.addon.web.mvc.jquery.GvNIXWebJQuery;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.json.JsonResponse;
import org.springframework.roo.addon.web.mvc.controller.finder.RooWebFinder;
import org.springframework.roo.addon.web.mvc.controller.scaffold.RooWebScaffold;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.ResponseBody;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.PlayListMediaBatchService;
import com.mediaserver.domain.security.User;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.types.path.PathBuilder;

/**
 * Controlador de la clase PlayListMedia
 * @author UO180258
 *
 */
@RequestMapping("/playlistmedias")
@Controller
@RooWebScaffold(path = "playlistmedias", formBackingObject = PlayListMedia.class)
@RooWebFinder
@GvNIXWebJQuery
```

```
@GvNIXWebJpaBatch(service = PlayListMediaBatchService.class)
@GvNIXDatatables(ajax = true, inlineEditing = true)
public class PlayListMediaController {
    public Map<String, Object> getPropertyMap(PlayListMedia playListMedia, HttpServletRequest
request) {
//  método generado incorrectamente
        return null;
    }

/**
 * Rellena el formulario de edición de PlayListMedias, con odos los medios y PlayLists para
administraador,
 * y solo los del usuario en otro caso
 * @param uiModel
 * @param playListMedia
 */
    void populateEditForm(Model uiModel, PlayListMedia playListMedia) {
        uiModel.addAttribute("playListMedia", playListMedia);
      //Obtenemos el nombre de su rol
                String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                // si usuario administrador
                if (roleString.equals("ADMINISTRATOR")) {
                uiModel.addAttribute("medias", Media.findAllMedias());
                uiModel.addAttribute("playlists", PlayList.findAllPlayLists());
                }
                else{
                        User u=User.getContextUser();
                         uiModel.addAttribute("medias",
Media.findMediasByOwner(u).getResultList());
                     uiModel.addAttribute("playlists",
PlayList.findPlayListsByOwner(u).getResultList());
                }
    }

    /**
     * Busca todos los PlayListMedia de la aplicación mediante AJAx para poder mostrarlos sin
necesidad de actualizar la página
     * @param criterias
     * @param PlayListMedia
     * @param request
     * @return
     */
    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", produces =
"application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>> findAllPlayListMedias(@DatatablesParams
DatatablesCriterias criterias, @ModelAttribute PlayListMedia PlayListMedia, HttpServletRequest
request) {
        // URL parameters are used as base search filters
        Map<String, Object> baseSearchValuesMap = getPropertyMap(PlayListMedia, request);

        setDatatablesBaseFilter(baseSearchValuesMap);
      //Obtenemos el nombre de su rol
                String
roleString=(SecurityContextHolder.getContext().getAuthentication().getAuthorities().toArray())[0
].toString();
                long totalRecords;
                long recordsFound;
                String pkFieldName = "id";
                DataSet<Map<String, String>> dataSet;

                BooleanBuilder baseSearch = new BooleanBuilder();
        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<PlayListMedia> entity = new PathBuilder<PlayListMedia>(PlayListMedia.class,
"entity");
        String referer = request.getHeader("referer");
        //peticion desde PlayList
        Long idPlayList=comprobarIdPlayList(referer);
        if(idPlayList!=null){
                //filtro por playlist
                baseSearch.and(entity.get("playList").get("id").eq(idPlayList));
        }
```

```
        if (!(roleString.equals("ADMINISTRATOR"))){
        //filtro por usuario
        baseSearch.and(entity.get("playList").get("owner").eq(User.getContextUser()));
        }

        SearchResults<PlayListMedia> searchResult = DatatablesUtils.findByCriteria(entity,
PlayListMedia.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        totalRecords = searchResult.getTotalCount();
        recordsFound = searchResult.getResultsCount();
        dataSet = DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName,
totalRecords, recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);

    }

    /**
     * Crea un playListMedia mediante petición AJAx para poder reflejar el resultado sin
necesidad de recargar la página
     * @param playListMedias
     * @param bindingResult
     * @param request
     * @return
     */
    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.POST, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<PlayListMedia>> createBatch(@RequestBody @Valid List<PlayListMedia>
playListMedias, BindingResult bindingResult, HttpServletRequest request) {
        String referer = request.getHeader("referer");
        //peticion desde PlayList
        Long idPlayList=comprobarIdPlayList(referer);
        if(idPlayList!=null){
                PlayList pl_destino=PlayList.findPlayList(idPlayList);
                for(PlayListMedia plm:playListMedias){
                        plm.setPlayList(pl_destino);
                }
        }
        JsonResponse<List<PlayListMedia>> jsonResponse = new
JsonResponse<List<PlayListMedia>>();
        jsonResponse.setValue(playListMedias);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            batchService.create(playListMedias);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setOid(getOIDList(playListMedias));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    /**
     * Encuenta el indice de la playlist que originó la peticion de inserción con objetivo de
poder crear un PlayListMedia asociado a al lista correcta
     * @param referer
     * @return
     */
        private Long comprobarIdPlayList(String referer) {
                //formato de la cadena:

        //http://156.35.98.43:8080/mediaserver/playlists/2?datatablesRedirect=http%3A%2F%2F156.3
5.98.43%3A8080%2Fmediaserver%2Fplaylists%3Fpage%3D1%26size%3D10
                //http://156.35.98.43:8080/mediaserver/playlists/2
                final String patron= "playlists/";
                int inicio=referer.indexOf(patron);
                if (inicio>-1){
```

```
                        inicio+=patron.length();
                        int fin=referer.indexOf("?");
                        String stid;
                        if(fin>-1){
                                stid=referer.substring(inicio,fin);
                        }else{
                                stid=referer.substring(inicio);
                        }
                        return Long.parseLong(stid);
                }

                return null;
        }

}
```

13.4.6.14 Fichero "PlayListMediaController_Roo_Controller_Finder.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.web.PlayListMediaController;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

privileged aspect PlayListMediaController_Roo_Controller_Finder {

    @RequestMapping(params = { "find=ByMedia", "form" }, method = RequestMethod.GET)
    public String PlayListMediaController.findPlayListMediasByMediaForm(Model uiModel) {
        uiModel.addAttribute("medias", Media.findAllMedias());
        return "playlistmedias/findPlayListMediasByMedia";
    }

    @RequestMapping(params = "find=ByMedia", method = RequestMethod.GET)
    public String PlayListMediaController.findPlayListMediasByMedia(@RequestParam("media") Media
media, @RequestParam(value = "page", required = false) Integer page, @RequestParam(value =
"size", required = false) Integer size, @RequestParam(value = "sortFieldName", required = false)
String sortFieldName, @RequestParam(value = "sortOrder", required = false) String sortOrder,
Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("playlistmedias",
PlayListMedia.findPlayListMediasByMedia(media, sortFieldName,
sortOrder).setFirstResult(firstResult).setMaxResults(sizeNo).getResultList());
            float nrOfPages = (float) PlayListMedia.countFindPlayListMediasByMedia(media) /
sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("playlistmedias",
PlayListMedia.findPlayListMediasByMedia(media, sortFieldName, sortOrder).getResultList());
        }
        return "playlistmedias/list";
    }

    @RequestMapping(params = { "find=ByPlayList", "form" }, method = RequestMethod.GET)
    public String PlayListMediaController.findPlayListMediasByPlayListForm(Model uiModel) {
        uiModel.addAttribute("playlists", PlayList.findAllPlayLists());
        return "playlistmedias/findPlayListMediasByPlayList";
    }

    @RequestMapping(params = "find=ByPlayList", method = RequestMethod.GET)
```

```
    public String PlayListMediaController.findPlayListMediasByPlayList(@RequestParam("playList")
PlayList playList, @RequestParam(value = "page", required = false) Integer page,
@RequestParam(value = "size", required = false) Integer size, @RequestParam(value =
"sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("playlistmedias",
PlayListMedia.findPlayListMediasByPlayList(playList, sortFieldName,
sortOrder).setFirstResult(firstResult).setMaxResults(sizeNo).getResultList());
            float nrOfPages = (float) PlayListMedia.countFindPlayListMediasByPlayList(playList)
/ sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("playlistmedias",
PlayListMedia.findPlayListMediasByPlayList(playList, sortFieldName, sortOrder).getResultList());
        }
        return "playlistmedias/list";
    }

}
```

13.4.6.15 Fichero "PlayListMediaController_Roo_Controller.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.web.PlayListMediaController;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

privileged aspect PlayListMediaController_Roo_Controller {

    @RequestMapping(method = RequestMethod.POST, produces = "text/html")
    public String PlayListMediaController.create(@Valid PlayListMedia playListMedia,
BindingResult bindingResult, Model uiModel, HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, playListMedia);
            return "playlistmedias/create";
        }
        uiModel.asMap().clear();
        playListMedia.persist();
        return "redirect:/playlistmedias/" +
encodeUrlPathSegment(playListMedia.getId().toString(), httpServletRequest);
    }

    @RequestMapping(params = "form", produces = "text/html")
    public String PlayListMediaController.createForm(Model uiModel) {
        populateEditForm(uiModel, new PlayListMedia());
        List<String[]> dependencies = new ArrayList<String[]>();
        if (PlayList.countPlayLists() == 0) {
            dependencies.add(new String[] { "playList", "playlists" });
        }
```

```
        if (Media.countMedias() == 0) {
            dependencies.add(new String[] { "media", "medias" });
        }
        uiModel.addAttribute("dependencies", dependencies);
        return "playlistmedias/create";
    }

    @RequestMapping(value = "/{id}", produces = "text/html")
    public String PlayListMediaController.show(@PathVariable("id") Long id, Model uiModel) {
        uiModel.addAttribute("playlistmedia", PlayListMedia.findPlayListMedia(id));
        uiModel.addAttribute("itemId", id);
        return "playlistmedias/show";
    }

    @RequestMapping(produces = "text/html")
    public String PlayListMediaController.list(@RequestParam(value = "page", required = false)
Integer page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value
= "sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("playlistmedias",
PlayListMedia.findPlayListMediaEntries(firstResult, sizeNo, sortFieldName, sortOrder));
            float nrOfPages = (float) PlayListMedia.countPlayListMedias() / sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("playlistmedias",
PlayListMedia.findAllPlayListMedias(sortFieldName, sortOrder));
        }
        return "playlistmedias/list";
    }

    @RequestMapping(method = RequestMethod.PUT, produces = "text/html")
    public String PlayListMediaController.update(@Valid PlayListMedia playListMedia,
BindingResult bindingResult, Model uiModel, HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, playListMedia);
            return "playlistmedias/update";
        }
        uiModel.asMap().clear();
        playListMedia.merge();
        return "redirect:/playlistmedias/" +
encodeUrlPathSegment(playListMedia.getId().toString(), httpServletRequest);
    }

    @RequestMapping(value = "/{id}", params = "form", produces = "text/html")
    public String PlayListMediaController.updateForm(@PathVariable("id") Long id, Model uiModel)
{
        populateEditForm(uiModel, PlayListMedia.findPlayListMedia(id));
        return "playlistmedias/update";
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE, produces = "text/html")
    public String PlayListMediaController.delete(@PathVariable("id") Long id,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        PlayListMedia playListMedia = PlayListMedia.findPlayListMedia(id);
        playListMedia.remove();
        uiModel.asMap().clear();
        uiModel.addAttribute("page", (page == null) ? "1" : page.toString());
        uiModel.addAttribute("size", (size == null) ? "10" : size.toString());
        return "redirect:/playlistmedias";
    }



    String PlayListMediaController.encodeUrlPathSegment(String pathSegment, HttpServletRequest
httpServletRequest) {
        String enc = httpServletRequest.getCharacterEncoding();
        if (enc == null) {
            enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
        }
        try {
```

```
            pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
        } catch (UnsupportedEncodingException uee) {}
        return pathSegment;
    }

}
```

13.4.6.16 Fichero "PlayListMediaController_Roo_GvNIXDatatables.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.core.exception.ExportException;
import com.github.dandelion.datatables.core.export.CsvExport;
import com.github.dandelion.datatables.core.export.DatatablesExport;
import com.github.dandelion.datatables.core.export.ExportConf;
import com.github.dandelion.datatables.core.export.ExportType;
import com.github.dandelion.datatables.core.export.ExportUtils;
import com.github.dandelion.datatables.core.export.XmlExport;
import com.github.dandelion.datatables.core.html.HtmlTable;
import com.github.dandelion.datatables.extras.export.itext.PdfExport;
import com.github.dandelion.datatables.extras.export.poi.XlsExport;
import com.github.dandelion.datatables.extras.export.poi.XlsxExport;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.web.PlayListMediaController;
import com.mediaserver.web.PlayListMediaController_Roo_Controller;
import com.mediaserver.web.PlayListMediaController_Roo_GvNIXDatatables;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.jpa.impl.JPAQuery;
import com.mysema.query.types.path.PathBuilder;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.StringWriter;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpServletResponseWrapper;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.ArrayUtils;
import org.apache.commons.lang3.StringUtils;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
```

```
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.validation.BeanPropertyBindingResult;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

privileged aspect PlayListMediaController_Roo_GvNIXDatatables {

    declare precedence: PlayListMediaController_Roo_GvNIXDatatables,
PlayListMediaController_Roo_Controller;

    @Autowired
    public ConversionService PlayListMediaController.conversionService_dtt;

    @Autowired
    public MessageSource PlayListMediaController.messageSource_dtt;

    public BeanWrapper PlayListMediaController.beanWrapper;

    @RequestMapping(method = RequestMethod.GET, produces = "text/html")
    public String PlayListMediaController.listDatatables(Model uiModel, HttpServletRequest
request) {
        Map<String, String> params = populateParametersMap(request);
        // Get parentId information for details render
        String parentId = params.remove("_dt_parentId");
        if (StringUtils.isNotBlank(parentId)) {
            uiModel.addAttribute("parentId", parentId);
        }
        String rowOnTopIds = params.remove("dtt_row_on_top_ids");
        if (StringUtils.isNotBlank(rowOnTopIds)) {
            uiModel.addAttribute("dtt_row_on_top_ids", rowOnTopIds);
        }
        String tableHashId = params.remove("dtt_parent_table_id_hash");
        if (StringUtils.isNotBlank(tableHashId)) {
            uiModel.addAttribute("dtt_parent_table_id_hash", tableHashId);
        }
        if (!params.isEmpty()) {
            uiModel.addAttribute("baseFilter", params);
        }
        return "playlistmedias/list";
    }

    @ModelAttribute
    public void PlayListMediaController.populateDatatablesConfig(Model uiModel) {
        uiModel.addAttribute("datatablesHasBatchSupport", true);
        uiModel.addAttribute("datatablesUseAjax",true);
        uiModel.addAttribute("datatablesInlineEditing",true);
        uiModel.addAttribute("datatablesInlineCreating",true);
        uiModel.addAttribute("datatablesSecurityApplied",false);
        uiModel.addAttribute("datatablesStandardMode",true);
        uiModel.addAttribute("finderNameParam","ajax_find");
    }

    @RequestMapping(produces = "text/html")
    public String PlayListMediaController.list(@RequestParam(value = "page", required = false)
Integer page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value
= "sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        // overrides the standard Roo list method and
        // delegates on datatables list method
        return listDatatables(uiModel, null);
    }

    public Map<String, String> PlayListMediaController.populateParametersMap(HttpServletRequest
request) {
        Map<String, Object> params;
        if (request == null) {
            params = Collections.emptyMap();
```

```
        } else {
            params = new HashMap<String, Object>(request.getParameterMap());
        }

        Map<String, String> allParams = new HashMap<String, String>(params.size());

        String value;
        Object objValue;
        for (String key : params.keySet()) {
            objValue = params.get(key);
            if (objValue instanceof String[]) {
                value = ((String[]) objValue)[0];
            } else {
                value = (String) objValue;
            }
            allParams.put(key, value);
        }
        return allParams;
    }

    public Map<String, Object> PlayListMediaController.getPropertyMap(PlayListMedia
playListMedia, Enumeration<Map<String, String>> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(playListMedia == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(playListMedia);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }



    public void PlayListMediaController.setDatatablesBaseFilter(Map<String, Object> propertyMap)
{
        // Add here your baseFilters to propertyMap.
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "getColumnType")
    public String PlayListMediaController.getColumnType(Model uiModel, HttpServletRequest
request, @RequestParam(value = "_columnName_", required = false) String columnName) {
        // Getting all declared fields
        boolean fieldExists = false;
        Field attr = null;
        for(Field field : PlayListMedia.class.getDeclaredFields()){
            if(field.getName().equals(columnName)){
                attr = field;
                fieldExists = true;
                break;
            }
        }
```

```
        // If current field not exists on entity, find on superclass
        if(!fieldExists){
            if(PlayListMedia.class.getSuperclass() != null){
                for(Field field : PlayListMedia.class.getSuperclass().getDeclaredFields()){
                    if(field.getName().equals(columnName)){
                        attr = field;
                        fieldExists = true;
                        break;
                    }
                }
            }
        }
        if(fieldExists){
            // Getting field type
            Object fieldType = null;
            if (attr != null) {
                fieldType = attr.getType();
                String type = fieldType.toString();
                // Returning value based on type
                if ("String".equals(type)){
                    return "{\"columnType\": \"string\"}";
                } else if ("float".equals(type) || type.contains("Float")){
                    return "{\"columnType\": \"number\"}";
                } else if ("short".equals(type) || type.contains("Short")){
                    return "{\"columnType\": \"number\"}";
                } else if ("long".equals(type) || type.contains("Long")){
                    return "{\"columnType\": \"number\"}";
                } else if ("double".equals(type) || type.contains("Double")){
                    return "{\"columnType\": \"number\"}";
                } else if ("int".equals(type) || type.contains("Integer")){
                    return "{\"columnType\": \"number\"}";
                } else if ("Date".equals(type)){
                    return "{\"columnType\": \"date\"}";
                } else if ("boolean".equals(type) || type.contains("Boolean")){
                    return "{\"columnType\": \"boolean\"}";
                } else {
                    // Returning by default
                    return "{\"columnType\": \"undefined\"}";
                }
            }
        }
        // Returning by default
        return "{\"columnType\": \"undefined\"}";
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "geti18nText")
    public String PlayListMediaController.geti18nText(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_locale_", required = false) String locale) {
        // Getting current locale
        Locale defaultLocale = new Locale(locale);
        // Building JSON response
        StringBuilder json = new StringBuilder();
        json.append("\"all_isnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.isnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"all_notnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.notnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_false\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.false", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_true\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.true", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.between", null, defaultLocale) +
"\"");
        json.append(",");
```

```
        json.append("\"date_date\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.date", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_day\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.day", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"date_month\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.month", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_year\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.year", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"number_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.number.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_contains\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.contains", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_ends\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.ends", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isempty", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isnotempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isnotempty", null, defaultLocale)
+ "\"");
        json.append(",");
        json.append("\"string_starts\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.starts", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"button_find\": \"" + messageSource_dtt.getMessage("button_find", null,
defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_isnull\": \"" + messageSource_dtt.getMessage("help.all.isnull",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_notnull\": \"" +
messageSource_dtt.getMessage("help.all.notnull", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_false\": \"" +
messageSource_dtt.getMessage("help.boolean.false", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_true\": \"" +
messageSource_dtt.getMessage("help.boolean.true", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_between\": \"" +
messageSource_dtt.getMessage("help.date.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_date\": \"" + messageSource_dtt.getMessage("help.date.date",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_day\": \"" + messageSource_dtt.getMessage("help.date.day",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_month\": \"" + messageSource_dtt.getMessage("help.date.month",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_year\": \"" + messageSource_dtt.getMessage("help.date.year",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_between\": \"" +
messageSource_dtt.getMessage("help.number.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_eq\": \"" + messageSource_dtt.getMessage("help.number.eq",
null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"help_number_neq\": \"" + messageSource_dtt.getMessage("help.number.neq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_gt\": \"" + messageSource_dtt.getMessage("help.number.gt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_lt\": \"" + messageSource_dtt.getMessage("help.number.lt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_goe\": \"" + messageSource_dtt.getMessage("help.number.goe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_loe\": \"" + messageSource_dtt.getMessage("help.number.loe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_contains\": \"" +
messageSource_dtt.getMessage("help.string.contains", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_ends\": \"" +
messageSource_dtt.getMessage("help.string.ends", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isempty\": \"" +
messageSource_dtt.getMessage("help.string.isempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isnotempty\": \"" +
messageSource_dtt.getMessage("help.string.isnotempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_starts\": \"" +
messageSource_dtt.getMessage("help.string.starts", null, defaultLocale) + "\"");
        json.append("}");
        // return JSON with locale strings
        return json.toString();
    }

    @RequestMapping(produces = "text/html", value = "/list")
    public String PlayListMediaController.listDatatablesDetail(Model uiModel, HttpServletRequest
request, @ModelAttribute PlayListMedia playListMedia) {
        // Do common datatables operations: get entity list filtered by request parameters
        listDatatables(uiModel, request);
        // Show only the list fragment (without footer, header, menu, etc.)
        return "forward:/WEB-INF/views/playlistmedias/list.jspx";
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.POST, params =
"datatablesRedirect")
    public String PlayListMediaController.createDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid PlayListMedia playlistmedia,
BindingResult bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common create operations (check errors, populate, persist, ...)
        String view = create(playlistmedia, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common create error view (remain in create
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM,
playlistmedia.getId());
        // If create success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.PUT, params =
"datatablesRedirect")
    public String PlayListMediaController.updateDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid PlayListMedia playlistmedia,
BindingResult bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common update operations (check errors, populate, merge, ...)
```

```
        String view = update(playlistmedia, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common update error view (remain in update
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM,
playlistmedia.getId());
        // If update success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.DELETE, params =
"datatablesRedirect", value = "/{id}")
    public String PlayListMediaController.deleteDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @PathVariable("id") Long id,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        // Do common delete operations (find, remove, add pagination attributes, ...)
        String view = delete(id, page, size, uiModel);
        // If no redirect, return common list view
        if (redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        // Redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }



    @RequestMapping(headers = "Accept=application/json", params = "checkFilters")
    @ResponseBody
    public ResponseEntity<String> PlayListMediaController.checkFilterExpressions(WebRequest
request, @RequestParam(value = "property", required = false) String property,
@RequestParam(value = "expression", required = false) String expression) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json; charset=utf-8");
        if(beanWrapper == null){
            beanWrapper = new BeanWrapperImpl(PlayListMedia.class);
        }
        Class type = beanWrapper.getPropertyType(property);
        boolean response = DatatablesUtils.checkFilterExpressions(type,expression,
messageSource_dtt);
        return new ResponseEntity<String>(String.format("{ \"response\": %s, \"property\":
\"%s\"}",response, property), headers, org.springframework.http.HttpStatus.OK);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", params =
"ajax_find=ByPlayList", produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>>
PlayListMediaController.findPlayListMediasByPlayList(@DatatablesParams DatatablesCriterias
criterias, @RequestParam("playList") PlayList playList) {
        BooleanBuilder baseSearch = new BooleanBuilder();

        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<PlayListMedia> entity = new PathBuilder<PlayListMedia>(PlayListMedia.class,
"entity");

        baseSearch.and(entity.get("playList").eq(playList));

        SearchResults<PlayListMedia> searchResult = DatatablesUtils.findByCriteria(entity,
PlayListMedia.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();
```

```
        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", params =
"ajax_find=ByMedia", produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>>
PlayListMediaController.findPlayListMediasByMedia(@DatatablesParams DatatablesCriterias
criterias, @RequestParam("media") Media media) {
        BooleanBuilder baseSearch = new BooleanBuilder();

        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<PlayListMedia> entity = new PathBuilder<PlayListMedia>(PlayListMedia.class,
"entity");

        baseSearch.and(entity.get("media").eq(media));

        SearchResults<PlayListMedia> searchResult = DatatablesUtils.findByCriteria(entity,
PlayListMedia.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(value = "/exportcsv", produces = "text/csv")
    public void PlayListMediaController.exportCsv(@DatatablesParams DatatablesCriterias
criterias, @ModelAttribute PlayListMedia playListMedia, HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException, ExportException {
        export(criterias, playListMedia, ExportType.CSV, new CsvExport(), request, response);
    }

    @RequestMapping(value = "/exportpdf", produces = "text/pdf")
    public void PlayListMediaController.exportPdf(@DatatablesParams DatatablesCriterias
criterias, @ModelAttribute PlayListMedia playListMedia, HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException, ExportException {
        export(criterias, playListMedia, ExportType.PDF, new PdfExport(), request, response);
    }

    @RequestMapping(value = "/exportxls", produces = "text/xls")
    public void PlayListMediaController.exportXls(@DatatablesParams DatatablesCriterias
criterias, @ModelAttribute PlayListMedia playListMedia, HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException, ExportException {
        export(criterias, playListMedia, ExportType.XLS, new XlsExport(), request, response);
    }

    @RequestMapping(value = "/exportxlsx", produces = "text/xlsx")
    public void PlayListMediaController.exportXlsx(@DatatablesParams DatatablesCriterias
criterias, @ModelAttribute PlayListMedia playListMedia, HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException, ExportException {
        export(criterias, playListMedia, ExportType.XLSX, new XlsxExport(), request, response);
    }

    @RequestMapping(value = "/exportxml", produces = "text/xml")
    public void PlayListMediaController.exportXml(@DatatablesParams DatatablesCriterias
criterias, @ModelAttribute PlayListMedia playListMedia, HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException, ExportException {
        export(criterias, playListMedia, ExportType.XML, new XmlExport(), request, response);
    }
```

```
    private void PlayListMediaController.export(DatatablesCriterias criterias, PlayListMedia
playListMedia, ExportType exportType, DatatablesExport datatablesExport, HttpServletRequest
request, HttpServletResponse response) throws ExportException {
        // Does the export process as is explained in
http://dandelion.github.io/datatables/tutorials/export/controller-based-exports.html
        // 1. Retrieve the data
        List<Map<String, String>> data = retrieveData(criterias, playListMedia, request);
        // 2. Build an instance of "ExportConf"
        ExportConf exportConf = new
ExportConf.Builder(exportType).header(true).exportClass(datatablesExport).autoSize(true).fileNam
e(playListMedia.getClass().getSimpleName()).build();
        // 3. Build an instance of "HtmlTable"
        HtmlTable table = DatatablesUtils.makeHtmlTable(data, criterias, exportConf, request);
        // 4. Render the generated export file
        ExportUtils.renderExport(table, exportConf, response);
    }

    private List<Map<String, String>> PlayListMediaController.retrieveData(DatatablesCriterias
criterias, PlayListMedia playListMedia, HttpServletRequest request) {
        // Cloned criteria in order to not paginate the results
        DatatablesCriterias noPaginationCriteria = new
DatatablesCriterias(criterias.getSearch(), 0, null, criterias.getColumnDefs(),
criterias.getSortingColumnDefs(), criterias.getInternalCounter());
        // Do the search to obtain the data
        Map<String, Object> baseSearchValuesMap = getPropertyMap(playListMedia, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        org.gvnix.web.datatables.query.SearchResults<com.mediaserver.domain.PlayListMedia>
searchResult = DatatablesUtils.findByCriteria(PlayListMedia.class,
PlayListMedia.entityManager(), noPaginationCriteria, baseSearchValuesMap);
        // Use ConversionService with the obtained data
        return DatatablesUtils.populateDataSet(searchResult.getResults(), "id",
searchResult.getTotalCount(), searchResult.getResultsCount(), criterias.getColumnDefs(), null,
conversionService_dtt).getRows();
    }

    @RequestMapping(value = "/datatables/createform", produces = "application/json", headers =
"Accept=application/json")
    @ResponseBody
    public List<Map<String, String>> PlayListMediaController.createJsonForm(HttpServletRequest
request, HttpServletResponse response, Model uiModel) throws ServletException, IOException {

        // Prepare result
        List<Map<String, String>> result = new ArrayList<Map<String, String>>();
        String controllerPath = "playlistmedias";
        String pageToUse = "create";
        String renderUrl = String.format("/WEB-INF/views/%s/%s.jspx", controllerPath,
pageToUse);

        Map<String, String> item = new HashMap<String, String>();
        final StringWriter buffer = new StringWriter();

        // Call JSP to render update form
        RequestDispatcher dispatcher = request.getRequestDispatcher(renderUrl);

        // spring from:input tag uses BindingResult to locate property editors
        // for each bean property. So, we add a request attribute (required key
        // id BindingResult.MODEL_KEY_PREFIX + object name) with a correctly
        // initialized bindingResult.
        PlayListMedia playListMedia = new PlayListMedia();
        BeanPropertyBindingResult bindingResult = new BeanPropertyBindingResult(playListMedia,
"playListMedia");
        bindingResult.initConversion(conversionService_dtt);
        request.setAttribute(BindingResult.MODEL_KEY_PREFIX + "playListMedia", bindingResult);

        populateItemForRender(request, playListMedia, true);

        dispatcher.include(request, new HttpServletResponseWrapper(response) {

            private PrintWriter writer = new PrintWriter(buffer);

            @Override
            public PrintWriter getWriter() throws IOException {
                return writer;
            }
        });
```

```
        String render = buffer.toString();

        // Put rendered content into first column
        item.put("form", render);
        result.add(item);

        return result;
    }

    @RequestMapping(value = "/datatables/updateforms", produces = "application/json", headers =
"Accept=application/json")
    @ResponseBody
    public List<Map<String, String>> PlayListMediaController.updateJsonForms(@RequestParam("id")
Long[] ids, HttpServletRequest request, HttpServletResponse response, Model uiModel) throws
ServletException, IOException {
        if (ArrayUtils.isEmpty(ids)) {
            return new ArrayList<Map<String, String>>();
        }

        // Using PathBuilder, a cascading builder for Predicate expressions
        PathBuilder entity = new PathBuilder(PlayListMedia.class, "entity");
        // URL parameters are used as base search filters
        Set set = new HashSet();
        set.addAll(Arrays.asList(ids));
        BooleanBuilder filterBy = QuerydslUtils.createPredicateByIn(entity, "id", set);
        // Create a query with filter
        JPAQuery query = new JPAQuery(PlayListMedia.entityManager());
        query = query.from(entity);
        // execute query
        List<PlayListMedia> playListMedias = query.where(filterBy).list(entity);
        List<Map<String, String>> udpateForms = renderUpdateForm(playListMedias, request,
response);
        return udpateForms;
    }

    public List<Map<String, String>>
PlayListMediaController.renderUpdateForm(List<PlayListMedia> playListMedias, HttpServletRequest
request, HttpServletResponse response) throws ServletException, IOException {
        // Prepare result
        List<Map<String, String>> result = new ArrayList<Map<String,
String>>(playListMedias.size());
        String controllerPath = "playlistmedias";
        String pageToUse = "update";
        String renderUrl = String.format("/WEB-INF/views/%s/%s.jspx", controllerPath,
pageToUse);
        // For every element
        for (PlayListMedia playListMedia : playListMedias) {
            Map<String, String> item = new HashMap<String, String>();
            final StringWriter buffer = new StringWriter();
            // Call JSP to render update form
            RequestDispatcher dispatcher = request.getRequestDispatcher(renderUrl);
            populateItemForRender(request, playListMedia, true);
            dispatcher.include(request, new HttpServletResponseWrapper(response) {

                private PrintWriter writer = new PrintWriter(buffer);

                @Override
                public PrintWriter getWriter() throws IOException {
                    return writer;
                }
            });
            String render = buffer.toString();
            // Load item id
            item.put("DT_RowId", conversionService_dtt.convert(playListMedia.getId(),
String.class));
            // Put rendered content into first column
            item.put("form", render);
            result.add(item);
        }
        return result;
    }

    public void PlayListMediaController.populateItemForRender(HttpServletRequest request,
PlayListMedia playListMedia, boolean editing) {
        org.springframework.ui.Model uiModel = new org.springframework.ui.ExtendedModelMap();
```

```
        request.setAttribute("playListMedia", playListMedia);
        request.setAttribute("itemId",
conversionService_dtt.convert(playListMedia.getId(),String.class));

        if (editing) {
            // spring from:input tag uses BindingResult to locate property editors for each bean
            // property. So, we add a request attribute (required key id
BindingResult.MODEL_KEY_PREFIX + object name)
            // with a correctly initialized bindingResult.
            BeanPropertyBindingResult bindindResult = new
BeanPropertyBindingResult(playListMedia, "playListMedia");
            bindindResult.initConversion(conversionService_dtt);
            request.setAttribute(BindingResult.MODEL_KEY_PREFIX +
"playListMedia",bindindResult);
            // Add date time patterns and enums to populate inputs
            populateEditForm(uiModel, playListMedia);
        }

        // Load uiModel attributes into request
        Map<String, Object> modelMap = uiModel.asMap();
        for (String key : modelMap.keySet()){
            request.setAttribute(key, modelMap.get(key));
        }
    }

}
```

13.4.6.17 Fichero "PlayListMediaController_Roo_GvNIXWebJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web;

import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.PlayListMediaBatchService;
import com.mediaserver.web.PlayListMediaController;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.gvnix.web.json.JsonResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

privileged aspect PlayListMediaController_Roo_GvNIXWebJpaBatch {

    @Autowired
    public ConversionService PlayListMediaController.conversionService_batch;
```

```
    public static Logger PlayListMediaController.LOGGER_BATCH =
LoggerFactory.getLogger(PlayListMediaController.class);;

    @Autowired
    public PlayListMediaBatchService PlayListMediaController.batchService;

    @RequestMapping(value = "/delete", produces = "application/json", method =
RequestMethod.POST)
    public ResponseEntity<Object> PlayListMediaController.deleteBatch(@RequestParam(value =
"all", required = false) boolean all, @RequestParam(value = "deleteIn", required = false)
Boolean deleteIn, @RequestParam(value = "idList[]", required = false) List<Long> idList,
@ModelAttribute PlayListMedia playListMedia, WebRequest request) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json");
        long count = 0;
        try {
            if (all) {
                Map<String, Object> baseFilter =
getRequestPropertyValues(playListMedia,request.getParameterNames());
                if (baseFilter == null || baseFilter.isEmpty()) {
                    count = batchService.deleteAll();
                } else {
                    count = batchService.deleteByValues(baseFilter);
                }
            } else {
                if (idList == null) {
                    throw new IllegalArgumentException("Missing request parameter 'idList[]'");
                }
                if (!idList.isEmpty()) {
                    if (deleteIn) {
                        count = batchService.deleteIn(idList);
                    } else {
                        count = batchService.deleteNotIn(idList);
                    }
                }
            }
        } catch (RuntimeException e) {
            LOGGER_BATCH.error("error deleting selection", e);
            return new ResponseEntity<Object>(e, headers, HttpStatus.INTERNAL_SERVER_ERROR);
        }
        LOGGER_BATCH.debug("deleted: " + count);
        return new ResponseEntity<Object>(count, headers, HttpStatus.OK);
    }

    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.PUT, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<PlayListMedia>> PlayListMediaController.updateBatch(@RequestBody
@Valid List<PlayListMedia> playListMedias, BindingResult bindingResult, HttpServletRequest
request) {
        JsonResponse<List<PlayListMedia>> jsonResponse = new
JsonResponse<List<PlayListMedia>>();
        jsonResponse.setValue(playListMedias);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            playListMedias = batchService.update(playListMedias);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setValue(playListMedias);
        jsonResponse.setOid(getOIDList(playListMedias));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }



    public List<String> PlayListMediaController.getOIDList(List<PlayListMedia> playListMedias) {
```

```
        List<String> result = new ArrayList<String>(playListMedias.size());
        for (PlayListMedia playListMedia : playListMedias) {
            result.add(conversionService_batch.convert(playListMedia.getId(), String.class));
        }
        return result;
    }

    public Map<String, Object> PlayListMediaController.getRequestPropertyValues(PlayListMedia
playListMedia, Iterator<String> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(playListMedia == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(playListMedia);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }

}
```

13.4.7 Paquete com.mediaserver.web.security

13.4.7.1 Fichero "ChangePasswordController.java":

```
package com.mediaserver.web.security;

import javax.persistence.TypedQuery;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import com.mediaserver.domain.security.User;

/**
 *
 * @author '<a href="mailto:ichsan@gmail.com">Muhammad Ichsan</a>'
 *
 */
@RequestMapping("/passwd")
@Controller
public class ChangePasswordController {

	@Autowired
	private PasswordEncoder passwordEncoder;

	@RequestMapping
	public String createForm(Model uiModel) {
		populateEditForm(uiModel, new ChangePasswordForm());
		return "change_password/form";
	}

	@RequestMapping(method = RequestMethod.POST)
	public String submit(
			@Valid @ModelAttribute("form") ChangePasswordForm form,
			BindingResult bindingResult, Model uiModel) {
		validateMore(form, bindingResult);

		if (!bindingResult.hasErrors()) {
			UserDetails userDetails = (UserDetails) SecurityContextHolder
					.getContext().getAuthentication().getPrincipal();
			String newPassword = form.getNewPassword();
			TypedQuery<User> query = User.findUsersByEmailAddress(userDetails
					.getUsername());

			try {
				User user = query.getSingleResult();

				if (passwordEncoder.matches(form.getOldPassword(),
						user.getPassword())) {
					user.setPassword(passwordEncoder.encode(newPassword));
					user.merge();
					return "redirect:/passwd/thanks";

				} else {
					bindingResult.rejectValue("oldPassword",
							"field_invalid_password");
				}
			} catch (EmptyResultDataAccessException e) {
				bindingResult
						.rejectValue("emailAddress",
"error_no_such_email");
```

```
                        }
                }

                populateEditForm(uiModel, form);
                return "change_password/form";
        }

        @RequestMapping("/thanks")
        public String thanks() {
                return "change_password/thanks";
        }

        private void validateMore(ChangePasswordForm form,
                        BindingResult bindingResult) {

                if (form.getNewPassword() != null
                                && !form.getNewPassword().equals(form.getRetypeNewPassword())) {
                        bindingResult.rejectValue("newPassword", "field_password_mismatch");
                }
        }

        private void populateEditForm(Model uiModel, ChangePasswordForm form) {
                uiModel.addAttribute("form", form);
        }
}
```

13.4.7.2 Fichero "ChangePasswordForm_Roo_JavaBean.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.web.security.ChangePasswordForm;

privileged aspect ChangePasswordForm_Roo_JavaBean {

    public String ChangePasswordForm.getOldPassword() {
        return this.oldPassword;
    }

    public void ChangePasswordForm.setOldPassword(String oldPassword) {
        this.oldPassword = oldPassword;
    }

    public String ChangePasswordForm.getNewPassword() {
        return this.newPassword;
    }

    public void ChangePasswordForm.setNewPassword(String newPassword) {
        this.newPassword = newPassword;
    }

    public String ChangePasswordForm.getRetypeNewPassword() {
        return this.retypeNewPassword;
    }

    public void ChangePasswordForm.setRetypeNewPassword(String retypeNewPassword) {
        this.retypeNewPassword = retypeNewPassword;
    }

}
```

13.4.7.3 Fichero "ChangePasswordForm.java":

```
package com.mediaserver.web.security;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
```

```
import org.springframework.roo.addon.javabean.RooJavaBean;

/**
 *
 * @author '<a href="mailto:ichsan@gmail.com">Muhammad Ichsan</a>'
 *
 */
@RooJavaBean
public class ChangePasswordForm {

        @NotNull
        private String oldPassword;

        @NotNull
        @Size(min = 5, max = 100)
        private String newPassword;
        private String retypeNewPassword;

}
```

13.4.7.4 Fichero "ForgotPasswordController.java":

```
package com.mediaserver.web.security;

import static com.mediaserver.util.EncryptionUtil.decrypt;
import static com.mediaserver.util.EncryptionUtil.encrypt;

import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.net.InetAddress;
import java.net.URLEncoder;
import java.security.InvalidKeyException;
import java.util.Calendar;
import java.util.Properties;

import javax.persistence.TypedQuery;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.mediaserver.domain.security.User;

/**
 *
 * @author '<a href="mailto:ichsan@gmail.com">Muhammad Ichsan</a>'
 *
 */
@RequestMapping("/forgot")
@Controller
public class ForgotPasswordController {

        //Cambiado Mailsender por JavaMailSenderImpl
        @Autowired
        private transient JavaMailSenderImpl mailSender;

        @Autowired
        @Qualifier("accountRecoveryMailTemplate")
        private SimpleMailMessage mailTemplate;

        private String encryptionKey = "miN6CHar4cters!";
```

```
	private int hoursToExpire = 24 * 2;

	@Autowired
	private PasswordEncoder passwordEncoder;

	@RequestMapping
	public String createForm(Model uiModel) {
		populateEditForm(uiModel, new ForgotPasswordForm());
		return "forgot_password/form";
	}

	@RequestMapping(method = RequestMethod.POST)
	public String submit(
			@Valid @ModelAttribute("form") ForgotPasswordForm form,
			BindingResult bindingResult, Model uiModel)
			throws UnsupportedEncodingException, InvalidKeyException {

		if (!bindingResult.hasErrors()) {
			TypedQuery<User> userQuery = User.findUsersByEmailAddress(form
					.getEmailAddress());
			try {
				User user = userQuery.getSingleResult();

				Calendar cal = Calendar.getInstance();
				cal.add(Calendar.HOUR, hoursToExpire);

				String key = encrypt(encryptionKey, form.getEmailAddress()
						+ ":" + cal.getTimeInMillis());

				sendAccountRecoveryRequestMail(user, key);
				return "redirect:/forgot/thanks";

			} catch (EmptyResultDataAccessException e) {
				bindingResult.rejectValue("emailAddress",
						"field_email_unregistered");
			}
		}

		populateEditForm(uiModel, form);
		return "forgot_password/form";
	}
	@RequestMapping(value = "/set", params = "k")
	public String createFormSetNewPassword(
			@RequestParam(value = "k") String key, Model uiModel) {
		populateEditForm(uiModel, new NewPasswordForm());
		return "forgot_password/set_new";
	}

	@RequestMapping(value = "/set", params = "k", method = RequestMethod.POST)
	public String submitSetNewPassword(@RequestParam(value = "k") String key,
			@Valid @ModelAttribute("form") NewPasswordForm form,
			BindingResult bindingResult, Model uiModel) {

		if (!form.getNewPassword().equals(form.getRetypeNewPassword())) {
			bindingResult.rejectValue("newPassword", "field_password_mismatch");
		}

		if (!bindingResult.hasErrors()) {
			String decrypted;
			try {
				decrypted = decrypt(encryptionKey, key);
			} catch (InvalidKeyException e) {
				throw new RuntimeException("Invalid encryption key!", e);
			}

			String[] splits = decrypted.split(":");
			String email = splits[0];
			long expireTime = Long.parseLong(splits[1]);

			TypedQuery<User> userQuery = User.findUsersByEmailAddress(email);

			try {
				User user = userQuery.getSingleResult();
```

```
                                if (System.currentTimeMillis() < expireTime) {
                                        user.setPassword(passwordEncoder.encode(form
                                                        .getNewPassword()));
                                        user.merge();
                                        return "redirect:/login";
                                } else {
                                        bindingResult.reject("error_key_expired");
                                }

                        } catch (EmptyResultDataAccessException e) {
                                bindingResult.reject("field_email_unregistered");
                        }

                }

                // form.setNewPassword("");
                // form.setRetypeNewPassword("");
                populateEditForm(uiModel, form);
                return "forgot_password/set_new";
        }

        @RequestMapping(value = "/thanks")
        public String thanks() {
                return "forgot_password/thanks";
        }

        private void populateEditForm(Model uiModel, ForgotPasswordForm form) {
                uiModel.addAttribute("form", form);
        }

        private void populateEditForm(Model uiModel, NewPasswordForm form) {
                uiModel.addAttribute("form", form);
        }

        private void sendAccountRecoveryRequestMail(User user, String key)
                        throws UnsupportedEncodingException {
                SimpleMailMessage mailMessage = new SimpleMailMessage(mailTemplate);
                mailMessage.setTo(user.getEmailAddress());
                InetAddress IP=null;
                String IPst="mediaserver.com";
                try {//consultamos la ip externa a google para escribir la direccion en el email
de acivación
                        IP=(new java.net.Socket("www.google.com", 80)).getLocalAddress();
                        IPst=IP.getHostAddress();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                mailMessage.setText(mailMessage.getText()
                                .replace("{{FIRST_NAME}}", user.getFirstName())
                                .replace("{{LAST_NAME}}", user.getLastName())
                                .replace("{{KEY}}", URLEncoder.encode(key, "UTF-8"))
                                .replace("{{IP_ADDRESS}}",IPst));

                //Propiedades adicionales
                mailSender.setPort(587);
                Properties prop = mailSender.getJavaMailProperties();
                prop.put("mail.transport.protocol", "smtp");
                prop.put("mail.smtp.starttls.enable","false");
                prop.put("mail.smtp.auth", "true");

                mailSender.send(mailMessage);
        }
        /* Accessors*********************************************************** */

        public void setHoursToExpire(int hoursToExpire) {
                this.hoursToExpire = hoursToExpire;
        }

        public void setEncryptionKey(String encryptionKey) {
                this.encryptionKey = encryptionKey;
        }
}
```

13.4.7.5 Fichero "ForgotPasswordForm_Roo_JavaBean.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.web.security.ForgotPasswordForm;

privileged aspect ForgotPasswordForm_Roo_JavaBean {

    public String ForgotPasswordForm.getEmailAddress() {
        return this.emailAddress;
    }

    public void ForgotPasswordForm.setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

}
```

13.4.7.6 Fichero "ForgotPasswordForm.java":

```
package com.mediaserver.web.security;

import javax.validation.constraints.NotNull;

import org.hibernate.validator.constraints.Email;
import org.springframework.roo.addon.javabean.RooJavaBean;

/**
 *
 * @author '<a href="mailto:ichsan@gmail.com">Muhammad Ichsan</a>'
 *
 */
@RooJavaBean
public class ForgotPasswordForm {

        @NotNull
        @Email
        private String emailAddress;

}
```

13.4.7.7 Fichero" RoleController_Roo_Controller.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.domain.security.Role;
import com.mediaserver.web.security.RoleController;
import java.io.UnsupportedEncodingException;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

privileged aspect RoleController_Roo_Controller {

    @RequestMapping(method = RequestMethod.POST, produces = "text/html")
```

```
    public String RoleController.create(@Valid Role role, BindingResult bindingResult, Model
uiModel, HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, role);
            return "security/roles/create";
        }
        uiModel.asMap().clear();
        role.persist();
        return "redirect:/security/roles/" + encodeUrlPathSegment(role.getId().toString(),
httpServletRequest);
    }

    @RequestMapping(params = "form", produces = "text/html")
    public String RoleController.createForm(Model uiModel) {
        populateEditForm(uiModel, new Role());
        return "security/roles/create";
    }

    @RequestMapping(value = "/{id}", produces = "text/html")
    public String RoleController.show(@PathVariable("id") Long id, Model uiModel) {
        uiModel.addAttribute("role", Role.findRole(id));
        uiModel.addAttribute("itemId", id);
        return "security/roles/show";
    }

    @RequestMapping(produces = "text/html")
    public String RoleController.list(@RequestParam(value = "page", required = false) Integer
page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value =
"sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("roles", Role.findRoleEntries(firstResult, sizeNo,
sortFieldName, sortOrder));
            float nrOfPages = (float) Role.countRoles() / sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("roles", Role.findAllRoles(sortFieldName, sortOrder));
        }
        return "security/roles/list";
    }

    @RequestMapping(method = RequestMethod.PUT, produces = "text/html")
    public String RoleController.update(@Valid Role role, BindingResult bindingResult, Model
uiModel, HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, role);
            return "security/roles/update";
        }
        uiModel.asMap().clear();
        role.merge();
        return "redirect:/security/roles/" + encodeUrlPathSegment(role.getId().toString(),
httpServletRequest);
    }

    @RequestMapping(value = "/{id}", params = "form", produces = "text/html")
    public String RoleController.updateForm(@PathVariable("id") Long id, Model uiModel) {
        populateEditForm(uiModel, Role.findRole(id));
        return "security/roles/update";
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE, produces = "text/html")
    public String RoleController.delete(@PathVariable("id") Long id, @RequestParam(value =
"page", required = false) Integer page, @RequestParam(value = "size", required = false) Integer
size, Model uiModel) {
        Role role = Role.findRole(id);
        role.remove();
        uiModel.asMap().clear();
        uiModel.addAttribute("page", (page == null) ? "1" : page.toString());
        uiModel.addAttribute("size", (size == null) ? "10" : size.toString());
        return "redirect:/security/roles";
    }
```

```
    void RoleController.populateEditForm(Model uiModel, Role role) {
        uiModel.addAttribute("role", role);
    }

    String RoleController.encodeUrlPathSegment(String pathSegment, HttpServletRequest
httpServletRequest) {
        String enc = httpServletRequest.getCharacterEncoding();
        if (enc == null) {
            enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
        }
        try {
            pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
        } catch (UnsupportedEncodingException uee) {}
        return pathSegment;
    }

}
```

13.4.7.8 Fichero "RoleController_Roo_GvNIXDatatables.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.core.exception.ExportException;
import com.github.dandelion.datatables.core.export.CsvExport;
import com.github.dandelion.datatables.core.export.DatatablesExport;
import com.github.dandelion.datatables.core.export.ExportConf;
import com.github.dandelion.datatables.core.export.ExportType;
import com.github.dandelion.datatables.core.export.ExportUtils;
import com.github.dandelion.datatables.core.export.XmlExport;
import com.github.dandelion.datatables.core.html.HtmlTable;
import com.github.dandelion.datatables.extras.export.itext.PdfExport;
import com.github.dandelion.datatables.extras.export.poi.XlsExport;
import com.github.dandelion.datatables.extras.export.poi.XlsxExport;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.types.path.PathBuilder;
import com.mediaserver.domain.security.Role;
import com.mediaserver.web.security.RoleController;
import com.mediaserver.web.security.RoleController_Roo_Controller;
import com.mediaserver.web.security.RoleController_Roo_GvNIXDatatables;
import java.io.IOException;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

privileged aspect RoleController_Roo_GvNIXDatatables {

    declare precedence: RoleController_Roo_GvNIXDatatables, RoleController_Roo_Controller;

    @Autowired
    public ConversionService RoleController.conversionService_dtt;

    @Autowired
    public MessageSource RoleController.messageSource_dtt;

    public BeanWrapper RoleController.beanWrapper;

    @RequestMapping(method = RequestMethod.GET, produces = "text/html")
    public String RoleController.listDatatables(Model uiModel, HttpServletRequest request) {
        Map<String, String> params = populateParametersMap(request);
        // Get parentId information for details render
        String parentId = params.remove("_dt_parentId");
        if (StringUtils.isNotBlank(parentId)) {
            uiModel.addAttribute("parentId", parentId);
        }
        String rowOnTopIds = params.remove("dtt_row_on_top_ids");
        if (StringUtils.isNotBlank(rowOnTopIds)) {
            uiModel.addAttribute("dtt_row_on_top_ids", rowOnTopIds);
        }
        String tableHashId = params.remove("dtt_parent_table_id_hash");
        if (StringUtils.isNotBlank(tableHashId)) {
            uiModel.addAttribute("dtt_parent_table_id_hash", tableHashId);
        }
        if (!params.isEmpty()) {
            uiModel.addAttribute("baseFilter", params);
        }
        return "security/roles/list";
    }

    @ModelAttribute
    public void RoleController.populateDatatablesConfig(Model uiModel) {
        uiModel.addAttribute("datatablesHasBatchSupport", true);
        uiModel.addAttribute("datatablesUseAjax",true);
        uiModel.addAttribute("datatablesInlineEditing",false);
        uiModel.addAttribute("datatablesInlineCreating",false);
        uiModel.addAttribute("datatablesSecurityApplied",false);
        uiModel.addAttribute("datatablesStandardMode",true);
        uiModel.addAttribute("finderNameParam","ajax_find");
    }

    @RequestMapping(produces = "text/html")
    public String RoleController.list(@RequestParam(value = "page", required = false) Integer
page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value =
"sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        // overrides the standard Roo list method and
        // delegates on datatables list method
        return listDatatables(uiModel, null);
    }

    public Map<String, String> RoleController.populateParametersMap(HttpServletRequest request)
{
        Map<String, Object> params;
        if (request == null) {
            params = Collections.emptyMap();
        } else {
            params = new HashMap<String, Object>(request.getParameterMap());
        }
```

```
        Map<String, String> allParams = new HashMap<String, String>(params.size());

        String value;
        Object objValue;
        for (String key : params.keySet()) {
            objValue = params.get(key);
            if (objValue instanceof String[]) {
                value = ((String[]) objValue)[0];
            } else {
                value = (String) objValue;
            }
            allParams.put(key, value);
        }
        return allParams;
    }

    public Map<String, Object> RoleController.getPropertyMap(Role role, Enumeration<Map<String,
String>> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(role == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(role);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }


    public void RoleController.setDatatablesBaseFilter(Map<String, Object> propertyMap) {
        // Add here your baseFilters to propertyMap.
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "getColumnType")
    public String RoleController.getColumnType(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_columnName_", required = false) String columnName) {
        // Getting all declared fields
        boolean fieldExists = false;
        Field attr = null;
        for(Field field : Role.class.getDeclaredFields()){
            if(field.getName().equals(columnName)){
                attr = field;
                fieldExists = true;
                break;
            }
        }
        // If current field not exists on entity, find on superclass
        if(!fieldExists){
            if(Role.class.getSuperclass() != null){
                for(Field field : Role.class.getSuperclass().getDeclaredFields()){
                    if(field.getName().equals(columnName)){
                        attr = field;
```

```
                        fieldExists = true;
                        break;
                    }
                }
            }
        }
        if(fieldExists){
            // Getting field type
            Object fieldType = null;
            if (attr != null) {
                fieldType = attr.getType();
                String type = fieldType.toString();
                // Returning value based on type
                if ("String".equals(type)){
                    return "{\"columnType\": \"string\"}";
                } else if ("float".equals(type) || type.contains("Float")){
                    return "{\"columnType\": \"number\"}";
                } else if ("short".equals(type) || type.contains("Short")){
                    return "{\"columnType\": \"number\"}";
                } else if ("long".equals(type) || type.contains("Long")){
                    return "{\"columnType\": \"number\"}";
                } else if ("double".equals(type) || type.contains("Double")){
                    return "{\"columnType\": \"number\"}";
                } else if ("int".equals(type) || type.contains("Integer")){
                    return "{\"columnType\": \"number\"}";
                } else if ("Date".equals(type)){
                    return "{\"columnType\": \"date\"}";
                } else if ("boolean".equals(type) || type.contains("Boolean")){
                    return "{\"columnType\": \"boolean\"}";
                } else {
                    // Returning by default
                    return "{\"columnType\": \"undefined\"}";
                }
            }
        }
        // Returning by default
        return "{\"columnType\": \"undefined\"}";
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "geti18nText")
    public String RoleController.geti18nText(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_locale_", required = false) String locale) {
        // Getting current locale
        Locale defaultLocale = new Locale(locale);
        // Building JSON response
        StringBuilder json = new StringBuilder();
        json.append("\"all_isnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.isnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"all_notnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.notnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_false\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.false", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_true\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.true", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_date\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.date", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_day\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.day", null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"date_month\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.month", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_year\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.year", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"number_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.number.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_contains\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.contains", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_ends\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.ends", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isempty", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isnotempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isnotempty", null, defaultLocale)
+ "\"");
        json.append(",");
        json.append("\"string_starts\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.starts", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"button_find\": \"" + messageSource_dtt.getMessage("button_find", null,
defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_isnull\": \"" + messageSource_dtt.getMessage("help.all.isnull",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_notnull\": \"" +
messageSource_dtt.getMessage("help.all.notnull", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_false\": \"" +
messageSource_dtt.getMessage("help.boolean.false", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_true\": \"" +
messageSource_dtt.getMessage("help.boolean.true", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_between\": \"" +
messageSource_dtt.getMessage("help.date.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_date\": \"" + messageSource_dtt.getMessage("help.date.date",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_day\": \"" + messageSource_dtt.getMessage("help.date.day",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_month\": \"" + messageSource_dtt.getMessage("help.date.month",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_year\": \"" + messageSource_dtt.getMessage("help.date.year",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_between\": \"" +
messageSource_dtt.getMessage("help.number.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_eq\": \"" + messageSource_dtt.getMessage("help.number.eq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_neq\": \"" + messageSource_dtt.getMessage("help.number.neq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_gt\": \"" + messageSource_dtt.getMessage("help.number.gt",
null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"help_number_lt\": \"" + messageSource_dtt.getMessage("help.number.lt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_goe\": \"" + messageSource_dtt.getMessage("help.number.goe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_loe\": \"" + messageSource_dtt.getMessage("help.number.loe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_contains\": \"" +
messageSource_dtt.getMessage("help.string.contains", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_ends\": \"" +
messageSource_dtt.getMessage("help.string.ends", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isempty\": \"" +
messageSource_dtt.getMessage("help.string.isempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isnotempty\": \"" +
messageSource_dtt.getMessage("help.string.isnotempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_starts\": \"" +
messageSource_dtt.getMessage("help.string.starts", null, defaultLocale) + "\"");
        json.append("}");
        // return JSON with locale strings
        return json.toString();
    }

    @RequestMapping(produces = "text/html", value = "/list")
    public String RoleController.listDatatablesDetail(Model uiModel, HttpServletRequest request,
@ModelAttribute Role role) {
        // Do common datatables operations: get entity list filtered by request parameters
        listDatatables(uiModel, request);
        // Show only the list fragment (without footer, header, menu, etc.)
        return "forward:/WEB-INF/views/security/roles/list.jspx";
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.POST, params =
"datatablesRedirect")
    public String RoleController.createDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid Role role, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common create operations (check errors, populate, persist, ...)
        String view = create(role, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common create error view (remain in create
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM, role.getId());
        // If create success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.PUT, params =
"datatablesRedirect")
    public String RoleController.updateDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid Role role, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common update operations (check errors, populate, merge, ...)
        String view = update(role, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common update error view (remain in update
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
```

```
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM, role.getId());
        // If update success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.DELETE, params =
"datatablesRedirect", value = "/{id}")
    public String RoleController.deleteDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @PathVariable("id") Long id,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        // Do common delete operations (find, remove, add pagination attributes, ...)
        String view = delete(id, page, size, uiModel);
        // If no redirect, return common list view
        if (redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        // Redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", produces =
"application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>> RoleController.findAllRoles(@DatatablesParams
DatatablesCriterias criterias, @ModelAttribute Role role, HttpServletRequest request) {
        // URL parameters are used as base search filters
        Map<String, Object> baseSearchValuesMap = getPropertyMap(role, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        SearchResults<Role> searchResult = DatatablesUtils.findByCriteria(Role.class,
Role.entityManager(), criterias, baseSearchValuesMap, conversionService_dtt, messageSource_dtt);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(headers = "Accept=application/json", params = "checkFilters")
    @ResponseBody
    public ResponseEntity<String> RoleController.checkFilterExpressions(WebRequest request,
@RequestParam(value = "property", required = false) String property, @RequestParam(value =
"expression", required = false) String expression) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json; charset=utf-8");
        if(beanWrapper == null){
            beanWrapper = new BeanWrapperImpl(Role.class);
        }
        Class type = beanWrapper.getPropertyType(property);
        boolean response = DatatablesUtils.checkFilterExpressions(type,expression,
messageSource_dtt);
        return new ResponseEntity<String>(String.format("{ \"response\": %s, \"property\":
\"%s\"}",response, property), headers, org.springframework.http.HttpStatus.OK);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", params =
"ajax_find=ByNameEquals", produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>>
RoleController.findRolesByNameEquals(@DatatablesParams DatatablesCriterias criterias,
@RequestParam("name") String name) {
        BooleanBuilder baseSearch = new BooleanBuilder();
```

```
        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<Role> entity = new PathBuilder<Role>(Role.class, "entity");

        baseSearch.and(entity.getString("name").eq(name));

        SearchResults<Role> searchResult = DatatablesUtils.findByCriteria(entity,
Role.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(value = "/exportcsv", produces = "text/csv")
    public void RoleController.exportCsv(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Role role, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, role, ExportType.CSV, new CsvExport(), request, response);
    }

    @RequestMapping(value = "/exportpdf", produces = "text/pdf")
    public void RoleController.exportPdf(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Role role, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, role, ExportType.PDF, new PdfExport(), request, response);
    }

    @RequestMapping(value = "/exportxls", produces = "text/xls")
    public void RoleController.exportXls(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Role role, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, role, ExportType.XLS, new XlsExport(), request, response);
    }

    @RequestMapping(value = "/exportxlsx", produces = "text/xlsx")
    public void RoleController.exportXlsx(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Role role, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, role, ExportType.XLSX, new XlsxExport(), request, response);
    }

    @RequestMapping(value = "/exportxml", produces = "text/xml")
    public void RoleController.exportXml(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute Role role, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, role, ExportType.XML, new XmlExport(), request, response);
    }

    private void RoleController.export(DatatablesCriterias criterias, Role role, ExportType
exportType, DatatablesExport datatablesExport, HttpServletRequest request, HttpServletResponse
response) throws ExportException {
        // Does the export process as is explained in
http://dandelion.github.io/datatables/tutorials/export/controller-based-exports.html
        // 1. Retrieve the data
        List<Map<String, String>> data = retrieveData(criterias, role, request);
        // 2. Build an instance of "ExportConf"
        ExportConf exportConf = new
ExportConf.Builder(exportType).header(true).exportClass(datatablesExport).autoSize(true).fileNam
e(role.getClass().getSimpleName()).build();
        // 3. Build an instance of "HtmlTable"
        HtmlTable table = DatatablesUtils.makeHtmlTable(data, criterias, exportConf, request);
        // 4. Render the generated export file
        ExportUtils.renderExport(table, exportConf, response);
    }
```

```
    private List<Map<String, String>> RoleController.retrieveData(DatatablesCriterias criterias,
Role role, HttpServletRequest request) {
        // Cloned criteria in order to not paginate the results
        DatatablesCriterias noPaginationCriteria = new
DatatablesCriterias(criterias.getSearch(), 0, null, criterias.getColumnDefs(),
criterias.getSortingColumnDefs(), criterias.getInternalCounter());
        // Do the search to obtain the data
        Map<String, Object> baseSearchValuesMap = getPropertyMap(role, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        org.gvnix.web.datatables.query.SearchResults<com.mediaserver.domain.security.Role>
searchResult = DatatablesUtils.findByCriteria(Role.class, Role.entityManager(),
noPaginationCriteria, baseSearchValuesMap);
        // Use ConversionService with the obtained data
        return DatatablesUtils.populateDataSet(searchResult.getResults(), "id",
searchResult.getTotalCount(), searchResult.getResultsCount(), criterias.getColumnDefs(), null,
conversionService_dtt).getRows();
    }

}
```

13.4.7.9 Fichero "RoleController_Roo_GvNIXWebJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.RoleBatchService;
import com.mediaserver.web.security.RoleController;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.gvnix.web.json.JsonResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

privileged aspect RoleController_Roo_GvNIXWebJpaBatch {

    @Autowired
    public ConversionService RoleController.conversionService_batch;

    public static Logger RoleController.LOGGER_BATCH =
LoggerFactory.getLogger(RoleController.class);;

    @Autowired
    public RoleBatchService RoleController.batchService;

    @RequestMapping(value = "/delete", produces = "application/json", method =
RequestMethod.POST)
```

```
    public ResponseEntity<Object> RoleController.deleteBatch(@RequestParam(value = "all",
required = false) boolean all, @RequestParam(value = "deleteIn", required = false) Boolean
deleteIn, @RequestParam(value = "idList[]", required = false) List<Long> idList, @ModelAttribute
Role role, WebRequest request) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json");
        long count = 0;
        try {
            if (all) {
                Map<String, Object> baseFilter =
getRequestPropertyValues(role,request.getParameterNames());
                if (baseFilter == null || baseFilter.isEmpty()) {
                    count = batchService.deleteAll();
                } else {
                    count = batchService.deleteByValues(baseFilter);
                }
            } else {
                if (idList == null) {
                    throw new IllegalArgumentException("Missing request parameter 'idList[]'");
                }
                if (!idList.isEmpty()) {
                    if (deleteIn) {
                        count = batchService.deleteIn(idList);
                    } else {
                        count = batchService.deleteNotIn(idList);
                    }
                }
            }
        } catch (RuntimeException e) {
            LOGGER_BATCH.error("error deleting selection", e);
            return new ResponseEntity<Object>(e, headers, HttpStatus.INTERNAL_SERVER_ERROR);
        }
        LOGGER_BATCH.debug("deleted: " + count);
        return new ResponseEntity<Object>(count, headers, HttpStatus.OK);
    }

    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.PUT, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<Role>> RoleController.updateBatch(@RequestBody @Valid List<Role>
roles, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<Role>> jsonResponse = new JsonResponse<List<Role>>();
        jsonResponse.setValue(roles);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            roles = batchService.update(roles);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setValue(roles);
        jsonResponse.setOid(getOIDList(roles));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.POST, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<Role>> RoleController.createBatch(@RequestBody @Valid List<Role>
roles, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<Role>> jsonResponse = new JsonResponse<List<Role>>();
        jsonResponse.setValue(roles);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
```

```
            batchService.create(roles);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setOid(getOIDList(roles));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    public List<String> RoleController.getOIDList(List<Role> roles) {
        List<String> result = new ArrayList<String>(roles.size());
        for (Role role : roles) {
            result.add(conversionService_batch.convert(role.getId(), String.class));
        }
        return result;
    }

    public Map<String, Object> RoleController.getRequestPropertyValues(Role role,
Iterator<String> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(role == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(role);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }

}
```

13.4.7.10 Fichero "RoleController.java":

```
package com.mediaserver.web.security;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import javax.servlet.http.HttpServletRequest;

import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.roo.addon.web.mvc.controller.scaffold.RooWebScaffold;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import com.mediaserver.domain.security.Role;

@RequestMapping("/security/roles")
@Controller
@RooWebScaffold(path = "security/roles", formBackingObject = Role.class)
public class RoleController {

    public Map<String, Object> getPropertyMap(Role role, HttpServletRequest request) {
//  método generado incorrectamente
        return null;
    }
}
```

13.4.7.11 Fichero "SignUpController.java":

```
package com.mediaserver.web.security;

import java.io.IOException;
import java.net.InetAddress;
import java.util.Date;
import java.util.Properties;

import javax.persistence.TypedQuery;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.mail.javamail.JavaMailSenderImpl;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;

import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;

/**
 *
 * @author '<a href="mailto:ichsan@gmail.com">Muhammad Ichsan</a>'
 *
 */
@RequestMapping("/signup")
@Controller
public class SignUpController {

        @Autowired
        private PasswordEncoder passwordEncoder;

        @Autowired
        private transient JavaMailSenderImpl mailSender;

        @Autowired
        @Qualifier("accountActivationMailTemplate")
        private SimpleMailMessage mailTemplate;

        private boolean isAccountAutoEnabled=true;

        // Tested
        @RequestMapping(method = RequestMethod.GET)
        public String createSignUpForm(Model uiModel) {
                populateEditForm(uiModel, new SignUpForm());
                return "signup/form";
        }

        // Tested
```

```
	@RequestMapping(method = RequestMethod.POST)
	public String submitSignUpForm(
			@Valid @ModelAttribute("form") SignUpForm form,
			BindingResult bindingResult, Model uiModel) {

		validateMore(form, bindingResult);

		if (!bindingResult.hasErrors()) {
			User user = new User();
			user.setActivationDate(null);
			user.setEmailAddress(form.getEmailAddress());
			user.setFirstName(form.getFirstName());
			user.setLastName(form.getLastName());
			user.setPassword(passwordEncoder.encode(form.getPassword()));
			user.setEnabled(false);
			user.setLocked(false);
			user.persist();
			setUserRole(user);

			String activationKey = createActivationKey(user);

			if (isAccountAutoEnabled) {
				try{
				sendActivationMail(activationKey, user);
				}catch( javax.mail.MessagingException e){
				activateUserNoKey(user);
				}
			}

			return "redirect:/signup/thanks";
		}

		populateEditForm(uiModel, form);
		return "signup/form";
	}

	public static void setUserRole(User user) {
		UserRole userRole=new UserRole();
		userRole.setUser(user);
		Role role=Role.findRolesByNameEquals("USER").getSingleResult();
		if (role==null){
			role=new Role("USER","Regular user");
			role.persist();
		}
		userRole.setRole(role);
		userRole.persist();
	}

	private void validateMore(SignUpForm form, BindingResult bindingResult) {
		//si pass no nula y no coincide con el retype
		if (form.getPassword() != null
				&& !form.getPassword().equals(form.getRetypePassword())) {
			bindingResult.rejectValue("password", "field_password_mismatch");
		}
		if(!form.isValidCaptcha()){
			bindingResult.rejectValue("retypePassword", "recaptcha.mismatch");
		}
		//comprobar usuario no registrado aun
		Long count=User.countFindUsersByEmailAddress(form.getEmailAddress());
		if (count!=0){
			bindingResult.rejectValue("emailAddress", "user_already_registerd");
		}
	}


	@RequestMapping(params = "k", method = RequestMethod.GET)
	public String activateUser(
			@RequestParam(value = "k", required = true) String activationKey,
			@RequestParam(value = "e", required = true) String emailAddress,
			Model uiModel) {
		TypedQuery<User> query = User.findUsersByEmailAddress(emailAddress);

		try {
			User user = query.getSingleResult();
```

```
                        if (createActivationKey(user).equals(activationKey)) {
                                activateUserNoKey(user);
                                return "redirect:/login?activated=1";
                        }

                } catch (EmptyResultDataAccessException e) {
                        // No such mail
                }

                return "signup/error";
        }

        private void activateUserNoKey(User user) {
                user.setActivationDate(new Date());
                user.setEnabled(true);
                user.merge();
        }

        @RequestMapping("/thanks")
        public String thanks(Model uiModel) {
                uiModel.addAttribute("accountAutoEnabled", isAccountAutoEnabled);
                return "signup/thanks";
        }


        private void sendActivationMail(String activationKey, User user) throws
javax.mail.MessagingException {
                SimpleMailMessage mailMessage = new SimpleMailMessage(
                                mailTemplate);
                mailMessage.setTo(user.getEmailAddress());
                InetAddress IP=null;
                String IPst="mediaserver.com";
                try {//consultamos la ip externa a google para escribir la direccion en el email
de acivación
                        IP=(new java.net.Socket("www.google.com", 80)).getLocalAddress();
                        IPst=IP.getHostAddress();
                } catch (IOException e) {
                        e.printStackTrace();
                }
                mailMessage.setText(mailMessage.getText()
                                .replace("{{FIRST_NAME}}", user.getFirstName())
                                .replace("{{LAST_NAME}}", user.getLastName())
                                .replace("{{EMAIL_ADDRESS}}", user.getEmailAddress())
                                .replace("{{ACTIVATION_KEY}}", activationKey)
                                .replace("{{IP_ADDRESS}}",IPst));
                mailSender.setPort(587);

                //Propiedades adicionales
                Properties prop = mailSender.getJavaMailProperties();
                prop.put("mail.transport.protocol", "smtp");
                prop.put("mail.smtp.starttls.enable","false");
                prop.put("mail.smtp.auth", "true");
                //prop.put("mail.smtp.starttls.enable", "true");
                //prop.put("mail.debug", "true");
                mailSender.send(mailMessage);
        }

        private String createActivationKey(User user) {
                return user.getPassword().substring(0, 5);
        }

        private void populateEditForm(Model uiModel, SignUpForm form) {
                uiModel.addAttribute("form", form);
                uiModel.addAttribute("captcha_form", form.getReCaptchaHtml());
        }

        /* Accessors ********************************************************** */

        public void setAccountAutoEnabled(boolean isAccountAutoEnabled) {
                this.isAccountAutoEnabled = isAccountAutoEnabled;
        }
}
```

13.4.7.12 Fichero "SignUpForm_Roo_JavaBean.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.web.security.SignUpForm;
import net.tanesha.recaptcha.ReCaptcha;

privileged aspect SignUpForm_Roo_JavaBean {

    public String SignUpForm.getFirstName() {
        return this.firstName;
    }

    public void SignUpForm.setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String SignUpForm.getLastName() {
        return this.lastName;
    }

    public void SignUpForm.setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String SignUpForm.getEmailAddress() {
        return this.emailAddress;
    }

    public void SignUpForm.setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public String SignUpForm.getPassword() {
        return this.password;
    }

    public void SignUpForm.setPassword(String password) {
        this.password = password;
    }

    public String SignUpForm.getRetypePassword() {
        return this.retypePassword;
    }

    public void SignUpForm.setRetypePassword(String retypePassword) {
        this.retypePassword = retypePassword;
    }

    public String SignUpForm.getRecaptchaChallenge() {
        return this.recaptchaChallenge;
    }

    public void SignUpForm.setRecaptchaChallenge(String recaptchaChallenge) {
        this.recaptchaChallenge = recaptchaChallenge;
    }

    public String SignUpForm.getRecaptchaResponse() {
        return this.recaptchaResponse;
    }

    public void SignUpForm.setRecaptchaResponse(String recaptchaResponse) {
        this.recaptchaResponse = recaptchaResponse;
    }

    public ReCaptcha SignUpForm.getReCatcpha() {
        return this.reCatcpha;
    }

    public void SignUpForm.setReCatcpha(ReCaptcha reCatcpha) {
        this.reCatcpha = reCatcpha;
```

```
    }

    public String SignUpForm.getReCaptchaErrorMessage() {
        return this.reCaptchaErrorMessage;
    }

    public void SignUpForm.setReCaptchaErrorMessage(String reCaptchaErrorMessage) {
        this.reCaptchaErrorMessage = reCaptchaErrorMessage;
    }

}
```

13.4.7.13 Fichero "SignUpForm.java":

```
package com.mediaserver.web.security;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import net.tanesha.recaptcha.ReCaptcha;
import net.tanesha.recaptcha.ReCaptchaFactory;
import net.tanesha.recaptcha.ReCaptchaResponse;

import org.hibernate.validator.constraints.Email;
import org.springframework.roo.addon.javabean.RooJavaBean;

/**
 * @author rohit
 *
 */
@RooJavaBean
public class SignUpForm {

        @NotNull
        @Size(min = 1, max = 100)
        private String firstName;

        @NotNull
        @Size(min = 1, max = 100)
        private String lastName;

        @NotNull
        @Size(min = 1, max = 100)
        @Email
        private String emailAddress;

        @NotNull
        @Size(min = 5, max = 100)
        private String password;

        private String retypePassword;

        private String recaptchaChallenge;
        private String recaptchaResponse;

        private ReCaptcha reCatcpha;
        private String reCaptchaErrorMessage;


        public SignUpForm() {
                reCatcpha = ReCaptchaFactory.newReCaptcha(
                                "6LdfmL8SAAAAAFnT0l3UNPOV8mkpHIown-ysSR1g",
                                "6LdfmL8SAAAAAHKPqUQV5SxrRX9Id6a8cQo-mgpE", false);
        }


        public void setRecaptcha_challenge_field(String recaptchaChallenge) {
                this.recaptchaChallenge = recaptchaChallenge;
        }

        public void setRecaptcha_response_field(String recaptchaResponse) {
                this.recaptchaResponse = recaptchaResponse;
```

```
        }

        public String getReCaptchaHtml() {
                return reCatcpha.createRecaptchaHtml(reCaptchaErrorMessage, null);
        }

        //comentado hasta que funcione correctamente
        //@AssertTrue(message = "Wrong captcha")
        public boolean isValidCaptcha() {
                ReCaptchaResponse reCaptchaResponse = reCatcpha.checkAnswer(
                                "localhost", getRecaptchaChallenge(), getRecaptchaResponse());
                boolean result = reCaptchaResponse.isValid();

                if (!result) {
                        reCaptchaErrorMessage = "Wrong captcha answer";
                }

                return result;
        }

}
```

13.4.7.14 Fichero "UserController_Roo_Controller.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.domain.security.User;
import com.mediaserver.web.security.UserController;
import java.io.UnsupportedEncodingException;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.joda.time.format.DateTimeFormat;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

privileged aspect UserController_Roo_Controller {



    @RequestMapping(params = "form", produces = "text/html")
    public String UserController.createForm(Model uiModel) {
        populateEditForm(uiModel, new User());
        return "security/users/create";
    }

    @RequestMapping(value = "/{id}", produces = "text/html")
    public String UserController.show(@PathVariable("id") Long id, Model uiModel) {
        addDateTimeFormatPatterns(uiModel);
        uiModel.addAttribute("user", User.findUser(id));
        uiModel.addAttribute("itemId", id);
        return "security/users/show";
    }

    @RequestMapping(produces = "text/html")
    public String UserController.list(@RequestParam(value = "page", required = false) Integer
page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value =
"sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
```

```
            uiModel.addAttribute("users", User.findUserEntries(firstResult, sizeNo,
sortFieldName, sortOrder));
            float nrOfPages = (float) User.countUsers() / sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("users", User.findAllUsers(sortFieldName, sortOrder));
        }
        addDateTimeFormatPatterns(uiModel);
        return "security/users/list";
    }



    @RequestMapping(value = "/{id}", params = "form", produces = "text/html")
    public String UserController.updateForm(@PathVariable("id") Long id, Model uiModel) {
        populateEditForm(uiModel, User.findUser(id));
        return "security/users/update";
    }


    void UserController.addDateTimeFormatPatterns(Model uiModel) {
        uiModel.addAttribute("user_activationdate_date_format",
DateTimeFormat.patternForStyle("M-", LocaleContextHolder.getLocale()));
    }

    void UserController.populateEditForm(Model uiModel, User user) {
        uiModel.addAttribute("user", user);
        addDateTimeFormatPatterns(uiModel);
    }

    String UserController.encodeUrlPathSegment(String pathSegment, HttpServletRequest
httpServletRequest) {
        String enc = httpServletRequest.getCharacterEncoding();
        if (enc == null) {
            enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
        }
        try {
            pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
        } catch (UnsupportedEncodingException uee) {}
        return pathSegment;
    }

}
```

13.4.7.15 Fichero "UserController_Roo_GvNIXDatatables.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.core.exception.ExportException;
import com.github.dandelion.datatables.core.export.CsvExport;
import com.github.dandelion.datatables.core.export.DatatablesExport;
import com.github.dandelion.datatables.core.export.ExportConf;
import com.github.dandelion.datatables.core.export.ExportType;
import com.github.dandelion.datatables.core.export.ExportUtils;
import com.github.dandelion.datatables.core.export.XmlExport;
import com.github.dandelion.datatables.core.html.HtmlTable;
import com.github.dandelion.datatables.extras.export.itext.PdfExport;
import com.github.dandelion.datatables.extras.export.poi.XlsExport;
import com.github.dandelion.datatables.extras.export.poi.XlsxExport;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mediaserver.domain.security.User;
import com.mediaserver.web.security.UserController;
import com.mediaserver.web.security.UserController_Roo_Controller;
import com.mediaserver.web.security.UserController_Roo_GvNIXDatatables;
```

```
import java.io.IOException;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

privileged aspect UserController_Roo_GvNIXDatatables {

    declare precedence: UserController_Roo_GvNIXDatatables, UserController_Roo_Controller;

    @Autowired
    public ConversionService UserController.conversionService_dtt;

    @Autowired
    public MessageSource UserController.messageSource_dtt;

    public BeanWrapper UserController.beanWrapper;

    @RequestMapping(method = RequestMethod.GET, produces = "text/html")
    public String UserController.listDatatables(Model uiModel, HttpServletRequest request) {
        Map<String, String> params = populateParametersMap(request);
        // Get parentId information for details render
        String parentId = params.remove("_dt_parentId");
        if (StringUtils.isNotBlank(parentId)) {
            uiModel.addAttribute("parentId", parentId);
        }
        String rowOnTopIds = params.remove("dtt_row_on_top_ids");
        if (StringUtils.isNotBlank(rowOnTopIds)) {
            uiModel.addAttribute("dtt_row_on_top_ids", rowOnTopIds);
        }
        String tableHashId = params.remove("dtt_parent_table_id_hash");
        if (StringUtils.isNotBlank(tableHashId)) {
            uiModel.addAttribute("dtt_parent_table_id_hash", tableHashId);
        }
        if (!params.isEmpty()) {
            uiModel.addAttribute("baseFilter", params);
        }
        return "security/users/list";
    }

    @ModelAttribute
    public void UserController.populateDatatablesConfig(Model uiModel) {
        uiModel.addAttribute("datatablesHasBatchSupport", true);
        uiModel.addAttribute("datatablesUseAjax",true);
```

```
        uiModel.addAttribute("datatablesInlineEditing",false);
        uiModel.addAttribute("datatablesInlineCreating",false);
        uiModel.addAttribute("datatablesSecurityApplied",false);
        uiModel.addAttribute("datatablesStandardMode",true);
        uiModel.addAttribute("finderNameParam","ajax_find");
    }

    @RequestMapping(produces = "text/html")
    public String UserController.list(@RequestParam(value = "page", required = false) Integer
page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value =
"sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        // overrides the standard Roo list method and
        // delegates on datatables list method
        return listDatatables(uiModel, null);
    }

    public Map<String, String> UserController.populateParametersMap(HttpServletRequest request)
{
        Map<String, Object> params;
        if (request == null) {
            params = Collections.emptyMap();
        } else {
            params = new HashMap<String, Object>(request.getParameterMap());
        }

        Map<String, String> allParams = new HashMap<String, String>(params.size());

        String value;
        Object objValue;
        for (String key : params.keySet()) {
            objValue = params.get(key);
            if (objValue instanceof String[]) {
                value = ((String[]) objValue)[0];
            } else {
                value = (String) objValue;
            }
            allParams.put(key, value);
        }
        return allParams;
    }

    public Map<String, Object> UserController.getPropertyMap(User user, Enumeration<Map<String,
String>> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(user == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(user);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }
```

```
    public void UserController.setDatatablesBaseFilter(Map<String, Object> propertyMap) {
        // Add here your baseFilters to propertyMap.
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "getColumnType")
    public String UserController.getColumnType(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_columnName_", required = false) String columnName) {
        // Getting all declared fields
        boolean fieldExists = false;
        Field attr = null;
        for(Field field : User.class.getDeclaredFields()){
            if(field.getName().equals(columnName)){
                attr = field;
                fieldExists = true;
                break;
            }
        }
        // If current field not exists on entity, find on superclass
        if(!fieldExists){
            if(User.class.getSuperclass() != null){
                for(Field field : User.class.getSuperclass().getDeclaredFields()){
                    if(field.getName().equals(columnName)){
                        attr = field;
                        fieldExists = true;
                        break;
                    }
                }
            }
        }
        if(fieldExists){
            // Getting field type
            Object fieldType = null;
            if (attr != null) {
                fieldType = attr.getType();
                String type = fieldType.toString();
                // Returning value based on type
                if ("String".equals(type)){
                    return "{\"columnType\": \"string\"}";
                } else if ("float".equals(type) || type.contains("Float")){
                    return "{\"columnType\": \"number\"}";
                } else if ("short".equals(type) || type.contains("Short")){
                    return "{\"columnType\": \"number\"}";
                } else if ("long".equals(type) || type.contains("Long")){
                    return "{\"columnType\": \"number\"}";
                } else if ("double".equals(type) || type.contains("Double")){
                    return "{\"columnType\": \"number\"}";
                } else if ("int".equals(type) || type.contains("Integer")){
                    return "{\"columnType\": \"number\"}";
                } else if ("Date".equals(type)){
                    return "{\"columnType\": \"date\"}";
                } else if ("boolean".equals(type) || type.contains("Boolean")){
                    return "{\"columnType\": \"boolean\"}";
                } else {
                    // Returning by default
                    return "{\"columnType\": \"undefined\"}";
                }
            }
        }
        // Returning by default
        return "{\"columnType\": \"undefined\"}";
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "geti18nText")
    public String UserController.geti18nText(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_locale_", required = false) String locale) {
        // Getting current locale
        Locale defaultLocale = new Locale(locale);
        // Building JSON response
        StringBuilder json = new StringBuilder();
```

```
        json.append("\"all_isnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.isnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"all_notnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.notnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_false\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.false", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_true\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.true", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_date\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.date", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_day\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.day", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"date_month\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.month", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_year\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.year", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"number_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.number.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_contains\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.contains", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_ends\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.ends", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isempty", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isnotempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isnotempty", null, defaultLocale)
+ "\"");
        json.append(",");
        json.append("\"string_starts\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.starts", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"button_find\": \"" + messageSource_dtt.getMessage("button_find", null,
defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_isnull\": \"" + messageSource_dtt.getMessage("help.all.isnull",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_notnull\": \"" +
messageSource_dtt.getMessage("help.all.notnull", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_false\": \"" +
messageSource_dtt.getMessage("help.boolean.false", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_true\": \"" +
messageSource_dtt.getMessage("help.boolean.true", null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"help_date_between\": \"" +
messageSource_dtt.getMessage("help.date.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_date\": \"" + messageSource_dtt.getMessage("help.date.date",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_day\": \"" + messageSource_dtt.getMessage("help.date.day",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_month\": \"" + messageSource_dtt.getMessage("help.date.month",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_year\": \"" + messageSource_dtt.getMessage("help.date.year",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_between\": \"" +
messageSource_dtt.getMessage("help.number.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_eq\": \"" + messageSource_dtt.getMessage("help.number.eq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_neq\": \"" + messageSource_dtt.getMessage("help.number.neq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_gt\": \"" + messageSource_dtt.getMessage("help.number.gt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_lt\": \"" + messageSource_dtt.getMessage("help.number.lt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_goe\": \"" + messageSource_dtt.getMessage("help.number.goe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_loe\": \"" + messageSource_dtt.getMessage("help.number.loe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_contains\": \"" +
messageSource_dtt.getMessage("help.string.contains", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_ends\": \"" +
messageSource_dtt.getMessage("help.string.ends", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isempty\": \"" +
messageSource_dtt.getMessage("help.string.isempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isnotempty\": \"" +
messageSource_dtt.getMessage("help.string.isnotempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_starts\": \"" +
messageSource_dtt.getMessage("help.string.starts", null, defaultLocale) + "\"");
        json.append("}");
        // return JSON with locale strings
        return json.toString();
    }

    @RequestMapping(produces = "text/html", value = "/list")
    public String UserController.listDatatablesDetail(Model uiModel, HttpServletRequest request,
@ModelAttribute User user) {
        // Do common datatables operations: get entity list filtered by request parameters
        listDatatables(uiModel, request);
        // Show only the list fragment (without footer, header, menu, etc.)
        return "forward:/WEB-INF/views/security/users/list.jspx";
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.POST, params =
"datatablesRedirect")
    public String UserController.createDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid User user, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common create operations (check errors, populate, persist, ...)
        String view = create(user, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common create error view (remain in create
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
```

```
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM, user.getId());
        // If create success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.PUT, params =
"datatablesRedirect")
    public String UserController.updateDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid User user, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common update operations (check errors, populate, merge, ...)
        String view = update(user, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common update error view (remain in update
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM, user.getId());
        // If update success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.DELETE, params =
"datatablesRedirect", value = "/{id}")
    public String UserController.deleteDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @PathVariable("id") Long id,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        // Do common delete operations (find, remove, add pagination attributes, ...)
        String view = delete(id, page, size, uiModel);
        // If no redirect, return common list view
        if (redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        // Redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", produces =
"application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>> UserController.findAllUsers(@DatatablesParams
DatatablesCriterias criterias, @ModelAttribute User user, HttpServletRequest request) {
        // URL parameters are used as base search filters
        Map<String, Object> baseSearchValuesMap = getPropertyMap(user, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        SearchResults<User> searchResult = DatatablesUtils.findByCriteria(User.class,
User.entityManager(), criterias, baseSearchValuesMap, conversionService_dtt, messageSource_dtt);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";
        org.springframework.ui.Model uiModel = new org.springframework.ui.ExtendedModelMap();
        addDateTimeFormatPatterns(uiModel);
        Map<String, Object> datePattern = uiModel.asMap();
```

```
        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), datePattern, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(headers = "Accept=application/json", params = "checkFilters")
    @ResponseBody
    public ResponseEntity<String> UserController.checkFilterExpressions(WebRequest request,
@RequestParam(value = "property", required = false) String property, @RequestParam(value =
"expression", required = false) String expression) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json; charset=utf-8");
        if(beanWrapper == null){
            beanWrapper = new BeanWrapperImpl(User.class);
        }
        Class type = beanWrapper.getPropertyType(property);
        boolean response = DatatablesUtils.checkFilterExpressions(type,expression,
messageSource_dtt);
        return new ResponseEntity<String>(String.format("{ \"response\": %s, \"property\":
\"%s\"}",response, property), headers, org.springframework.http.HttpStatus.OK);
    }

    @RequestMapping(value = "/exportcsv", produces = "text/csv")
    public void UserController.exportCsv(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute User user, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, user, ExportType.CSV, new CsvExport(), request, response);
    }

    @RequestMapping(value = "/exportpdf", produces = "text/pdf")
    public void UserController.exportPdf(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute User user, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, user, ExportType.PDF, new PdfExport(), request, response);
    }

    @RequestMapping(value = "/exportxls", produces = "text/xls")
    public void UserController.exportXls(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute User user, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, user, ExportType.XLS, new XlsExport(), request, response);
    }

    @RequestMapping(value = "/exportxlsx", produces = "text/xlsx")
    public void UserController.exportXlsx(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute User user, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, user, ExportType.XLSX, new XlsxExport(), request, response);
    }

    @RequestMapping(value = "/exportxml", produces = "text/xml")
    public void UserController.exportXml(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute User user, HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException, ExportException {
        export(criterias, user, ExportType.XML, new XmlExport(), request, response);
    }

    private void UserController.export(DatatablesCriterias criterias, User user, ExportType
exportType, DatatablesExport datatablesExport, HttpServletRequest request, HttpServletResponse
response) throws ExportException {
        // Does the export process as is explained in
http://dandelion.github.io/datatables/tutorials/export/controller-based-exports.html
        // 1. Retrieve the data
        List<Map<String, String>> data = retrieveData(criterias, user, request);
        // 2. Build an instance of "ExportConf"
        ExportConf exportConf = new
ExportConf.Builder(exportType).header(true).exportClass(datatablesExport).autoSize(true).fileNam
e(user.getClass().getSimpleName()).build();
        // 3. Build an instance of "HtmlTable"
        HtmlTable table = DatatablesUtils.makeHtmlTable(data, criterias, exportConf, request);
        // 4. Render the generated export file
        ExportUtils.renderExport(table, exportConf, response);
    }
```

```
    private List<Map<String, String>> UserController.retrieveData(DatatablesCriterias criterias,
User user, HttpServletRequest request) {
        // Cloned criteria in order to not paginate the results
        DatatablesCriterias noPaginationCriteria = new
DatatablesCriterias(criterias.getSearch(), 0, null, criterias.getColumnDefs(),
criterias.getSortingColumnDefs(), criterias.getInternalCounter());
        // Do the search to obtain the data
        Map<String, Object> baseSearchValuesMap = getPropertyMap(user, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        org.gvnix.web.datatables.query.SearchResults<com.mediaserver.domain.security.User>
searchResult = DatatablesUtils.findByCriteria(User.class, User.entityManager(),
noPaginationCriteria, baseSearchValuesMap);
        org.springframework.ui.Model uiModel = new org.springframework.ui.ExtendedModelMap();
        addDateTimeFormatPatterns(uiModel);
        Map<String, Object> datePattern = uiModel.asMap();
        // Use ConversionService with the obtained data
        return DatatablesUtils.populateDataSet(searchResult.getResults(), "id",
searchResult.getTotalCount(), searchResult.getResultsCount(), criterias.getColumnDefs(),
datePattern, conversionService_dtt).getRows();
    }

}
```

13.4.7.16 Fichero "UserController_Roo_GvNIXWebJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserBatchService;
import com.mediaserver.web.security.UserController;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.gvnix.web.json.JsonResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

privileged aspect UserController_Roo_GvNIXWebJpaBatch {

    @Autowired
    public ConversionService UserController.conversionService_batch;

    public static Logger UserController.LOGGER_BATCH =
LoggerFactory.getLogger(UserController.class);;

    @Autowired
    public UserBatchService UserController.batchService;
```

```
    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.PUT, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<User>> UserController.updateBatch(@RequestBody @Valid List<User>
users, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<User>> jsonResponse = new JsonResponse<List<User>>();
        jsonResponse.setValue(users);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            users = batchService.update(users);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setValue(users);
        jsonResponse.setOid(getOIDList(users));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.POST, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<User>> UserController.createBatch(@RequestBody @Valid List<User>
users, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<User>> jsonResponse = new JsonResponse<List<User>>();
        jsonResponse.setValue(users);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            batchService.create(users);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setOid(getOIDList(users));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    public List<String> UserController.getOIDList(List<User> users) {
        List<String> result = new ArrayList<String>(users.size());
        for (User user : users) {
            result.add(conversionService_batch.convert(user.getId(), String.class));
        }
        return result;
    }

    public Map<String, Object> UserController.getRequestPropertyValues(User user,
Iterator<String> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(user == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }
```

```
        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(user);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }

}
```

13.4.7.17 Fichero "UserController.java":

```
package com.mediaserver.web.security;

import java.util.Date;
import java.util.List;
import java.util.Map;

import javax.persistence.EntityManager;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import javax.validation.Validator;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.mail.MailSender;
import org.springframework.mail.SimpleMailMessage;
import org.springframework.roo.addon.web.mvc.controller.scaffold.RooWebScaffold;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.context.request.WebRequest;

import com.mediaserver.domain.Media;
import com.mediaserver.domain.PlayList;
import com.mediaserver.domain.PlayListMedia;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;
import com.mediaserver.domain.security.attribute.CreateAttributes;

/**
 *
 * @author '<a href="mailto:ichsan@gmail.com">Muhammad Ichsan</a>'
 *
 */
@RequestMapping("/security/users")
@Controller
@RooWebScaffold(path = "security/users", formBackingObject = User.class)
public class UserController {
        @Autowired
        private PasswordEncoder passwordEncoder;

        @Autowired
```

```
        private transient MailSender mailSender;

        @Autowired
        @Qualifier("accountActivationMailTemplate")
        private SimpleMailMessage mailTemplate;

        @Autowired
        private Validator validator;

        @RequestMapping(method = RequestMethod.POST, produces = "text/html")
        public String create(@Valid User user, BindingResult bindingResult, Model uiModel,
                        HttpServletRequest httpServletRequest) {

                // Validate during creation
                validator.validate(user, CreateAttributes.class);

                if (bindingResult.hasErrors()) {
                        populateEditForm(uiModel, user);
                        return "security/users/create";
                }

                user.setPassword(passwordEncoder.encode(user.getPassword()));
                user.persist();
                SignUpController.setUserRole(user);
                return "redirect:/security/users/"
                                + encodeUrlPathSegment(user.getId().toString(),
httpServletRequest);
        }

    @RequestMapping(method = RequestMethod.PUT, produces = "text/html")
    public String update(@Valid User user, BindingResult bindingResult, Model uiModel,
HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, user);
            return "security/users/update";
        }

        User storedUser = User.findUser(user.getId());
        user.setPassword(storedUser.getPassword()); // Prevent overriding into null

        // If user is get enabled (but previously not), send activation mail
        if (user.getEnabled() && !storedUser.getEnabled()) {
        user.setActivationDate(new Date());
                        //sendActivationMail(user);
        }

        uiModel.asMap().clear();
        user.merge();
        return "redirect:/security/users/" + encodeUrlPathSegment(user.getId().toString(),
httpServletRequest);
    }

        @SuppressWarnings("unused")
        private void sendActivationMail(User user) {
                SimpleMailMessage mailMessage = new SimpleMailMessage(mailTemplate);
                mailMessage.setTo(user.getEmailAddress());

                String activationKey = user.getPassword().substring(0, 5);

                mailMessage.setText(mailMessage.getText()
                        .replace("{{FIRST_NAME}}", user.getFirstName())
                        .replace("{{LAST_NAME}}", user.getLastName())
                        .replace("{{EMAIL_ADDRESS}}", user.getEmailAddress())
                        .replace("{{ACTIVATION_KEY}}", activationKey));

                mailSender.send(mailMessage);
        }
         @RequestMapping(value = "/{id}", method = RequestMethod.DELETE, produces = "text/html")
            public String delete(@PathVariable("id") Long id, @RequestParam(value = "page",
required = false) Integer page, @RequestParam(value = "size", required = false) Integer size,
Model uiModel) {
                User user = User.findUser(id);
                unBind(user);
                user.remove();
                uiModel.asMap().clear();
```

```
                uiModel.addAttribute("page", (page == null) ? "1" : page.toString());
                uiModel.addAttribute("size", (size == null) ? "10" : size.toString());
                return "redirect:/security/users";
            }

            @RequestMapping(value = "/delete", produces = "application/json", method =
RequestMethod.POST)
            public ResponseEntity<Object> deleteBatch(@RequestParam(value = "all", required =
false) boolean all, @RequestParam(value = "deleteIn", required = false) Boolean deleteIn,
@RequestParam(value = "idList[]", required = false) List<Long> idList, @ModelAttribute User
user, WebRequest request) {
                HttpHeaders headers = new HttpHeaders();
                headers.add("Content-Type", "application/json");
                long count = 0;
                try {
                    if (all) {
                        Map<String, Object> baseFilter =
getRequestPropertyValues(user,request.getParameterNames());
                        if (baseFilter == null || baseFilter.isEmpty()) {
                            count = batchService.deleteAll();
                        } else {
                            count = batchService.deleteByValues(baseFilter);
                        }
                    } else {
                        if (idList == null) {
                            throw new IllegalArgumentException("Missing request parameter
'idList[]'");
                        }
                        if (!idList.isEmpty()) {
                            if (deleteIn) {
                                for(Long id:idList){
                                User u=User.findUser(id);
                                unBind(u);
                                }
                                count = batchService.deleteIn(idList);

                            } else {
                                count = batchService.deleteNotIn(idList);
                            }
                        }
                    }
                } catch (RuntimeException e) {
                    LOGGER_BATCH.error("error deleting selection", e);
                    return new ResponseEntity<Object>(e, headers,
HttpStatus.INTERNAL_SERVER_ERROR);
                }
                LOGGER_BATCH.debug("deleted: " + count);
                return new ResponseEntity<Object>(count, headers, HttpStatus.OK);
            }


        private void unBind(User user) {
                List<UserRole> roleList=UserRole.findUserRolesByUser(user).getResultList();
                for (UserRole ur:roleList){
                        ur.remove();
                }
                List<PlayList> plList=PlayList.findPlayListsByOwner(user).getResultList();
                for (PlayList pl:plList){
                        List<PlayListMedia>
plmList=PlayListMedia.findPlayListMediasByPlayList(pl).getResultList();
                        for(PlayListMedia plm:plmList){
                                plm.remove();
                        }
                        pl.remove();
                }
                //medios restante que no esten en ninguna playList
                List<Media> mediaList=Media.findMediasByOwner(user).getResultList();
                for (Media m:mediaList){
                        m.remove();
                }

        }


        public Map<String, Object> getPropertyMap(User user, HttpServletRequest request) {
```

```
//        // URL parameters are used as base search filters
//        @SuppressWarnings("unchecked") Map<String, Object> propertyValuesMap =
getPropertyMap(user, request.getParameterNames());
//        // Add to the property map the parameters used as query operators
//        Map<String, Object> params = new HashMap<String,
Object>(populateParametersMap(request));
//        Set<String> keySet = params.keySet();
//        for (String key : keySet) {
//            if (key.startsWith(QuerydslUtils.OPERATOR_PREFIX)) {
//                propertyValuesMap.put(key, params.get(key));
//            } else if (DatatablesUtils.ROWS_ON_TOP_IDS_PARAM.equals(key)) {
//                propertyValuesMap.put(key, request.getParameterMap().get(key));
//            } else if(DatatablesUtils.BOUNDING_BOX_PARAM.equals(key) ||
DatatablesUtils.BOUNDING_BOX_FIELDS_PARAM.equals(key)){
//                propertyValuesMap.put(key, request.getParameterMap().get(key));
//            }
//        }
//        return propertyValuesMap;
        return null;
    }
}
```

13.4.7.18 Fichero "UserRoleController_Roo_Controller.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.domain.security.Role;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;
import com.mediaserver.web.security.UserRoleController;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.List;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.util.UriUtils;
import org.springframework.web.util.WebUtils;

privileged aspect UserRoleController_Roo_Controller {

    @RequestMapping(method = RequestMethod.POST, produces = "text/html")
    public String UserRoleController.create(@Valid UserRole userRole, BindingResult
bindingResult, Model uiModel, HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, userRole);
            return "security/assignrole/create";
        }
        uiModel.asMap().clear();
        userRole.persist();
        return "redirect:/security/assignrole/" +
encodeUrlPathSegment(userRole.getId().toString(), httpServletRequest);
    }

    @RequestMapping(params = "form", produces = "text/html")
    public String UserRoleController.createForm(Model uiModel) {
        populateEditForm(uiModel, new UserRole());
        List<String[]> dependencies = new ArrayList<String[]>();
        if (User.countUsers() == 0) {
            dependencies.add(new String[] { "user", "security/users" });
        }
        if (Role.countRoles() == 0) {
            dependencies.add(new String[] { "role", "security/roles" });
        }
```

```
        uiModel.addAttribute("dependencies", dependencies);
        return "security/assignrole/create";
    }

    @RequestMapping(value = "/{id}", produces = "text/html")
    public String UserRoleController.show(@PathVariable("id") Long id, Model uiModel) {
        uiModel.addAttribute("userrole", UserRole.findUserRole(id));
        uiModel.addAttribute("itemId", id);
        return "security/assignrole/show";
    }

    @RequestMapping(produces = "text/html")
    public String UserRoleController.list(@RequestParam(value = "page", required = false)
Integer page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value
= "sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        if (page != null || size != null) {
            int sizeNo = size == null ? 10 : size.intValue();
            final int firstResult = page == null ? 0 : (page.intValue() - 1) * sizeNo;
            uiModel.addAttribute("userroles", UserRole.findUserRoleEntries(firstResult, sizeNo,
sortFieldName, sortOrder));
            float nrOfPages = (float) UserRole.countUserRoles() / sizeNo;
            uiModel.addAttribute("maxPages", (int) ((nrOfPages > (int) nrOfPages || nrOfPages ==
0.0) ? nrOfPages + 1 : nrOfPages));
        } else {
            uiModel.addAttribute("userroles", UserRole.findAllUserRoles(sortFieldName,
sortOrder));
        }
        return "security/assignrole/list";
    }

    @RequestMapping(method = RequestMethod.PUT, produces = "text/html")
    public String UserRoleController.update(@Valid UserRole userRole, BindingResult
bindingResult, Model uiModel, HttpServletRequest httpServletRequest) {
        if (bindingResult.hasErrors()) {
            populateEditForm(uiModel, userRole);
            return "security/assignrole/update";
        }
        uiModel.asMap().clear();
        userRole.merge();
        return "redirect:/security/assignrole/" +
encodeUrlPathSegment(userRole.getId().toString(), httpServletRequest);
    }

    @RequestMapping(value = "/{id}", params = "form", produces = "text/html")
    public String UserRoleController.updateForm(@PathVariable("id") Long id, Model uiModel) {
        populateEditForm(uiModel, UserRole.findUserRole(id));
        return "security/assignrole/update";
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE, produces = "text/html")
    public String UserRoleController.delete(@PathVariable("id") Long id, @RequestParam(value =
"page", required = false) Integer page, @RequestParam(value = "size", required = false) Integer
size, Model uiModel) {
        UserRole userRole = UserRole.findUserRole(id);
        userRole.remove();
        uiModel.asMap().clear();
        uiModel.addAttribute("page", (page == null) ? "1" : page.toString());
        uiModel.addAttribute("size", (size == null) ? "10" : size.toString());
        return "redirect:/security/assignrole";
    }

    void UserRoleController.populateEditForm(Model uiModel, UserRole userRole) {
        uiModel.addAttribute("userRole", userRole);
        uiModel.addAttribute("roles", Role.findAllRoles());
        uiModel.addAttribute("users", User.findAllUsers());
    }

    String UserRoleController.encodeUrlPathSegment(String pathSegment, HttpServletRequest
httpServletRequest) {
        String enc = httpServletRequest.getCharacterEncoding();
        if (enc == null) {
            enc = WebUtils.DEFAULT_CHARACTER_ENCODING;
        }
        try {
```

```
            pathSegment = UriUtils.encodePathSegment(pathSegment, enc);
        } catch (UnsupportedEncodingException uee) {}
        return pathSegment;
    }

}
```

13.4.7.19 Fichero "UserRoleController_Roo_GvNIXDatatables.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.github.dandelion.datatables.core.ajax.DataSet;
import com.github.dandelion.datatables.core.ajax.DatatablesCriterias;
import com.github.dandelion.datatables.core.ajax.DatatablesResponse;
import com.github.dandelion.datatables.core.exception.ExportException;
import com.github.dandelion.datatables.core.export.CsvExport;
import com.github.dandelion.datatables.core.export.DatatablesExport;
import com.github.dandelion.datatables.core.export.ExportConf;
import com.github.dandelion.datatables.core.export.ExportType;
import com.github.dandelion.datatables.core.export.ExportUtils;
import com.github.dandelion.datatables.core.export.XmlExport;
import com.github.dandelion.datatables.core.html.HtmlTable;
import com.github.dandelion.datatables.extras.export.itext.PdfExport;
import com.github.dandelion.datatables.extras.export.poi.XlsExport;
import com.github.dandelion.datatables.extras.export.poi.XlsxExport;
import com.github.dandelion.datatables.extras.spring3.ajax.DatatablesParams;
import com.mysema.query.BooleanBuilder;
import com.mysema.query.types.path.PathBuilder;
import com.mediaserver.domain.security.User;
import com.mediaserver.domain.security.UserRole;
import com.mediaserver.web.security.UserRoleController;
import com.mediaserver.web.security.UserRoleController_Roo_Controller;
import com.mediaserver.web.security.UserRoleController_Roo_GvNIXDatatables;
import java.io.IOException;
import java.lang.reflect.Field;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Enumeration;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;
import java.util.Set;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.apache.commons.lang3.StringUtils;
import org.gvnix.web.datatables.query.SearchResults;
import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.MessageSource;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.ResponseEntity;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
```

```
privileged aspect UserRoleController_Roo_GvNIXDatatables {

    declare precedence: UserRoleController_Roo_GvNIXDatatables,
UserRoleController_Roo_Controller;

    @Autowired
    public ConversionService UserRoleController.conversionService_dtt;

    @Autowired
    public MessageSource UserRoleController.messageSource_dtt;

    public BeanWrapper UserRoleController.beanWrapper;

    @RequestMapping(method = RequestMethod.GET, produces = "text/html")
    public String UserRoleController.listDatatables(Model uiModel, HttpServletRequest request) {
        Map<String, String> params = populateParametersMap(request);
        // Get parentId information for details render
        String parentId = params.remove("_dt_parentId");
        if (StringUtils.isNotBlank(parentId)) {
            uiModel.addAttribute("parentId", parentId);
        }
        String rowOnTopIds = params.remove("dtt_row_on_top_ids");
        if (StringUtils.isNotBlank(rowOnTopIds)) {
            uiModel.addAttribute("dtt_row_on_top_ids", rowOnTopIds);
        }
        String tableHashId = params.remove("dtt_parent_table_id_hash");
        if (StringUtils.isNotBlank(tableHashId)) {
            uiModel.addAttribute("dtt_parent_table_id_hash", tableHashId);
        }
        if (!params.isEmpty()) {
            uiModel.addAttribute("baseFilter", params);
        }
        return "security/assignrole/list";
    }

    @ModelAttribute
    public void UserRoleController.populateDatatablesConfig(Model uiModel) {
        uiModel.addAttribute("datatablesHasBatchSupport", true);
        uiModel.addAttribute("datatablesUseAjax",true);
        uiModel.addAttribute("datatablesInlineEditing",false);
        uiModel.addAttribute("datatablesInlineCreating",false);
        uiModel.addAttribute("datatablesSecurityApplied",false);
        uiModel.addAttribute("datatablesStandardMode",true);
        uiModel.addAttribute("finderNameParam","ajax_find");
    }

    @RequestMapping(produces = "text/html")
    public String UserRoleController.list(@RequestParam(value = "page", required = false)
Integer page, @RequestParam(value = "size", required = false) Integer size, @RequestParam(value
= "sortFieldName", required = false) String sortFieldName, @RequestParam(value = "sortOrder",
required = false) String sortOrder, Model uiModel) {
        // overrides the standard Roo list method and
        // delegates on datatables list method
        return listDatatables(uiModel, null);
    }

    public Map<String, String> UserRoleController.populateParametersMap(HttpServletRequest
request) {
        Map<String, Object> params;
        if (request == null) {
            params = Collections.emptyMap();
        } else {
            params = new HashMap<String, Object>(request.getParameterMap());
        }

        Map<String, String> allParams = new HashMap<String, String>(params.size());

        String value;
        Object objValue;
        for (String key : params.keySet()) {
            objValue = params.get(key);
            if (objValue instanceof String[]) {
                value = ((String[]) objValue)[0];
            } else {
```

```
                value = (String) objValue;
            }
            allParams.put(key, value);
        }
        return allParams;
    }

    public Map<String, Object> UserRoleController.getPropertyMap(UserRole userRole,
Enumeration<Map<String, String>> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(userRole == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(userRole);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }



    public void UserRoleController.setDatatablesBaseFilter(Map<String, Object> propertyMap) {
        // Add here your baseFilters to propertyMap.
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "getColumnType")
    public String UserRoleController.getColumnType(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_columnName_", required = false) String columnName) {
        // Getting all declared fields
        boolean fieldExists = false;
        Field attr = null;
        for(Field field : UserRole.class.getDeclaredFields()){
            if(field.getName().equals(columnName)){
                attr = field;
                fieldExists = true;
                break;
            }
        }
        // If current field not exists on entity, find on superclass
        if(!fieldExists){
            if(UserRole.class.getSuperclass() != null){
                for(Field field : UserRole.class.getSuperclass().getDeclaredFields()){
                    if(field.getName().equals(columnName)){
                        attr = field;
                        fieldExists = true;
                        break;
                    }
                }
            }
        }
        if(fieldExists){
            // Getting field type
```

```
            Object fieldType = null;
            if (attr != null) {
                fieldType = attr.getType();
                String type = fieldType.toString();
                // Returning value based on type
                if ("String".equals(type)){
                    return "{\"columnType\": \"string\"}";
                } else if ("float".equals(type) || type.contains("Float")){
                    return "{\"columnType\": \"number\"}";
                } else if ("short".equals(type) || type.contains("Short")){
                    return "{\"columnType\": \"number\"}";
                } else if ("long".equals(type) || type.contains("Long")){
                    return "{\"columnType\": \"number\"}";
                } else if ("double".equals(type) || type.contains("Double")){
                    return "{\"columnType\": \"number\"}";
                } else if ("int".equals(type) || type.contains("Integer")){
                    return "{\"columnType\": \"number\"}";
                } else if ("Date".equals(type)){
                    return "{\"columnType\": \"date\"}";
                } else if ("boolean".equals(type) || type.contains("Boolean")){
                    return "{\"columnType\": \"boolean\"}";
                } else {
                    // Returning by default
                    return "{\"columnType\": \"undefined\"}";
                }
            }
        }
        // Returning by default
        return "{\"columnType\": \"undefined\"}";
    }

    @ResponseBody
    @RequestMapping(headers = "Accept=application/json", params = "geti18nText")
    public String UserRoleController.geti18nText(Model uiModel, HttpServletRequest request,
@RequestParam(value = "_locale_", required = false) String locale) {
        // Getting current locale
        Locale defaultLocale = new Locale(locale);
        // Building JSON response
        StringBuilder json = new StringBuilder();
        json.append("\"all_isnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.isnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"all_notnull\": \"" +
messageSource_dtt.getMessage("global.filters.operations.all.notnull", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_false\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.false", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"boolean_true\": \"" +
messageSource_dtt.getMessage("global.filters.operations.boolean.true", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_date\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.date", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_day\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.day", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"date_month\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.month", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"date_year\": \"" +
messageSource_dtt.getMessage("global.filters.operations.date.year", null, defaultLocale) +
"\"");
        json.append(",");
```

```
        json.append("\"number_between\": \"" +
messageSource_dtt.getMessage("global.filters.operations.number.between", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_contains\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.contains", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_ends\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.ends", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isempty", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"string_isnotempty\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.isnotempty", null, defaultLocale)
+ "\"");
        json.append(",");
        json.append("\"string_starts\": \"" +
messageSource_dtt.getMessage("global.filters.operations.string.starts", null, defaultLocale) +
"\"");
        json.append(",");
        json.append("\"button_find\": \"" + messageSource_dtt.getMessage("button_find", null,
defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_isnull\": \"" + messageSource_dtt.getMessage("help.all.isnull",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_all_notnull\": \"" +
messageSource_dtt.getMessage("help.all.notnull", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_false\": \"" +
messageSource_dtt.getMessage("help.boolean.false", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_boolean_true\": \"" +
messageSource_dtt.getMessage("help.boolean.true", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_between\": \"" +
messageSource_dtt.getMessage("help.date.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_date\": \"" + messageSource_dtt.getMessage("help.date.date",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_day\": \"" + messageSource_dtt.getMessage("help.date.day",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_month\": \"" + messageSource_dtt.getMessage("help.date.month",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_date_year\": \"" + messageSource_dtt.getMessage("help.date.year",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_between\": \"" +
messageSource_dtt.getMessage("help.number.between", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_eq\": \"" + messageSource_dtt.getMessage("help.number.eq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_neq\": \"" + messageSource_dtt.getMessage("help.number.neq",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_gt\": \"" + messageSource_dtt.getMessage("help.number.gt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_lt\": \"" + messageSource_dtt.getMessage("help.number.lt",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_goe\": \"" + messageSource_dtt.getMessage("help.number.goe",
null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_number_loe\": \"" + messageSource_dtt.getMessage("help.number.loe",
null, defaultLocale) + "\"");
        json.append(",");
```

```
        json.append("\"help_string_contains\": \"" +
messageSource_dtt.getMessage("help.string.contains", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_ends\": \"" +
messageSource_dtt.getMessage("help.string.ends", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isempty\": \"" +
messageSource_dtt.getMessage("help.string.isempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_isnotempty\": \"" +
messageSource_dtt.getMessage("help.string.isnotempty", null, defaultLocale) + "\"");
        json.append(",");
        json.append("\"help_string_starts\": \"" +
messageSource_dtt.getMessage("help.string.starts", null, defaultLocale) + "\"");
        json.append("}");
        // return JSON with locale strings
        return json.toString();
    }

    @RequestMapping(produces = "text/html", value = "/list")
    public String UserRoleController.listDatatablesDetail(Model uiModel, HttpServletRequest
request, @ModelAttribute UserRole userRole) {
        // Do common datatables operations: get entity list filtered by request parameters
        listDatatables(uiModel, request);
        // Show only the list fragment (without footer, header, menu, etc.)
        return "forward:/WEB-INF/views/security/assignrole/list.jspx";
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.POST, params =
"datatablesRedirect")
    public String UserRoleController.createDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid UserRole userrole, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common create operations (check errors, populate, persist, ...)
        String view = create(userrole, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common create error view (remain in create
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM,
userrole.getId());
        // If create success, redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.PUT, params =
"datatablesRedirect")
    public String UserRoleController.updateDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @Valid UserRole userrole, BindingResult
bindingResult, Model uiModel, RedirectAttributes redirectModel, HttpServletRequest
httpServletRequest) {
        // Do common update operations (check errors, populate, merge, ...)
        String view = update(userrole, bindingResult, uiModel, httpServletRequest);
        // If binding errors or no redirect, return common update error view (remain in update
form)
        if (bindingResult.hasErrors() || redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        String[] paramValues = httpServletRequest.getParameterValues("dtt_table_id_hash");
        if(paramValues != null && paramValues.length > 0) {
            redirectModel.addFlashAttribute("dtt_table_id_hash", paramValues[0]);
        }else{
            redirectModel.addFlashAttribute("dtt_table_id_hash", "");
        }
        redirectModel.addFlashAttribute(DatatablesUtils.ROWS_ON_TOP_IDS_PARAM,
userrole.getId());
        // If update success, redirect to given URL: master datatables
```

```
        return "redirect:".concat(redirect);
    }

    @RequestMapping(produces = "text/html", method = RequestMethod.DELETE, params =
"datatablesRedirect", value = "/{id}")
    public String UserRoleController.deleteDatatablesDetail(@RequestParam(value =
"datatablesRedirect", required = true) String redirect, @PathVariable("id") Long id,
@RequestParam(value = "page", required = false) Integer page, @RequestParam(value = "size",
required = false) Integer size, Model uiModel) {
        // Do common delete operations (find, remove, add pagination attributes, ...)
        String view = delete(id, page, size, uiModel);
        // If no redirect, return common list view
        if (redirect == null || redirect.trim().isEmpty()) {
            return view;
        }
        // Redirect to given URL: master datatables
        return "redirect:".concat(redirect);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", produces =
"application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>>
UserRoleController.findAllUserRoles(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute UserRole userRole, HttpServletRequest request) {
        // URL parameters are used as base search filters
        Map<String, Object> baseSearchValuesMap = getPropertyMap(userRole, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        SearchResults<UserRole> searchResult = DatatablesUtils.findByCriteria(UserRole.class,
UserRole.entityManager(), criterias, baseSearchValuesMap, conversionService_dtt,
messageSource_dtt);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(headers = "Accept=application/json", params = "checkFilters")
    @ResponseBody
    public ResponseEntity<String> UserRoleController.checkFilterExpressions(WebRequest request,
@RequestParam(value = "property", required = false) String property, @RequestParam(value =
"expression", required = false) String expression) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json; charset=utf-8");
        if(beanWrapper == null){
            beanWrapper = new BeanWrapperImpl(UserRole.class);
        }
        Class type = beanWrapper.getPropertyType(property);
        boolean response = DatatablesUtils.checkFilterExpressions(type,expression,
messageSource_dtt);
        return new ResponseEntity<String>(String.format("{ \"response\": %s, \"property\":
\"%s\"}",response, property), headers, org.springframework.http.HttpStatus.OK);
    }

    @RequestMapping(headers = "Accept=application/json", value = "/datatables/ajax", params =
"ajax_find=ByUser", produces = "application/json")
    @ResponseBody
    public DatatablesResponse<Map<String, String>>
UserRoleController.findUserRolesByUser(@DatatablesParams DatatablesCriterias criterias,
@RequestParam("user") User user) {
        BooleanBuilder baseSearch = new BooleanBuilder();

        // Base Search. Using BooleanBuilder, a cascading builder for
        // Predicate expressions
        PathBuilder<UserRole> entity = new PathBuilder<UserRole>(UserRole.class, "entity");

        baseSearch.and(entity.get("user").eq(user));
```

```
        SearchResults<UserRole> searchResult = DatatablesUtils.findByCriteria(entity,
UserRole.entityManager(), criterias, baseSearch);

        // Get datatables required counts
        long totalRecords = searchResult.getTotalCount();
        long recordsFound = searchResult.getResultsCount();

        // Entity pk field name
        String pkFieldName = "id";

        DataSet<Map<String, String>> dataSet =
DatatablesUtils.populateDataSet(searchResult.getResults(), pkFieldName, totalRecords,
recordsFound, criterias.getColumnDefs(), null, conversionService_dtt);
        return DatatablesResponse.build(dataSet,criterias);
    }

    @RequestMapping(value = "/exportcsv", produces = "text/csv")
    public void UserRoleController.exportCsv(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute UserRole userRole, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, userRole, ExportType.CSV, new CsvExport(), request, response);
    }

    @RequestMapping(value = "/exportpdf", produces = "text/pdf")
    public void UserRoleController.exportPdf(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute UserRole userRole, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, userRole, ExportType.PDF, new PdfExport(), request, response);
    }

    @RequestMapping(value = "/exportxls", produces = "text/xls")
    public void UserRoleController.exportXls(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute UserRole userRole, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, userRole, ExportType.XLS, new XlsExport(), request, response);
    }

    @RequestMapping(value = "/exportxlsx", produces = "text/xlsx")
    public void UserRoleController.exportXlsx(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute UserRole userRole, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, userRole, ExportType.XLSX, new XlsxExport(), request, response);
    }

    @RequestMapping(value = "/exportxml", produces = "text/xml")
    public void UserRoleController.exportXml(@DatatablesParams DatatablesCriterias criterias,
@ModelAttribute UserRole userRole, HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException, ExportException {
        export(criterias, userRole, ExportType.XML, new XmlExport(), request, response);
    }

    private void UserRoleController.export(DatatablesCriterias criterias, UserRole userRole,
ExportType exportType, DatatablesExport datatablesExport, HttpServletRequest request,
HttpServletResponse response) throws ExportException {
        // Does the export process as is explained in
http://dandelion.github.io/datatables/tutorials/export/controller-based-exports.html
        // 1. Retrieve the data
        List<Map<String, String>> data = retrieveData(criterias, userRole, request);
        // 2. Build an instance of "ExportConf"
        ExportConf exportConf = new
ExportConf.Builder(exportType).header(true).exportClass(datatablesExport).autoSize(true).fileNam
e(userRole.getClass().getSimpleName()).build();
        // 3. Build an instance of "HtmlTable"
        HtmlTable table = DatatablesUtils.makeHtmlTable(data, criterias, exportConf, request);
        // 4. Render the generated export file
        ExportUtils.renderExport(table, exportConf, response);
    }

    private List<Map<String, String>> UserRoleController.retrieveData(DatatablesCriterias
criterias, UserRole userRole, HttpServletRequest request) {
        // Cloned criteria in order to not paginate the results
        DatatablesCriterias noPaginationCriteria = new
DatatablesCriterias(criterias.getSearch(), 0, null, criterias.getColumnDefs(),
criterias.getSortingColumnDefs(), criterias.getInternalCounter());
```

```
        // Do the search to obtain the data
        Map<String, Object> baseSearchValuesMap = getPropertyMap(userRole, request);
        setDatatablesBaseFilter(baseSearchValuesMap);
        org.gvnix.web.datatables.query.SearchResults<com.mediaserver.domain.security.UserRole>
searchResult = DatatablesUtils.findByCriteria(UserRole.class, UserRole.entityManager(),
noPaginationCriteria, baseSearchValuesMap);
        // Use ConversionService with the obtained data
        return DatatablesUtils.populateDataSet(searchResult.getResults(), "id",
searchResult.getTotalCount(), searchResult.getResultsCount(), criterias.getColumnDefs(), null,
conversionService_dtt).getRows();
    }

}
```

13.4.7.20 Fichero "UserRoleController_Roo_GvNIXWebJpaBatch.aj":

```
// WARNING: DO NOT EDIT THIS FILE. THIS FILE IS MANAGED BY SPRING ROO.
// You may push code into the target .java compilation unit if you wish to edit any member(s).

package com.mediaserver.web.security;

import com.mediaserver.domain.security.UserRole;
import com.mediaserver.domain.security.UserRoleBatchService;
import com.mediaserver.web.security.UserRoleController;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import javax.servlet.http.HttpServletRequest;
import javax.validation.Valid;
import org.apache.commons.collections.CollectionUtils;
import org.gvnix.web.json.JsonResponse;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.BeanWrapper;
import org.springframework.beans.BeanWrapperImpl;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.core.convert.ConversionService;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.context.request.WebRequest;

privileged aspect UserRoleController_Roo_GvNIXWebJpaBatch {

    @Autowired
    public ConversionService UserRoleController.conversionService_batch;

    public static Logger UserRoleController.LOGGER_BATCH =
LoggerFactory.getLogger(UserRoleController.class);;

    @Autowired
    public UserRoleBatchService UserRoleController.batchService;

    @RequestMapping(value = "/delete", produces = "application/json", method =
RequestMethod.POST)
    public ResponseEntity<Object> UserRoleController.deleteBatch(@RequestParam(value = "all",
required = false) boolean all, @RequestParam(value = "deleteIn", required = false) Boolean
deleteIn, @RequestParam(value = "idList[]", required = false) List<Long> idList, @ModelAttribute
UserRole userRole, WebRequest request) {
        HttpHeaders headers = new HttpHeaders();
        headers.add("Content-Type", "application/json");
```

```
        long count = 0;
        try {
            if (all) {
                Map<String, Object> baseFilter =
getRequestPropertyValues(userRole,request.getParameterNames());
                if (baseFilter == null || baseFilter.isEmpty()) {
                    count = batchService.deleteAll();
                } else {
                    count = batchService.deleteByValues(baseFilter);
                }
            } else {
                if (idList == null) {
                    throw new IllegalArgumentException("Missing request parameter 'idList[]'");
                }
                if (!idList.isEmpty()) {
                    if (deleteIn) {
                        count = batchService.deleteIn(idList);
                    } else {
                        count = batchService.deleteNotIn(idList);
                    }
                }
            }
        } catch (RuntimeException e) {
            LOGGER_BATCH.error("error deleting selection", e);
            return new ResponseEntity<Object>(e, headers, HttpStatus.INTERNAL_SERVER_ERROR);
        }
        LOGGER_BATCH.debug("deleted: " + count);
        return new ResponseEntity<Object>(count, headers, HttpStatus.OK);
    }

    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.PUT, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<UserRole>> UserRoleController.updateBatch(@RequestBody @Valid
List<UserRole> userRoles, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<UserRole>> jsonResponse = new JsonResponse<List<UserRole>>();
        jsonResponse.setValue(userRoles);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            userRoles = batchService.update(userRoles);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
        }
        jsonResponse.setValue(userRoles);
        jsonResponse.setOid(getOIDList(userRoles));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    @RequestMapping(produces = "application/json", consumes = "application/json", method =
RequestMethod.POST, headers = "Accept=application/json")
    @ResponseBody
    public JsonResponse<List<UserRole>> UserRoleController.createBatch(@RequestBody @Valid
List<UserRole> userRoles, BindingResult bindingResult, HttpServletRequest request) {
        JsonResponse<List<UserRole>> jsonResponse = new JsonResponse<List<UserRole>>();
        jsonResponse.setValue(userRoles);
        if (bindingResult.hasErrors()) {
            jsonResponse.setBindingResult(bindingResult);
            jsonResponse.setStatus("ERROR");
            return jsonResponse;
        }
        try {
            batchService.create(userRoles);
        }
        catch(Exception ex) {
            jsonResponse.setStatus("ERROR");
            jsonResponse.setExceptionMessage(ex.getLocalizedMessage());
            return jsonResponse;
```

```
        }
        jsonResponse.setOid(getOIDList(userRoles));
        jsonResponse.setStatus("SUCCESS");
        return jsonResponse;
    }

    public List<String> UserRoleController.getOIDList(List<UserRole> userRoles) {
        List<String> result = new ArrayList<String>(userRoles.size());
        for (UserRole userRole : userRoles) {
            result.add(conversionService_batch.convert(userRole.getId(), String.class));
        }
        return result;
    }

    public Map<String, Object> UserRoleController.getRequestPropertyValues(UserRole userRole,
Iterator<String> propertyNames) {
        Map<String, Object> propertyValuesMap = new HashMap<String, Object>();

        // If no entity or properties given, return empty Map
        if(userRole == null || propertyNames == null) {
            return propertyValuesMap;
        }

        List<String> properties = new ArrayList<String>();
        CollectionUtils.addAll(properties, propertyNames);

        // There must be at least one property name, otherwise return empty Map
        if (properties.isEmpty()) {
            return propertyValuesMap;
        }

        // Iterate over given properties to get each property value
        BeanWrapper entityBean = new BeanWrapperImpl(userRole);
        for (String propertyName : properties) {
            if (entityBean.isReadableProperty(propertyName)) {
                Object propertyValue = null;
                try {
                    propertyValue = entityBean.getPropertyValue(propertyName);
                } catch (Exception e){
                    // TODO log warning
                    continue;
                }
                propertyValuesMap.put(propertyName, propertyValue);
            }
        }
        return propertyValuesMap;
    }

}
```

13.4.7.21 Fichero "UserRoleController.java":

```
package com.mediaserver.web.security;
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

import javax.servlet.http.HttpServletRequest;

import org.gvnix.web.datatables.util.DatatablesUtils;
import org.gvnix.web.datatables.util.QuerydslUtils;
import org.springframework.roo.addon.web.mvc.controller.scaffold.RooWebScaffold;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import com.mediaserver.domain.security.UserRole;

@RequestMapping("/security/assignrole")
@Controller
@RooWebScaffold(path = "security/assignrole", formBackingObject = UserRole.class)
public class UserRoleController {
```

```
        public Map<String, Object> getPropertyMap(UserRole userRole, HttpServletRequest request)
{
//método generado incorrectamente
                return null;
    }
}
```

13.4.8 Paquete com.websocket.config

13.4.8.1 Fichero "CORSFilter.java":

```
package com.websocket.config;


import java.io.IOException;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.web.filter.OncePerRequestFilter;

/**
 * Enabling CORS support  - Access-Control-Allow-Origin
 * @author zeroows@gmail.com
 *
 * <code>
        <!-- Add this to your web.xml to enable "CORS" -->
        <filter>
          <filter-name>cors</filter-name>
          <filter-class>com.elm.mb.rest.filters.CORSFilter</filter-class>
        </filter>

        <filter-mapping>
          <filter-name>cors</filter-name>
          <url-pattern>/*</url-pattern>
        </filter-mapping>
 * </code>
 */
public class CORSFilter extends OncePerRequestFilter {
        private static final Log LOG = LogFactory.getLog(CORSFilter.class);

        @Override
        protected void doFilterInternal(HttpServletRequest request, HttpServletResponse
response, FilterChain filterChain) throws ServletException, IOException {
                response.addHeader("Access-Control-Allow-Origin", "*");

                if (request.getHeader("Access-Control-Request-Method") != null &&
"OPTIONS".equals(request.getMethod())) {
                        LOG.trace("Sending Header....");
                        // CORS "pre-flight" request
                        response.addHeader("Access-Control-Allow-Methods", "GET, POST, PUT,
DELETE");
//                      response.addHeader("Access-Control-Allow-Headers", "Authorization");
            response.addHeader("Access-Control-Allow-Headers", "Content-Type");
                        response.addHeader("Access-Control-Max-Age", "1");
                }

                filterChain.doFilter(request, response);
        }

}
```

13.4.8.2 Fichero "WebSocketConfig.java":

```
package com.websocket.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Profile;
import org.springframework.messaging.simp.config.ChannelRegistration;
import org.springframework.messaging.simp.config.MessageBrokerRegistry;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.stereotype.Controller;
import org.springframework.web.socket.config.annotation.EnableWebSocketMessageBroker;
import org.springframework.web.socket.config.annotation.StompEndpointRegistry;
import org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer;

//@Configuration
@Profile("default")
@EnableWebSocketMessageBroker
@EnableScheduling
@Controller
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
    /**
     * Configure STOMP
     */
    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        System.out.println("registrando endpoint /simplemessages");
        registry.addEndpoint("/simplemessages").withSockJS();

    }

    /**
     * Configure message broker options.
     */
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/topic/", "/queue/");
        config.setApplicationDestinationPrefixes("/app");
        System.out.println("configurando MessageBroker");
    }


    @Override
    public void configureClientInboundChannel(ChannelRegistration registration) {
    }

    @Override
    public void configureClientOutboundChannel(ChannelRegistration registration) {
        registration.taskExecutor().corePoolSize(4).maxPoolSize(10);
        System.out.println("configurando Canal de salida");
    }

}
```

13.4.9 Paquete com.websocket.controllers

13.4.9.1 Fichero "WebSocketBroadcastController.java":

```
package com.websocket.controllers;

import java.security.Principal;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.messaging.handler.annotation.MessageExceptionHandler;
import org.springframework.messaging.handler.annotation.MessageMapping;
import org.springframework.messaging.simp.SimpMessagingTemplate;
import org.springframework.messaging.simp.annotation.SendToUser;
import org.springframework.security.crypto.password.StandardPasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import com.mediaserver.domain.security.User;
import com.websocket.model.MessageBroadcast;
import com.websocket.model.SimpleMessage;
import com.websocket.util.Base64;
import com.websocket.util.Util;

@Controller
public class WebSocketBroadcastController {

         @Autowired
        private SimpMessagingTemplate template;
         @Autowired
         private StandardPasswordEncoder passwordEncoder;

    /*
     * target depending on user name
     */
    @MessageMapping("/simplemessages")
    public void processMessageFromClient(SimpleMessage message,
            Principal principal) throws Exception {

        MessageBroadcast payload= new MessageBroadcast("Server response: Did you send
&lt;b&gt;'" + message.getMessage() + "'&lt;/b&gt;? (Server Response at: " + Util.getSimpleDate()
+ ")");

this.template.convertAndSend("/topic/simplemessagesresponse/"+principal.getName(),payload);
    }


    @MessageExceptionHandler
    @SendToUser("/queue/errors")
    public String handleException(Throwable exception) {
        return exception.getMessage();
    }

    /**
     * Metodo de control remoto, destinado a test
     * Paso de parametros por get
     * @param order
     * @param userName
     * @param userPass
     * @return
     * @throws Exception
     */
    @RequestMapping(value="/playersecure",method=RequestMethod.GET)
    public @ResponseBody String CambiarReproductorSecure (@RequestParam(value="o",required=true)
String order, @RequestParam(value="u",required=true) String
userName,@RequestParam(value="p",required=true) String userPass) throws Exception{
        User u=User.findUsersByEmailAddress(userName).getSingleResult();
        if (u==null){
        return "wrongUsername";}
        if(!passwordEncoder.matches(userPass, u.getPassword())){
```

```
                return "wrongPassword";}
        SimpleMessage message=new SimpleMessage();
        message.setMessage(order);
        if(checkUser(userName)){
                MessageBroadcast payload= new MessageBroadcast( message.getMessage());
                this.template.convertAndSend("/topic/simplemessagesresponse/"+userName,payload);
                return order;
        }
        return "error";
    }

    /**
     * Reciabe ordenes de control remoto por POST
     * @param order
     * @param userName
     * @param userPass
     * @return
     * @throws Exception
     */
    @RequestMapping(value="/playersecure",method=RequestMethod.POST)
    public @ResponseBody String CambiarReproductorSecurePost
(@RequestParam(value="o",required=true) String order,
                @RequestParam(value="u",required=true) String
userName,@RequestParam(value="p",required=true) String userPass) throws Exception{
        User u=User.findUsersByEmailAddress(userName).getSingleResult();
        if (u==null){
        return "wrongUsername";}
        String password=new String( Base64.decode( userPass, Base64.DEFAULT-userName.length())
);
        if(!passwordEncoder.matches(password, u.getPassword())){
                return "wrongPassword";}
        SimpleMessage message=new SimpleMessage();
        message.setMessage(order);
        if(checkUser(userName)){
                MessageBroadcast payload= new MessageBroadcast( message.getMessage());
                this.template.convertAndSend("/topic/simplemessagesresponse/"+userName,payload);
                return order;
        }
        return "error";
    }

        private boolean checkUser(String userName) {
                return true;
        }

}
```

13.4.10 Paquete com.websocket.model

13.4.10.1 Fichero "MessageBroadcast.java":

```
package com.websocket.model;

/**
 * Clase que representa el mensaje básico de multidifusión
 * @author UO180258
 *
 */
public class MessageBroadcast {

    private String messageContent;

    public MessageBroadcast() {
    }

    public MessageBroadcast(String messageContent) {
        this.messageContent = messageContent;
    }
```

```
    public String getMessageContent() {
        return messageContent;
    }

    public void setMessageContent(String messageContent) {
        this.messageContent = messageContent;
    }

}
```

13.4.10.2 Fichero "SimpleMessage.java":

```
package com.websocket.model;

public class SimpleMessage {

        private String message;

        public String getMessage() {
                return this.message;
        }

        public void setMessage(String message) {
                this.message = message;
        }
}
```

13.4.11 Paquete com.websocket.util

13.4.11.1 Fichero "Util.java":

```
/**
 * Copyright &copy; Sunit Katkar (sunitkatkar@gmail.com) http://sunitkatkar.blogspot.com
 */
package com.websocket.util;

import java.text.DateFormat;
import java.util.Date;
import java.util.Locale;
import java.util.Locale.Category;

/**
 * {@link Util} is for some utility methods.
 *
 * @author <a href="mailto:sunitkatkar@gmail.com">Sunit Katkar</a>
 * @since 1.0
 * @version 1.0.0.1
 */
public final class Util {

    /**
     * Utility method to get a simple human readable date and time using the default locale set
for the JVM on the
     * server is used.
     *
     * @return - Simple date and time
     */
    public static String getSimpleDate() {
        return getSimpleDate(null);
    }

    /**
     * Utility method to get a simple human readable date and time
     *
     * @param locale
     *            - If null then the default locale set for the JVM on the server is used.
     * @return - Simple date and time
```

```
     */
    public static String getSimpleDate(Locale locale) {
        Date date = new Date();
        DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.SHORT,
DateFormat.MEDIUM,
                ((locale == null) ? Locale.getDefault(Category.DISPLAY) : locale));
        String formattedDate = dateFormat.format(date);
        return formattedDate;
    }
}
```

13.4.12 Script de comunicación websockets para control de video

13.4.12.1 Fichero "videoSocket.js":

```
var stompClient = null;

//Function to connect the web client to the websocket server
function connect() {
        var socket = new SockJS(socketDest);
        stompClient = Stomp.over(socket);
        stompClient.connect('', '', function(frame) {
                //setConnected(true);
                // In production code remove the next line
                console.log("Connected: " + frame);

                stompClient.subscribe("/topic/simplemessagesresponse/"
                                + username, function(servermessage) {//Callback when server
responds
                        comprobarOrden(JSON.parse(servermessage.body).messageContent);
                        //Server responded so hide the info alert

                });
        });
}

// Function to disconnect the web client to the websocket server
function disconnect() {

        stompClient.disconnect();
        // Set the connect and disconnect button states
        setConnected(false);
        // In production remove the next line
        console.log("Disconnected");
}

function sendMessageToServer(messageForServer) {
        stompClient.send("/app/simplemessages", {}, JSON.stringify({
                'message' : messageForServer
        }));
}

/**
 * Utility function to return the date time in simple format
 * like Tue Jan 07 2014 @ 11:47:24 AM
 */
function getCurrentDateTime() {
        var date = new Date();
        var n = date.toDateString();
        var time = date.toLocaleTimeString();
        return n + " @ " + time;
}

function comprobarOrden(servermessage) {
        var decoded = $("<div/>").html(servermessage).text();
        var estado = $("#textoEstado").html();
```

```
        if (decoded.search("stop") >= 0 && estado != "stop") {
                $("#textoEstado").html("stop");
                $("#medioActual").html("");
                vlc.playlist.stop();
        } else if (decoded.search("play") >= 0 && estado == "stop") {
                $("#textoEstado").html("play");
                vlc.playlist.play();
        }else if (decoded.search("play") >= 0 && estado == "pause") {
                $("#textoEstado").html("play");
                vlc.playlist.togglePause();
        } else if (decoded.search("pause") >= 0 && estado == "play") {
                $("#textoEstado").html("pause");
                vlc.playlist.pause();
        } else if (decoded.search("next") >= 0) {
                vlc.playlist.next();
        } else if (decoded.search("prev") >= 0) {
                vlc.playlist.prev();
                vlc.playlist.prev();
        }else if (decoded.search("vol")>=0){
                var textoVolumen=$("<b/>").html(decoded).text();
                textoVolumen=textoVolumen.replace('vol','');
                vlc.audio.volume=parseInt(textoVolumen);
        }else if (decoded.search("pos")>=0){
                var textoPosicion=$("<b/>").html(decoded).text();
                textoPosicion=textoPosicion.replace('pos','');
                var pos=parseFloat(textoPosicion);
                pos=pos/100;
                vlc.input.position=pos;
        }else if(decoded.search("fullsc")>=0){
                vlc.video.fullscreen=1;
        }else if(decoded.search("minsc")>=0){
                vlc.video.fullscreen=0;
        }
}

function initPlayer() {

        initListeners();
}

function registerVLCEvent(event, handler) {
        //var vlc = getElementById("vlc");
        if (vlc) {
                if (vlc.attachEvent) {
                        // Microsoft
                        vlc.attachEvent(event, handler);
                } else if (vlc.addEventListener) {
                        // Mozilla: DOM level 2
                        vlc.addEventListener(event, handler, false);
                } else {
                        // DOM level 0
                        vlc["on" + event] = handler;
                }
        }
}

// stop listening to event
function unregisterVLCEvent(event, handler) {
        //var vlc = getElementById("vlc");
        if (vlc) {
                if (vlc.detachEvent) {
                        // Microsoft
                        vlc.detachEvent(event, handler);
                } else if (vlc.removeEventListener) {
                        // Mozilla: DOM level 2
                        vlc.removeEventListener(event, handler, false);
                } else {
                        // DOM level 0
                        vlc["on" + event] = null;
                }
        }
}
```

```
function handle_MediaPlayerPlaying() {
        //alert("Playing");
        if($("#textoEstado").html()!="play"){
                $("#textoEstado").html("play");
                sendMessageToServer("play");
        }
        var current=vlc.playlist.currentItem;
        if (current!=-1){
                $("#medioActual").html("["+current+":"+((vlc.playlist.items.count)-1)+"]");
        }
}

function handle_MediaPlayerPaused() {
        if($("#textoEstado").html()!="pause"){
                sendMessageToServer("pause");
        }
}

function handle_MediaPlayerEndReached(){
        if($("#textoEstado").html()!="ended"){
                $("#textoEstado").html("ended");
        }
}

function handle_MediaPlayerEncounteredError(){
        if($("#textoEstado").html()!="error"){
                $("#textoEstado").html("error");
        }
}


function initListeners() {
        registerVLCEvent("MediaPlayerPlaying", handle_MediaPlayerPlaying);
        registerVLCEvent("MediaPlayerPaused", handle_MediaPlayerPaused);
        registerVLCEvent("MediaPlayerEndReached", handle_MediaPlayerEndReached);
        registerVLCEvent("MediaPlayerEncounteredError", handle_MediaPlayerEncounteredError);

}
```

13.4.13 Archivos jspx

13.4.13.1 Directorio WEB-INF

13.4.13.1.1 Fichero “accessDeniedException.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags"
xmlns:util="urn:jsptagdir:/WEB-INF/tags/jquery/util" version="2.0">

    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
        <jsp:output omit-xml-declaration="yes" />
        <spring:message var="title" code="error_accessdenied_title"
                htmlEscape="false" />
        <util:panel id="title" title="${title}">
                <h2>${fn:escapeXml(title)}</h2>
                <p>
                        <spring:message code="error_accessdenied_problemdescription" />
                </p>
                <c:if test="${not empty exception}">
                        <p>
                        <h4>
                                <spring:message code="exception_details" />
                        </h4>
                        <spring:message var="message" code="exception_message"
                                htmlEscape="false" />
```

```
                        <util:panel id="_message" title="${message}" openPane="false">
                                <c:out value="${exception.localizedMessage}" />
                        </util:panel>
                        <spring:message var="stacktrace" code="exception_stacktrace"
                                htmlEscape="false" />
                        <util:panel id="_exception" title="${stacktrace}" openPane="false">
                                <c:forEach items="${exception.stackTrace}" var="trace">
                                        <c:out value="${trace}" />
                                        <br />
                                </c:forEach>
                        </util:panel>
                        </p>
                </c:if>
        </util:panel>
</div>
```

13.4.13.1.2 Fichero "dataAccessFailure.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions" xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags" xmlns:util="urn:jsptagdir:/WEB-
INF/tags/jquery/util" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <spring:message code="error_dataaccessfailure_title" htmlEscape="false" var="title"/>
    <util:panel id="title" title="${title}">
        <h2>${fn:escapeXml(title)}</h2>
        <p>
            <spring:message code="error_dataaccessfailure_problemdescription"/>
        </p>
        <c:if test="${not empty exception}">
            <p>
                <h4>
                    <spring:message code="exception_details"/>
                </h4>
                <spring:message code="exception_message" var="message"/>
                <util:panel id="_message" openPane="false" title="${message}">
                    <c:out value="${exception.localizedMessage}"/>
                </util:panel>
                <spring:message code="exception_stacktrace" var="stacktrace"/>
                <util:panel id="_exception" openPane="false" title="${stacktrace}">
                    <c:forEach items="${exception.stackTrace}" var="trace">
                        <c:out value="${trace}"/>
                        <br/>
                    </c:forEach>
                </util:panel>
            </p>
        </c:if>
    </util:panel>
</div>
```

13.4.13.1.3 Fichero "footer.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div id="footer" class="container" xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions"
xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:spring="http://www.springframework.org/tags"
xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <jsp:output omit-xml-declaration="yes"/>

  <spring:url value="/" var="home"/>

  <div class="col-xs-12 col-sm-9">
<!--      <span> -->
<!--        <a href="${home}"> -->
<!--          <spring:message code="button_home"/> -->
<!--        </a> -->
<!--      </span> -->
```

```
          <span><spring:message code="contact_mail"/>: <a
href="mailto:support@mediaserver.com?Subject=MediaserverIssue"
target="_top">support@mediaserver.com</a></span>
          <c:out value=" | "/>
          <spring:message code="last_review"/><c:out value=": 2015/05/19" />
          <!-- Movido al header -->
<!--      <c:if test="${pageContext['request'].userPrincipal != null}">
            <c:out value=" | "/>
            <span>
              <spring:url value="/resources/j_spring_security_logout" var="logout"/>
              <a href="${logout}">
                <spring:message code="security_logout"/>
              </a>
            </span>
            <c:out value=" | "/>
            <span>
            <spring:url value="/passwd" var="passwd"/>
            <a href="${passwd}">
                <spring:message code="change_password"/>
            </a>
        </span>
          </c:if>

          <span id="language">
            <c:out value=" | "/>
            <spring:message code="global_language"/>
            <c:out value=": "/>
          <util:language label="English" locale="en"/>
            <util:language label="Español" locale="es"/>
            </span>-->

        <c:out value=" | " />
          <util:theme/>
 </div>
 <div class="col-xs-12 col-sm-3 hidden-xs">
          <spring:url value="/resources/images/springsource-logo.png" var="logo"/>
          <spring:message code="global_sponsored" htmlEscape="false" var="sponsored"/>
          <span>
            <a href="http://springsource.com" title="${fn:escapeXml(sponsored)}">
              <img  alt="${fn:escapeXml(sponsored)}" src="${logo}"/>
            </a>
          </span>

  </div>
</div>
```

13.4.13.1.4 Fichero "header.jspx":

```
<div id="header" xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:fn="http://java.sun.com/jsp/jstl/functions"
xmlns:c="http://java.sun.com/jsp/jstl/core" xmlns:spring="http://www.springframework.org/tags"
xmlns:security="http://www.springframework.org/security/tags" xmlns:util="urn:jsptagdir:/WEB-
INF/tags/util"
version="2.0">
  <jsp:directive.page contentType="text/html;charset=UTF-8" />
  <jsp:output omit-xml-declaration="yes" />
      <spring:url var="logo_gvnix" value="/resources/images/mediaServerIconLetras.png" />
      <spring:url var="logo_help" value="/resources/images/help.png" />
      <spring:url var="logo_options" value="/resources/images/option.png" />
      <spring:url var="home" value="/" />
      <spring:message code="button_home" var="home_label" htmlEscape="false" />
      <spring:message code="application_name" var="project_name" htmlEscape="false" />
                <div class="pull-left">
                <a id="homeA" href="${home}"  name="${fn:escapeXml(home_label)}"
                title="${fn:escapeXml(home_label)}"><img src="${logo_gvnix}" alt="Logo
mediaserver"/></a>
                </div>
<!--                    <div class="pull-right">
                              <a class="navbar-brand nav navbar-right" href="${home}"
                            name="${fn:escapeXml(home_label)}"

        title="${fn:escapeXml(home_label)}">${fn:escapeXml(project_name)}</a>
                        </div>
```

```
-->
                <div class="pull-right">
                        <ul class="nav navbar-nav" id="ul_derecho">
                                <li class="dropdown">
                                        <a data-toggle="dropdown" class="dropdown-toggle"
href="#" name="Options" title="Options">
                                                <img src="${logo_options}" class="topnav_icon"
alt="options icon"/>
                                        </a>
                                        <ul class="dropdown-menu">
                                                <security:authorize
ifAnyGranted="USER,ADMINISTRATOR">
                                                        <li><spring:url
value="/resources/j_spring_security_logout"
                                                                var="logout" /> <a
href="${logout}"> <spring:message

        code="security_logout" /> <b>
${pageContext['request'].userPrincipal.principal.username}</b>
                                                        </a></li>
                                                        <li><spring:url value="/passwd"
var="passwd" /> <a
                                                                href="${passwd}">
<spring:message code="change_password" />       </a></li>
                                                        </security:authorize>
                                                        <li id="language"><spring:message
code="global_language" />: <util:language
                                                                label=" Change language to
English" locale="en" locale_name="English"/>
                                                                <util:language label="Cambiar
idioma a Español"       locale="es" locale_name="Español"/></li>
                                        </ul>
                                </li>
                                <security:authorize ifAnyGranted="USER,ADMINISTRATOR">
                                <li>
                                <a href="/mediaserver/help" name="Help" title="Help"><img
src="${logo_help}" class="topnav_icon"/></a>
                                </li>
                                </security:authorize>
                        </ul>

                </div>

</div>
```

13.4.13.1.5 Fichero “hlep.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags"
xmlns:util="urn:jsptagdir:/WEB-INF/tags/jquery/util"
version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <spring:message code="application_name" htmlEscape="false" var="app_name"/>
        <div class="jumbotron">
            <h1><spring:message code="hlp1"/></h1>
            <p><spring:message code="hlp2"/></p>
            <p><a href="chrome://flags/#enable-npapi">chrome://flags/#enable-npapi</a></p>
            <p><spring:message code="hlp3"/></p>

            <p><spring:message code="hlp4"/> <a
href="https://drive.google.com/file/d/0B6z1VakuQS8ob0hTUHJCRzYzM0E/view?usp=sharing"><spring:mes
sage code="here"/></a></p>
        </div>

<style type="text/css">
        p {     margin: 10px 0px 10px;}
</style>

</div>
```

13.4.13.1.6 Fichero "index-template.jspx":

```
<div xmlns:spring="http://www.springframework.org/tags" xmlns:util="urn:jsptagdir:/WEB-
INF/tags/util" xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
  <jsp:directive.page contentType="text/html;charset=UTF-8" />
  <jsp:output omit-xml-declaration="yes" />
  <spring:message var="title" htmlEscape="false" />
  <util:panel id="title" title="${title}">
    <spring:message code="application_name" var="app_name" htmlEscape="false" />
    <h3>
      <spring:message code="welcome_titlepane" arguments="${app_name}" />
    </h3>
  </util:panel>
</div>
```

13.4.13.1.7 Fichero "index.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:spring="http://www.springframework.org/tags"
xmlns:util="urn:jsptagdir:/WEB-INF/tags/jquery/util" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <spring:message code="application_name" htmlEscape="false" var="app_name"/>
    <spring:message arguments="${app_name}" code="welcome_titlepane" htmlEscape="false"
var="title"/>
    <spring:message code="get_started" htmlEscape="false" var="get_started"/>
        <div class="jumbotron"> <!-- hidden-xs">-->
            <h1><spring:message arguments="${app_name}" code="welcome_h3"/></h1>
                <spring:url value="/resources/images/mediaServerBig.png" var="logoBig"/>
            <a href="./medias"  title="${get_started}">

            <img  id="logoBig" alt="Big logo Mediaserver" src="${logoBig}"/>
            </a>

            <p><spring:message code="mediaserver_desc"/></p>
            <p><spring:message code="mediaserver_desc2"/></p>
            <p><spring:message code="mediaserver_desc3"/></p>
                        <spring:message code="url_app" var="url_app"/>
                        <a href="${url_app}">
                    <div id="anuncioDescarga" class="visible-xs">

                        <p><spring:message code="not_yet_app"/></p>
                        <p id="obtener"><spring:message code="get_it"/> <spring:message
code="here"/></p>

                                </div>
                        </a>
        </div>
<script type="text/javascript">
($( "#homeA" ).css('cursor', 'default')).click(function( event ) {
  event.preventDefault();
});
</script>
<style type="text/css">
p{margin: 10px 0px 10px;}
#anuncioDescarga {
        background-color: rgba(239, 253, 28, 0.99);
        border-radius: 2em;
        clear: both;
        max-width: 20em;
        font-size: 2em;
        border-color: rgb(255, 174, 0);
        font-weight: 700;
        text-align: center;
        padding: 0.5em;
        border: solid;
        margin: 1em auto;
        color: black;
}
</style>
</div>
```

13.4.13.1.8 Fichero "login.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div  class="well" xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions" xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags" xmlns:util="urn:jsptagdir:/WEB-
INF/tags/jquery/util" xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
version="2.0">
  <jsp:directive.page contentType="text/html;charset=UTF-8"/>
  <jsp:output omit-xml-declaration="yes"/>

    <c:if test="${not empty param.login_error}">
      <div class="alert alert-danger">
        <p>
          <spring:message code="security_login_unsuccessful"/>
          <c:out value="${SPRING_SECURITY_LAST_EXCEPTION.message}"/>
          .
        </p>
      </div>
    </c:if>

        <c:if test="${not empty param.activated}">
      <div class="alert alert-info">
              <p><spring:message code="security_user_activated"/></p>
        </div>
    </c:if>

    <c:if test="${empty param.login_error and empty param.activated}">
      <div class="alert alert-info">
              <p><spring:message code="security_login_message"/></p>
        </div>
    </c:if>

        <spring:url value="/resources/j_spring_security_check" var="form_url"/>

        <form action="${fn:escapeXml(form_url)}" method="POST" name="f" class="form-signin">

        <h2 class="form-signin-heading"><spring:message code="security_login_title"/></h2>

                <spring:message code="security_login_form_name" var="name" />
                <spring:message code="security_login_form_name_message" htmlEscape="false"
var="name_msg"/>
                <input class="form-control" type="text" placeholder="${name}"
autofocus="autofocus" id="j_username" name="j_username" data-required="true" data-minlength="3"
data-maxlength="30" data-original-title="${name_msg}" />
                <!-- <script type="text/javascript">
                  <c:set var="sec_name_msg">
                    <spring:escapeBody javaScriptEscape="true">${name_msg}</spring:escapeBody>
                  </c:set>
                  Spring.addDecoration(new Spring.ElementDecoration({elementId : "j_username",
widgetType : "dijit.form.ValidationTextBox", widgetAttrs : {promptMessage: "${sec_name_msg}",
required : true}}));
                </script> -->

                <spring:message code="security_login_form_password" var="pass"/>
        <spring:message code="security_login_form_password_message" htmlEscape="false"
var="pwd_msg"/>
        <input type="password" class="form-control" placeholder="${pass}" id="j_password"
name="j_password" data-required="true" data-minlength="3" data-maxlength="30" data-original-
title="${pwd_msg}" />
                <!-- <script type="text/javascript">
                  <c:set var="sec_pwd_msg">
                    <spring:escapeBody javaScriptEscape="true">${pwd_msg}</spring:escapeBody>
                  </c:set>
                  Spring.addDecoration(new Spring.ElementDecoration({elementId : "j_password",
widgetType : "dijit.form.ValidationTextBox", widgetAttrs : {promptMessage: "${sec_pwd_msg}",
required : true}}));
                </script> -->

        <div class="submit">
        <script type="text/javascript">Spring.addDecoration(new
Spring.ValidateAllDecoration({elementId:'proceed', event:'onclick'}));</script>
            <button type="submit" class="btn btn-primary btn-lg btn-block"
id="proceed"><spring:message code="button_signin"/></button>
```

```
                        <button type="reset" class="btn btn-default btn-sm btn-block"
id="reset"><spring:message code="button_reset"/></button>

        </div>
<br/>
            <div>
                <span>
                    <spring:url value="/forgot" var="forgot"/>
                    <a href="${forgot}">
                        <spring:message code="forgot_password"/>
                    </a>
                </span>
                <c:out value=" | "/>
                <span>
                    <spring:message code="not_a_user_yet"/>
                    <c:out value=" "/>
                    <spring:url value="/signup" var="signup"/>
                    <a href="${signup}">
                        <spring:message code="sign_up"/>
                    </a>
                </span>
            </div>

      </form>

</div>
```

13.4.13.1.9 Fichero “menú.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:menu="urn:jsptagdir:/WEB-INF/tags/menu"
xmlns:security="http://www.springframework.org/security/tags"
id="menu" version="2.0" >
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>

            <menu:menu id="_menu" z="nZaf43BjUg1iM0v70HJVEsXDopc=">
                    <security:authorize ifAnyGranted="USER,ADMINISTRATOR">
                        <menu:category id="c_media" z="SqacaDVnSrXmQJB69xz5K1fP2Fs=">
                            <menu:item id="i_media_new" messageCode="global_menu_new"
url="/medias?form" z="OIfbj50qi1kFkSFTSJLE93WzBC0="/>
                            <menu:item id="i_media_list" messageCode="global_menu_list"
url="/medias?page=1&amp;size=${empty param.size ? 10 : param.size}"
z="VgX3ic5+KXqm7H7gzcyK4PHeDvg="/>
                                <menu:item id="fi_media_type" messageCode="global_menu_find"
url="/medias?find=ByType&amp;form&amp;page=1&amp;size=${empty param.size ? 10 : param.size}"
z="s7IuHZH8i8xjL3dTe6fH/h3HDZ4="/>

                        </menu:category>
                        <menu:category id="c_playlist" z="user-managed">
                            <menu:item id="i_playlist_new" messageCode="global_menu_new"
url="/playlists?form" z="user-managed"/>
                            <menu:item id="i_playlist_list" messageCode="global_menu_list"
url="/playlists?page=1&amp;size=${empty param.size ? 10 : param.size}" z="user-managed"/>
                        </menu:category>
                    </security:authorize>

                    <security:authorize ifNotGranted="USER,ADMINISTRATOR">
                         <menu:item id="i_signin" messageCode="button_signin" url="/login"
z="user-managed"/>
                    </security:authorize>

                <security:authorize ifAllGranted="ADMINISTRATOR">
                        <menu:category id="c_playlistmedia" z="user-managed">
                            <menu:item id="i_playlistmedia_new" messageCode="global_menu_new"
url="/playlistmedias?form" z="user-managed"/>
                            <menu:item id="i_playlistmedia_list" messageCode="global_menu_list"
url="/playlistmedias?page=1&amp;size=${empty param.size ? 10 : param.size}" z="user-managed"/>
                        </menu:category>

                        <menu:category id="c_user" z="uoTSf/uIZ821ZppidzS9aO776yk=">
                            <menu:item id="i_user_new" messageCode="global_menu_new"
url="/security/users?form" z="c85KUEJlx3IraUTnttCLgXcDCQ0="/>
```

```
                            <menu:item id="i_user_list" messageCode="global_menu_list"
url="/security/users?page=1&amp;size=${empty param.size ? 10 : param.size}"
z="KgqH4ORr9lFHMlXLUaE+G8A2Evc="/>
                        </menu:category>
                        <menu:category id="c_role" render="false" z="user-managed">
                            <menu:item id="i_role_new" messageCode="global_menu_new"
url="/security/roles?form" z="dgsLLIun+bcm3IZkzkc6kSms+Uk="/>
                            <menu:item id="i_role_list" messageCode="global_menu_list"
url="/security/roles?page=1&amp;size=${empty param.size ? 10 : param.size}"
z="liweb2IoVTDM/V4zpp61Ic8z9D8="/>
                        </menu:category>
                        <menu:category id="c_userrole" z="C4C1ZBdes4PUIjEM3Ajwwhn1EwA=">
                            <menu:item id="i_userrole_new" messageCode="global_menu_edit"
url="/security/assignrole?form" z="y9WtUvel50F2A4VOVOeY7PVEYhk="/>
                            <menu:item id="i_userrole_list" messageCode="global_menu_list"
url="/security/assignrole?page=1&amp;size=${empty param.size ? 10 : param.size}"
z="P6AEL8JMAbKhOIlyw65OJwVYiww="/>
                        </menu:category>
                    </security:authorize>
                </menu:menu>

</div>
```

13.4.13.1.10 Fichero “resourceNotFound.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags"
xmlns:util="urn:jsptagdir:/WEB-INF/tags/jquery/util" version="2.0">

    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <spring:message code="error_resourcenotfound_title" htmlEscape="false" var="title"/>
    <util:panel id="title" title="${title}">
        <h2>${fn:escapeXml(title)}</h2>
        <p>
            <spring:message code="error_resourcenotfound_problemdescription"/>
        </p>
        <c:if test="${not empty exception}">
            <p>
                <h4>
                    <spring:message code="exception_details"/>
                </h4>
                <spring:message code="exception_message" htmlEscape="false" var="message"/>
                <util:panel id="_message" openPane="false" title="${message}">
                    <c:out value="${exception.localizedMessage}"/>
                </util:panel>
                <spring:message code="exception_stacktrace" htmlEscape="false"
var="stacktrace"/>
                <util:panel id="_exception" openPane="false" title="${stacktrace}">
                    <c:forEach items="${exception.stackTrace}" var="trace">
                        <c:out value="${trace}"/>
                        <br/>
                    </c:forEach>
                </util:panel>
            </p>
        </c:if>
    </util:panel>
</div>
```

13.4.13.1.11 Fichero “uncaughtException.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:fn="http://java.sun.com/jsp/jstl/functions" xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags" xmlns:util="urn:jsptagdir:/WEB-
INF/tags/jquery/util" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
```

```
    <spring:message code="error_uncaughtexception_title" htmlEscape="false" var="title"/>
    <util:panel id="title" title="${title}">
        <!-- repeat <h2> in util/panel.tagx -->
        <!-- <h2>${fn:escapeXml(title)}</h2> -->
        <p>
            <spring:message code="error_uncaughtexception_problemdescription"/>
        </p>
        <c:if test="${not empty exception}">
            <p>
                <h4>
                    <spring:message code="exception_details"/>
                </h4>
                <spring:message code="exception_message" htmlEscape="false" var="message"/>
                <util:panel id="_message" openPane="false" title="${message}">
                    <c:out value="${exception.localizedMessage}"/>
                </util:panel>
                <spring:message code="exception_stacktrace" htmlEscape="false"
var="stacktrace"/>
                <util:panel id="_exception" openPane="false" title="${stacktrace}">
                    <c:forEach items="${exception.stackTrace}" var="trace">
                        <c:out value="${trace}"/>
                        <br/>
                    </c:forEach>
                </util:panel>
            </p>
        </c:if>
    </util:panel>
</div>
```

13.4.13.1.12 Fichero “views.xml”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
        <definition extends="default" name="index">
                <put-attribute name="body" value="/WEB-INF/views/index.jspx" />
        </definition>
        <definition extends="public" name="dataAccessFailure">
                <put-attribute name="body"      value="/WEB-INF/views/dataAccessFailure.jspx" />
        </definition>
        <definition extends="public" name="resourceNotFound">
                <put-attribute name="body"      value="/WEB-INF/views/resourceNotFound.jspx" />
        </definition>
        <definition extends="public" name="uncaughtException">
                <put-attribute name="body"      value="/WEB-INF/views/uncaughtException.jspx" />
        </definition>
        <definition extends="public" name="login">
                <put-attribute name="body" value="/WEB-INF/views/login.jspx" />
        </definition>
        <definition extends="public" name="accessDenied">
                <put-attribute name="body" value="/WEB-INF/views/accessDeniedException.jspx"/>
        </definition>
        <definition extends="default" name="help">
                <put-attribute name="body" value="/WEB-INF/views/help.jspx" />
        </definition>
</tiles-definitions>
```

13.4.13.2 Fichero “”:

13.4.13.3 Fichero “”:

13.4.13.4 Directorio change_password

13.4.13.4.1 Fichero "form.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:field="urn:jsptagdir:/WEB-INF/tags/form/fields"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:fn="http://java.sun.com/jsp/jstl/functions"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_change_password" htmlEscape="false"
                var="title" />
        <util:panel id="title" title="${title}">

                <form:form action="" method="POST" commandName="form">
                        <field:input field="oldPassword" id="c_current_password" min="1"
                                required="true" z="?" type="password" />
                        <field:input field="newPassword" id="c_new_password" min="1"
                                required="true" z="?" type="password" />
                        <field:input field="retypeNewPassword" id="c_retype_new_password"
                                min="1" required="true" type="password" z="?" />
                        <div class="submit">
                                <script type="text/javascript">
                                        Spring.addDecoration(new Spring.ValidateAllDecoration({
                                                elementId : 'proceed',
                                                event : 'onclick'
                                        }));
                                </script>
                                <spring:message code="button_submit" htmlEscape="false"
                                        var="submit_label" />
                                <input id="proceed" type="submit"
                                        value="${fn:escapeXml(submit_label)}" />
                        </div>
                </form:form>

        </util:panel>
        <script type="text/javascript">
        $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                $("#form").attr("class","form-horizontal");

                $("#form > div").attr("class","control-group form-group");
                $("#form > div > label").attr("class","control-label col-xs-5 col-sm-4 col-md-3
col-lg-3");
                $(".dijitInputInner").attr("class","form-control input-sm");
                $("#proceed").attr("class","btn btn-primary btn-block");
//              $("#").attr("class","");
                $(".errors").attr("class","alert-danger");
        });
        </script>
        <style>
                 #form, #form >div > div { margin: auto;  width: 60%;  min-width: 400px;}
                 #recaptcha_table{margin:auto!important;}
                 .alert-danger {  border-radius: 4px;
                          }
        </style>
</div>
```

13.4.13.4.2 Fichero "thanks.jspx":

```
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:jsp="http://java.sun.com/JSP/Page"
```

```
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_change_password" htmlEscape="false"
                var="title" />
        <util:panel id="title" title="${title}">
                <h2>
                        <spring:message code="header_password_changed" />
                </h2>
                <p>
                        <spring:message code="desc_password_changed" />
                </p>
        </util:panel>
        <script type="text/javascript">
                $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                });
        </script>
</div>
```

13.4.13.4.3 Fichero “views.xml”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
        <definition extends="default" name="change_password/form">
                <put-attribute name="body"
                        value="/WEB-INF/views/change_password/form.jspx" />
        </definition>
        <definition extends="default" name="change_password/thanks">
                <put-attribute name="body"
                        value="/WEB-INF/views/change_password/thanks.jspx" />
        </definition>
</tiles-definitions>
```

13.4.13.5 Directorio forgot_password

13.4.13.5.1 Fichero “form.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:field="urn:jsptagdir:/WEB-INF/tags/form/fields"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:fn="http://java.sun.com/jsp/jstl/functions"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_forgot_password" htmlEscape="false"
                var="title" />
        <util:panel id="title" title="${title}">

                <form:form action="" method="POST" commandName="form">
                        <field:input field="emailAddress" id="c_email" min="1"
                                required="true" z="?" />
                        <div class="submit">
                                <script type="text/javascript">
                                        Spring.addDecoration(new Spring.ValidateAllDecoration({
                                                elementId : 'proceed',
                                                event : 'onclick'
                                        }));
```

```
                                </script>
                                <spring:message code="button_submit" htmlEscape="false"
                                        var="submit_label" />
                                <input id="proceed" type="submit"
                                        value="${fn:escapeXml(submit_label)}" />
                        </div>
                </form:form>

        </util:panel>
        <script type="text/javascript">
                $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                $("#form").attr("class","form-horizontal");

                $("#form > div").attr("class","control-group form-group");
                $("#form > div > label").attr("class","control-label col-xs-5 col-sm-4 col-md-3
col-lg-3");
                $(".dijitInputInner").attr("class","form-control input-sm");
                $("#proceed").attr("class","btn btn-primary btn-block");
//              $("#").attr("class","");
                $(".errors").attr("class","alert-danger");
                });
        </script>
        <style>
                 #form, #form >div > div { margin: auto;  width: 60%;  min-width: 400px;}
                 #recaptcha_table{margin:auto!important;}
                 .alert-danger {  border-radius: 4px;
                          }
        </style>
</div>
```

13.4.13.5.2 Fichero “set_new.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:field="urn:jsptagdir:/WEB-INF/tags/form/fields"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:fn="http://java.sun.com/jsp/jstl/functions"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_forgot_password" htmlEscape="false"
                var="title" />
        <util:panel id="title" title="${title}">

                <form:form action="" method="POST" commandName="form">
                        <field:input field="newPassword" id="c_new_password" min="1"
                                required="true" z="?" type="password" />
                        <field:input field="retypeNewPassword" id="c_retype_new_password"
                                min="1" required="true" z="?" type="password" />
                        <div class="submit">
                                <script type="text/javascript">
                                        Spring.addDecoration(new Spring.ValidateAllDecoration({
                                                elementId : 'proceed',
                                                event : 'onclick'
                                        }));
                                </script>
                                <spring:message code="button_submit" htmlEscape="false"
                                        var="submit_label" />
                                <input id="proceed" type="submit"
                                        value="${fn:escapeXml(submit_label)}" />
                        </div>
                </form:form>

        </util:panel>
        <script type="text/javascript">
                $(window).load(function() {
```

```
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                $("#form").attr("class","form-horizontal");
                $("#form > div").attr("class","control-group form-group");
                $("#form > div > label").attr("class","control-label col-xs-5 col-sm-4 col-md-3
col-lg-3");
                $(".dijitInputInner").attr("class","form-control input-sm");
                $("#proceed").attr("class","btn btn-primary btn-block");
                //$(".errors").attr("class","alert-danger");
        });
        </script>
        <style>
                 #form, #form >div > div { margin: auto;  width: 60%;  min-width: 400px;}
                 #recaptcha_table{margin:auto!important;}
                 .alert-danger { border-radius: 4px; }
        </style>
</div>
```

13.4.13.5.3 Fichero “thanks.jspx”:

```
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_forgot_password" htmlEscape="false"
                var="title" />
        <util:panel id="title" title="${title}">
                <h2>
                        <spring:message code="header_reset_password_sent" />
                </h2>
                <p>
                        <spring:message code="desc_reset_password_sent" />
                </p>
        </util:panel>
        <script type="text/javascript">
                $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                });
        </script>
</div>
```

13.4.13.5.4 Fichero “views.xml”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
        <definition extends="public" name="forgot_password/form">
                <put-attribute name="body"
                        value="/WEB-INF/views/forgot_password/form.jspx" />
        </definition>
        <definition extends="public" name="forgot_password/thanks">
                <put-attribute name="body"
                        value="/WEB-INF/views/forgot_password/thanks.jspx" />
        </definition>
        <definition extends="public" name="forgot_password/set_new">
                <put-attribute name="body"
                        value="/WEB-INF/views/forgot_password/set_new.jspx" />
        </definition>
</tiles-definitions>
```

13.4.13.6 Directorio medias

13.4.13.6.1 Fichero “create.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core" xmlns:field="urn:jsptagdir:/WEB-
INF/tags/jquery/form/fields"
xmlns:form="urn:jsptagdir:/WEB-INF/tags/datatables" xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <form:create id="fc_com_mediaserver_domain_Media" modelAttribute="media" path="/medias"
render="${empty dependencies}" z="xv5TGJQrC4ssYNHGJm8ORIP7+1o=">
        <h4><spring:message code="search_media"/></h4>
                <div id="busqueda">
                        <script type="text/javascript">
                        //funciona correctamente, pero el inspector muestra un error 404, bug
reportado en

        //https://productforums.google.com/forum/#!topic/customsearch/9WJ2G1rJXWE
                          (function() {
                            var cx = '015143781089279757786:9g9l0v8g6wc';
                            var gcse = document.createElement('script');
                            gcse.type = 'text/javascript';
                            gcse.async = true;
                            gcse.src = (document.location.protocol == 'https:' ? 'https:' :
'http:') +
                                '//cse.google.com/cse.js?cx=' + cx;
                            var s = document.getElementsByTagName('script')[0];
                            s.parentNode.insertBefore(gcse, s);

                          })();

                          //asociaremos el evento a todos los elementos de clase gs-title que se
creen dentro del div #busqeuda
                          $("#busqueda").on("click", "a.gs-title", function (event) {
                                  event.preventDefault();
                                  var enlace = $(event.target);
                                  if($(event.target).is("b")){//si es un elemento negrita, el
enlace completo es el elemeto padre
                                          enlace= enlace.parent();
                                  }
                                  var nombre=enlace.text().split(" - ")[0];
                                  if(nombre.length > 30) {
                                            nombre = nombre.substring(0,29);
                                         }
                                  $("#_name_id").val(nombre).change();
                                  $("#_source_id").val(enlace.attr("data-ctorig")).change();
                                  //reocultar el overlay de busqueda
                                  $(".gsc-results-wrapper-overlay").removeClass("gsc-results-
wrapper-visible");
                                  $(".gsc-modal-background-image").removeClass("gsc-modal-
background-image-visible");
                                  });
                          $('script [type="text/javascript",
src="http://www.google.com/uds/?file=ads&amp;v=3&amp;packages=search&amp;async=2"]').remove();
                          $("#busqueda").on("DOMContentLoaded", "input.gsc-search-button",
function (event) {
                                  $(event.target).removeClass(gsc-search-button-v2);

                          }

                          );

                          </script>
                        <div class="gcse-search"/>
                </div>

        <field:input field="name" id="c_com_mediaserver_domain_Media_name" max="30" min="3"
required="true" z="nF5o63Br9aZnaOjUgMt9kXMBQew="/>
        <field:input field="source" id="c_com_mediaserver_domain_Media_source"
z="EBWy6sLJ1GRopANVz8raxyPVrz4="/>
```

```
        <field:select field="type" id="c_com_mediaserver_domain_Media_type"
items="${mediatypes}" path="mediatypes" z="ZXvnReyBcwbw5YbWQjR021M7UOU="/>
    </form:create>
    <form:dependency dependencies="${dependencies}" id="d_com_mediaserver_domain_Media"
render="${not empty dependencies}" z="uVNjSkHPchisiH+IW7c2piArgIU="/>

        <style type="text/css">
                .control-label{width:33%;text-align: left !important;}
                .controls{width:99% !important;}
                #proceed { margin: 22px 0;}
                #_c_com_mediaserver_domain_Media_name_id {width:27% ;float:left;}
                #_c_com_mediaserver_domain_Media_source_id {width:42% ;float:left;}
                #_c_com_mediaserver_domain_Media_type_id {width:27% ;float:left;}
                #fc_com_mediaserver_domain_Media_submit{width:100px; ;float:left;padding: 0px
10px}
                #busqueda > br{display:none}
                #command{width:400px ; margin: 0 auto;}
    </style>

</div>
```

13.4.13.6.2 Fichero "findMediasByType.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:form="urn:jsptagdir:/WEB-INF/tags/jquery/form"
xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <form:find finderName="ByType" id="ff_com_mediaserver_domain_Media" path="/medias"
z="TcgDTRuS4q8rQWDGBBtMnjLql3s=">
        <field:select disableFormBinding="true" field="type"
id="f_com_mediaserver_domain_Media_type"
        items="${mediatypes}" path="/mediatypes" required="true"
z="m8XVOTXF5boKTgzWNvW3dEg+uO4="/>
    </form:find>

    <style type="text/css">
     #ff_com_mediaserver_domain_Media_ByType_submit{width:100px; float:left; padding: 0px 10px;}
     #_f_com_mediaserver_domain_Media_type_id{width:250px ;float:left;}
     #_f_com_mediaserver_domain_Media_type_id > label{width:70x;}
    #command > br{display:none}
     #command{width:400px ;margin: 0 auto;}
    </style>

</div>
```

13.4.13.6.3 Fichero "list.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-
INF/tags/datatables"
xmlns:table="urn:jsptagdir:/WEB-INF/tags/datatables" version="2.0"
xmlns:spring="http://www.springframework.org/tags">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>


    <page:list id="pl_com_mediaserver_domain_Media" items="${medias}"
z="FApsqlMxYYQuB+0RBGUxRjk/r2U=">

        <h4><spring:message code="search_media"/></h4>
                <div id="busqueda">
                        <script>
                        //funciona correctamente, pero el inspector muestra un error 404, bug
reportado en

        //https://productforums.google.com/forum/#!topic/customsearch/9WJ2G1rJXWE
                          (function() {
                            var cx = '015143781089279757786:9g9l0v8g6wc';
                            var gcse = document.createElement('script');
                            gcse.type = 'text/javascript';
```

```
                            gcse.async = true;
                            gcse.src = (document.location.protocol == 'https:' ? 'https:' :
'http:') +
                                '//cse.google.com/cse.js?cx=' + cx;
                            var s = document.getElementsByTagName('script')[0];
                            s.parentNode.insertBefore(gcse, s);

                          })();
                          //asociaremos el evento a todos los elementos de clase gs-title que se
creen dentro del div #busqeuda
                          $("#busqueda").on("click", "a.gs-title", function (event) {
                                  event.preventDefault();
                                  var enlace = $(event.target);
                                  if($(event.target).is("b")){//si es un elemento negrita, el
enlace completo es el elemeto padre
                                          enlace= enlace.parent();
                                  }
                                  var nombre=enlace.text().split(" - ")[0];
                                  if(nombre.length > 30) {
                                          nombre = nombre.substring(0,29);
                                      }
                                  $("#_name_id_create_0").val(nombre).change();
                                  $("#_source_id_create_0").val(enlace.attr("data-
ctorig")).change();
                                  //reocultar el overlay de busqueda
                                  $(".gsc-results-wrapper-overlay").removeClass("gsc-results-
wrapper-visible");
                                  $(".gsc-modal-background-image").removeClass("gsc-modal-
background-image-visible");
                                  });
                          $('script [type="text/javascript",
src="http://www.google.com/uds/?file=ads&amp;v=3&amp;packages=search&amp;async=2"]').remove();
                          $("#busqueda").on("DOMContentLoaded", "input.gsc-search-button",
function (event) {
                                  $(event.target).removeClass(gsc-search-button-v2);
                          });
                          </script>
                        <div class="gcse-search"/>
                </div>


        <table:table data="${medias}" id="l_com_mediaserver_domain_Media" path="/medias"
z="zcc+YvVzEHqSWaxsXNGEMGBlTM0=">
            <table:column id="c_com_mediaserver_domain_Media_name" property="name"
z="lANO9VHBvsSd02LQWp1Qx1kCP1U="/>
            <table:column id="c_com_mediaserver_domain_Media_source" property="source"
z="4x+MSNdO7GtcvzWC1vlR4KJqC98="/>
            <table:column id="c_com_mediaserver_domain_Media_type" property="type"
z="e3Xf7Ckj7MV5/DxMdF5ke1zJ79Y="/>
        </table:table>
    </page:list>


</div>
```

13.4.13.6.4 Fichero “show.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:page="urn:jsptagdir:/WEB-INF/tags/datatables"
xmlns:spring="http://www.springframework.org/tags" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>

    <page:show id="ps_com_mediaserver_domain_Media" object="${media}" path="/medias"
z="aQfEbfwwoLtps2gZ1aPhyejZl0k=">
        <field:display field="name" id="s_com_mediaserver_domain_Media_name" object="${media}"
z="7wY2dRijzfFQRxKfHhKNwDzKgCs="/>
        <field:display field="source" id="s_com_mediaserver_domain_Media_source"
object="${media}" z="+6+H9KhWHq+jeCNr78VG3XnsCsY="/>
        <field:display field="type" id="s_com_mediaserver_domain_Media_type" object="${media}"
z="WPxh8gT0FCzUHDEw2Egw7EysvfM="/>
```

```
        <p id="pPlayer" align="center"><b><spring:message code="player_status"/></b><span
id="textoEstado">stop</span> <span id="medioActual"></span>
        </p>

        <div id="player">
                <embed type="application/x-vlc-plugin"
                                pluginspage="http://www.videolan.org" width="640" height="480"
                                id="vlc" align="center" version="VideoLAN.VLCPlugin.2" />
                        <object classid="clsid:9BE31822-FDAD-461B-AD51-BE1D1C159921"

       codebase="http://download.videolan.org/pub/videolan/vlc/last/win32/axvlc.cab" />

                        <br />
                        <div id="botones">
<!--            <button class="bprev" onclick="sendMessageToServer('prev')">prev</button> -->
                                <button class="bplay" onclick="sendMessageToServer('play')"
title="play"></button>
                                <button class="bstop" onclick="sendMessageToServer('stop')"
title="stop"></button>
                                <button class="bpause" onclick="sendMessageToServer('pause')"
title="pause"></button>
<!--                    <button class="bnext"
onclick="sendMessageToServer('next')">next</button> -->
                        </div>
                </div>

    </page:show>
<script type="text/javascript">
var username="${pageContext.request.userPrincipal.name}";
var socketDest='/mediaserver/simplemessages';
$(document).ready(function() {
        connect();
});

window.onload = function set() {//$( document ).ready no funciona así para chrome
        var nombreReproductor = "vlc";
        var uri = ""; //${media.source}";

        var tgt = document.getElementById(nombreReproductor);

        tgt.playlist.add(uri, uri, "");

        initPlayer();
        vlc.playlist.add("${media.source}");
        comprobarOrden("${reproductor.estado}");
};

</script>
<style type="text/css">
#medioActual{
        display: none;
}
</style>
</div>
```

13.4.13.6.5 Fichero “views.xml”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
    <definition extends="default" name="medias/list">
        <put-attribute name="body" value="/WEB-INF/views/medias/list.jspx"/>
    </definition>
<definition extends="default" name="medias/show">
        <put-attribute name="body" value="/WEB-INF/views/medias/show.jspx"/>
    </definition>
<definition extends="default" name="medias/create">
        <put-attribute name="body" value="/WEB-INF/views/medias/create.jspx"/>
    </definition>
<definition extends="default" name="medias/update">
        <put-attribute name="body" value="/WEB-INF/views/medias/update.jspx"/>
```

```
    </definition>
<definition extends="default" name="medias/findMediasByType">
        <put-attribute name="body" value="/WEB-INF/views/medias/findMediasByType.jspx"/>
    </definition>
</tiles-definitions>
```

13.4.13.6.6 Fichero "":

13.4.13.7 Directorio playlistmedias

13.4.13.7.1 Fichero "list.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-
INF/tags/datatables"
xmlns:table="urn:jsptagdir:/WEB-INF/tags/datatables" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:list id="pl_com_mediaserver_domain_PlayListMedia" items="${playlistmedias}"
z="BHf6fmCsBFK3JnClK+FCU404MD4=">
        <table:table data="${playlistmedias}" id="l_com_mediaserver_domain_PlayListMedia"
path="/playlistmedias" z="VFfcIz/g0MLNV+YANZOu83BQgTE=">
            <table:column id="c_com_mediaserver_domain_PlayListMedia_playList"
property="playList" z="j3/2NDDQmZ5oEeTEl1ryspUs6RM="/>
            <table:column id="c_com_mediaserver_domain_PlayListMedia_media" property="media"
z="XVaGseFlfCNQty3QQYvAK+/xCr8="/>
            <table:column id="c_com_mediaserver_domain_security_User_emailAddress"
property="playList.owner.emailAddress" z="user-managed"/>
        </table:table>
    </page:list>
    <!-- ocultar el filtro -->
    <style>div.dataTables_filter label {
  display: none;}</style>
</div>
```

13.4.13.7.2 Fichero "show.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:page="urn:jsptagdir:/WEB-INF/tags/datatables" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:show id="ps_com_mediaserver_domain_PlayListMedia" object="${playlistmedia}"
path="/playlistmedias" z="hfYQQ/3X1RW8LVqFmViL7Qk/NbI=">
        <field:display field="playList" id="s_com_mediaserver_domain_PlayListMedia_playList"
object="${playlistmedia}" z="7IQSMeNW7PniRt6H3UbMGvlRleo="/>
        <field:display field="media" id="s_com_mediaserver_domain_PlayListMedia_media"
object="${playlistmedia}" z="d9YIJ1j7McLMijoaoLZSScsF6is="/>
    </page:show>
</div>
```

13.4.13.7.3 "views.xml":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
```

```
<tiles-definitions>
    <definition extends="default" name="playlistmedias/list">
        <put-attribute name="body" value="/WEB-INF/views/playlistmedias/list.jspx"/>
    </definition>
<definition extends="default" name="playlistmedias/show">
        <put-attribute name="body" value="/WEB-INF/views/playlistmedias/show.jspx"/>
    </definition>
<definition extends="default" name="playlistmedias/create">
        <put-attribute name="body" value="/WEB-INF/views/playlistmedias/create.jspx"/>
    </definition>
<definition extends="default" name="playlistmedias/update">
        <put-attribute name="body" value="/WEB-INF/views/playlistmedias/update.jspx"/>
    </definition>
<definition extends="default" name="playlistmedias/findPlayListMediasByPlayList">
        <put-attribute name="body" value="/WEB-
INF/views/playlistmedias/findPlayListMediasByPlayList.jspx"/>
    </definition>
<definition extends="default" name="playlistmedias/findPlayListMediasByMedia">
        <put-attribute name="body" value="/WEB-
INF/views/playlistmedias/findPlayListMediasByMedia.jspx"/>
    </definition>
</tiles-definitions>
```

13.4.13.8 Directirio playlists

13.4.13.8.1 Fichero “create.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core" xmlns:field="urn:jsptagdir:/WEB-
INF/tags/jquery/form/fields" xmlns:form="urn:jsptagdir:/WEB-INF/tags/datatables"
xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:spring="http://www.springframework.org/tags"
version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <form:create id="fc_com_mediaserver_domain_PlayList" modelAttribute="playList"
path="/playlists" render="${empty dependencies}" z="nNrDoEdQeMbakjmgE7yXw34x2Vo=">
        <field:input field="playListName" id="c_com_mediaserver_domain_PlayList_playListName"
max="30" min="3" required="true" z="V4A1wJo0leqHgzrs7a5g06I9aS0="/>
    </form:create>
    <form:dependency dependencies="${dependencies}" id="d_com_mediaserver_domain_PlayList"
render="${not empty dependencies}" z="yylSp+jfrbFwIET+v/R2WBbO+Wg="/>

        <style type="text/css">
                #_c_com_mediaserver_domain_PlayList_playListName_id {width:80% ;float:left;}
                #fc_com_mediaserver_domain_PlayList_submit {width:100px; ;float:left;padding:
0px 10px}
                #playList > br{display:none}
    </style>

</div>
```

13.4.13.8.2 Fichero “list.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-
INF/tags/datatables" xmlns:table="urn:jsptagdir:/WEB-INF/tags/datatables" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:list id="pl_com_mediaserver_domain_PlayList" items="${playlists}"
z="sQsaY/PNnlsiadbm4wNRl4UFnX4=">
        <table:table data="${playlists}" id="l_com_mediaserver_domain_PlayList"
path="/playlists" z="Q3ProbwxP9aG/7JonwK9BmAlbpY=">
            <table:column id="c_com_mediaserver_domain_PlayList_playListName"
property="playListName" z="u/n3eIsa+knCuJrvSLpkFPl8Tro="/>
        </table:table>
    </page:list>
</div>
```

13.4.13.8.3 Fichero "show.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:page="urn:jsptagdir:/WEB-INF/tags/datatables"
        xmlns:table="urn:jsptagdir:/WEB-INF/tags/datatables"
        xmlns:spring="http://www.springframework.org/tags"
        version="2.0">
        <jsp:directive.page contentType="text/html;charset=UTF-8" />
        <jsp:output omit-xml-declaration="yes" />
        <spring:message code="button_add" htmlEscape="false" var="button_add"/>
        <spring:message code="add_medias_to_playlist" htmlEscape="false"
var="add_medias_to_playlist"/>

        <page:show id="ps_com_mediaserver_domain_PlayList" object="${playlist}"
                path="/playlists" z="/KDv2yqjDqgQ9jd8GgRSAvBbzxI=">
                <field:display field="playListName"
                        id="s_com_mediaserver_domain_PlayList_playListName"
                        object="${playlist}" z="veqzQDIg0n3b7v5XIRTNijzO5KU=" />

                <p id="pPlayer" align="center"><b><spring:message
code="player_status"/></b><span id="textoEstado">stop</span> <span id="medioActual"></span>
                 </p>
                 <div>
                <div id="player">
                        <embed type="application/x-vlc-plugin"
                                pluginspage="http://www.videolan.org" width="640" height="480"
                                id="vlc" align="center" version="VideoLAN.VLCPlugin.2" />
                        <object classid="clsid:9BE31822-FDAD-461B-AD51-BE1D1C159921"

        codebase="http://download.videolan.org/pub/videolan/vlc/last/win32/axvlc.cab" />

                        <br />
                        <div id="botones">
                                <button class="bprev" onclick="sendMessageToServer('prev')"
title="prev"></button>
                                <button class="bplay" onclick="sendMessageToServer('play')"
title="play"></button>
                                <button class="bstop" onclick="sendMessageToServer('stop')"
title="stop"></button>
                                <button class="bpause" onclick="sendMessageToServer('pause')"
title="pause"></button>
                                <button class="bnext" onclick="sendMessageToServer('next')"
title="next"></button>
                        </div>
                </div>
        </div>
        </page:show>
        <script type="text/javascript">
                var username = "${pageContext.request.userPrincipal.name}";
                var socketDest = '/mediaserver/simplemessages';
                $(document).ready(function() {
                        connect();
                        adaptarEdicionInline();
                });

                window.onload = function set() {//$( document ).ready no funciona así para
chrome
                        var nombreReproductor = "vlc";
                        var uri = "${media.source}";

                        var tgt = document.getElementById(nombreReproductor);

                        tgt.playlist.add(uri, uri, "");

                        initPlayer();
                        initPlayList();
                };

                function adaptarEdicionInline(){
                        $("#player").on("inline_update_ready", "#_media_id_create_0", function
(event) {
                                //ordenar opciones del select
```

```
                                $("#_media_id_create_0").append($("#_media_id_create_0
option").remove().sort(function(a, b) {
                                        var at = $(a).text().toLowerCase(), bt =
$(b).text().toLowerCase();
                                                return (at > bt)?1:-1;
                                });
                                //Cambiar guardar por  añadir

$("#l_com_mediaserver_domain_PlayListMedia_submit_btn").replaceWith(
                                                '<a href="#"
id="l_com_mediaserver_domain_PlayListMedia_submit_btn" alt="${button_add}" title="${button_add}"
class="btn btn-primary btn-sm">${button_add}</a>');

$("#l_com_mediaserver_domain_PlayListMedia_submit_btn").click(function() {
                                        location.reload();
                                 });

                        });
                        //Cambiar el titulo del cuadro de añadir

        $("#l_com_mediaserver_domain_PlayListMediaCreateForm").find("h4").html("${add_medias_to_
playlist}");

                }
                function initPlayList() {
                        var playlistURLS = ${playlistURLS};
                        if (playlistURLS.length==0){
                                hidePlayer();
                                return;
                        }
                        for (item in playlistURLS) {
                                vlc.playlist.add(playlistURLS[item]);
                        }
                        //vlc.playlist.stop();
                        comprobarOrden("${reproductor.estado}");
                }

                function hidePlayer(){
                        $("#vlc").hide();
                        $("#pPlayer").hide();
                        $("#botones").hide();
                }
        </script>
        <style type="text/css">
        #playListMediaCreateForm > table >thead >tr >th {display: none;}
        </style>

        <page:list id="pl_com_mediaserver_domain_PlayListMedia"
                items="${playlistmedias}" z="user-managed">
                <table:table data="${playlistmedias}"
        id="l_com_mediaserver_domain_PlayListMedia" path="/playlistmedias"       z="user-managed"
show="false">
                        <table:column id="c_com_mediaserver_domain_PlayListMedia_media"
property="media" z="user-managed" />
                </table:table>
        </page:list>
        <!-- ocultar el filtro -->
        <style>
                div.dataTables_filter label {   display: none;}
                #datatablesRedirect {   display: none;}
                #_title_pl_com_mediaserver_domain_PlayListMedia_id{text-align: left;}
                #_title_pl_com_mediaserver_domain_PlayListMedia_id>div.panel-
heading{display:none}
        </style>
</div>
```

13.4.13.8.4 Fichero “views.xml”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
```

```
    <definition extends="default" name="playlists/list">
        <put-attribute name="body" value="/WEB-INF/views/playlists/list.jspx"/>
    </definition>
<definition extends="default" name="playlists/show">
        <put-attribute name="body" value="/WEB-INF/views/playlists/show.jspx"/>
    </definition>
<definition extends="default" name="playlists/create">
        <put-attribute name="body" value="/WEB-INF/views/playlists/create.jspx"/>
    </definition>
<definition extends="default" name="playlists/update">
        <put-attribute name="body" value="/WEB-INF/views/playlists/update.jspx"/>
    </definition>
<definition extends="default" name="playlists/findPlayListsByPlayListName">
        <put-attribute name="body" value="/WEB-
INF/views/playlists/findPlayListsByPlayListName.jspx"/>
    </definition>
</tiles-definitions>
```

13.4.13.9 Directorio security.assignrole

13.4.13.9.1 Fichero "create.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core" xmlns:field="urn:jsptagdir:/WEB-
INF/tags/jquery/form/fields"
xmlns:form="urn:jsptagdir:/WEB-INF/tags/datatables" xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <form:create id="fc_com_mediaserver_domain_security_UserRole" modelAttribute="userRole"
path="/security/assignrole" render="${empty dependencies}" z="gpPm3qdDBgZBswL3xiJ+bGwJ/a4=">
        <field:select field="user" id="c_com_mediaserver_domain_security_UserRole_user"
itemValue="id" items="${users}" path="/security/users" required="true"
z="aPCUG0AWz8m6SkTnSSDevwkhazs="/>
        <field:select field="role" id="c_com_mediaserver_domain_security_UserRole_role"
itemValue="id" items="${roles}" path="/security/roles" required="true"
z="TxUp1oJ+Jkv3GNV0Oo27/15hvt8="/>
    </form:create>
    <form:dependency dependencies="${dependencies}"
id="d_com_mediaserver_domain_security_UserRole" render="${not empty dependencies}"
z="0ccgCpAOMTqegTsjZZHorr16Mrc="/>
</div>
```

13.4.13.9.2 Fichero "list.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-
INF/tags/datatables"
xmlns:table="urn:jsptagdir:/WEB-INF/tags/datatables" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:list id="pl_com_mediaserver_domain_security_UserRole" items="${userroles}"
z="8a1TNqbzduFSqkSLzeVjOBBXmi0=">
        <table:table data="${userroles}" id="l_com_mediaserver_domain_security_UserRole"
path="/security/assignrole" z="nEjefCmZAHhz1T6PvmKxrUfRvjQ=">
            <table:column id="c_com_mediaserver_domain_security_UserRole_user" property="user"
z="qw8/pqnR3+166atTtaB5nSggzq8="/>
            <table:column id="c_com_mediaserver_domain_security_UserRole_role" property="role"
z="eATsphUwCW7hMEGUgrupZorAU4E="/>
        </table:table>
    </page:list>
        <!-- ocultar el filtro -->
    <style>div.dataTables_filter label {
  display: none;}</style>
</div>
```

13.4.13.9.3 Fichero "show.jspx":

```
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-INF/tags/datatables"
version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:show id="ps_com_mediaserver_domain_security_UserRole" object="${userrole}"
path="/security/assignrole" z="+a/Gfc+EbxQ6AbxWGaAd/v0Pv6M=">
        <field:display field="user" id="s_com_mediaserver_domain_security_UserRole_user"
object="${userrole}" z="+LR2Bz0mZ8HEBgx+nBg/37D00gM="/>
        <field:display field="role" id="s_com_mediaserver_domain_security_UserRole_role"
object="${userrole}" z="g4udvDp4+Mm+Jaq7+8yjHsLwr7s="/>
    </page:show>
</div>
```

13.4.13.9.4 Fichero "update.jspx.xml":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:form="urn:jsptagdir:/WEB-INF/tags/datatables" xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <form:update id="fu_com_mediaserver_domain_security_UserRole" modelAttribute="userRole"
path="/security/assignrole" versionField="Version" z="c/rLU7FMYBoxNnW7Zai25p0Z2YI=">
        <field:select field="user" id="c_com_mediaserver_domain_security_UserRole_user"
itemValue="id" items="${users}" path="/security/users" required="true"
z="aPCUG0AWz8m6SkTnSSDevwkhazs="/>
        <field:select field="role" id="c_com_mediaserver_domain_security_UserRole_role"
itemValue="id" items="${roles}" path="/security/roles" required="true"
z="TxUp1oJ+Jkv3GNV0Oo27/15hvt8="/>
    </form:update>
</div>
```

13.4.13.9.5 Fichero "views.xml":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
    <definition extends="default" name="security/assignrole/list">
        <put-attribute name="body" value="/WEB-INF/views/security/assignrole/list.jspx"/>
    </definition>
<definition extends="default" name="security/assignrole/show">
        <put-attribute name="body" value="/WEB-INF/views/security/assignrole/show.jspx"/>
    </definition>
<definition extends="default" name="security/assignrole/create">
        <put-attribute name="body" value="/WEB-INF/views/security/assignrole/create.jspx"/>
    </definition>
<definition extends="default" name="security/assignrole/update">
        <put-attribute name="body" value="/WEB-INF/views/security/assignrole/update.jspx"/>
    </definition>
</tiles-definitions>
```

13.4.13.10 Directorio security.users

13.4.13.10.1 Fichero "create.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:form="urn:jsptagdir:/WEB-INF/tags/datatables"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:spring="http://www.springframework.org/tags" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
```

```
    <form:create id="fc_com_mediaserver_domain_security_User" modelAttribute="user"
path="/security/users" render="${empty dependencies}" z="gI8aimltWRAltgV6Xhhc8RGZafI=">
        <field:input field="firstName" id="c_com_mediaserver_domain_security_User_firstName"
min="1" required="true" z="iiIHU7n738046CRf6I6GUqLq4KI="/>
        <field:input field="lastName" id="c_com_mediaserver_domain_security_User_lastName"
min="1" required="true" z="wMMOnhCLXbhj/zvOWeaYx7VcdjE="/>
        <field:input field="emailAddress"
id="c_com_mediaserver_domain_security_User_emailAddress" min="1" required="true"
validationMessageCode="field_invalid_email" z="yFCKkoPyuzErREJwIF7Q6R/Sgu4="/>
        <field:input field="password" id="c_com_mediaserver_domain_security_User_password"
min="1" required="true" z="wKzlwJpG5k15v36RvCXK+QvIh4g="/>
        <field:datetime dateTimePattern="${user_activationdate_date_format}"
field="activationDate" id="c_com_mediaserver_domain_security_User_activationDate" render="false"
z="user-managed"/>
        <field:checkbox field="enabled" id="c_com_mediaserver_domain_security_User_enabled"
z="CWMV648stFq768SI4OT2F6rfA88="/>
        <field:checkbox field="locked" id="c_com_mediaserver_domain_security_User_locked"
z="KGcH58CbqXPEJU7r7yhq7jQeCzI="/>
    </form:create>
    <form:dependency dependencies="${dependencies}" id="d_com_mediaserver_domain_security_User"
render="${not empty dependencies}" z="2EJQTvsVk2mpk0gqHULWr5RLVa4="/>
</div>
```

13.4.13.10.2 Fichero “list.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-
INF/tags/datatables"
xmlns:table="urn:jsptagdir:/WEB-INF/tags/datatables" version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:list id="pl_com_mediaserver_domain_security_User" items="${users}"
z="WxlsRuQhb7RE4/cLARuG8NF5sU4=">
        <table:table data="${users}" id="l_com_mediaserver_domain_security_User"
path="/security/users" z="KQbVzv7mcewK5lj2R+m7+tbvdzs=">
            <table:column id="c_com_mediaserver_domain_security_User_emailAddress"
property="emailAddress" z="hVv8ftdIKm4s+C5uwZSGW+6N4RM="/>
            <table:column id="c_com_mediaserver_domain_security_User_password"
property="password" render="false" z="user-managed"/>
            <table:column date="true" dateTimePattern="${user_activationdate_date_format}"
id="c_com_mediaserver_domain_security_User_activationDate" property="activationDate"
z="jYIGaAvOgYgMYMFs0qrNk5d8J9A="/>
            <table:column id="c_com_mediaserver_domain_security_User_enabled" property="enabled"
z="U72iCSN1Idt/MZIVJNzJbKuqZP4="/>
            <table:column id="c_com_mediaserver_domain_security_User_locked" property="locked"
z="?"/>
            <table:column id="c_com_mediaserver_domain_security_User_firstName"
property="firstName" z="v5mOeZYSTDhEuNqj1a7ucBHxS7k="/>
            <table:column id="c_com_mediaserver_domain_security_User_lastName"
property="lastName" z="O+H2x4BOFhIcbaOzFvGvJmynXYw="/>
        </table:table>
    </page:list>

</div>
```

13.4.13.10.3 Fichero “show.jspx”:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:page="urn:jsptagdir:/WEB-INF/tags/datatables"
version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <page:show id="ps_com_mediaserver_domain_security_User" object="${user}"
path="/security/users" z="fA7ailTW1alRg7e0+xYbZ65QTIo=">
        <field:display field="firstName" id="s_com_mediaserver_domain_security_User_firstName"
object="${user}" z="DB8fU3U9oloMUiwIMfQpSlAbtnc="/>
        <field:display field="lastName" id="s_com_mediaserver_domain_security_User_lastName"
object="${user}" z="sSWv9UIpEwGGc/4gyKUJKpdBb7Q="/>
```

```
        <field:display field="emailAddress"
id="s_com_mediaserver_domain_security_User_emailAddress" object="${user}"
z="cPQuDmZLTO2X+m7cWUJnaxjuALI="/>
        <field:display field="password" id="s_com_mediaserver_domain_security_User_password"
object="${user}" render="false" z="user-managed"/>
        <field:display date="true" dateTimePattern="${user_activationdate_date_format}"
field="activationDate" id="s_com_mediaserver_domain_security_User_activationDate"
object="${user}" z="whQAq90GlYhB1y1QKPLBMbi/ISY="/>
        <field:display field="enabled" id="s_com_mediaserver_domain_security_User_enabled"
object="${user}" z="RXoS85drP0waWbSYfNmyKSJtPHo="/>
        <field:display field="locked" id="s_com_mediaserver_domain_security_User_locked"
object="${user}" z="Fac2/+xuX3MQWaNrxzZSa+xDJ/I="/>
    </page:show>
</div>
```

13.4.13.10.4 Fichero "update.jspx.xml":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:field="urn:jsptagdir:/WEB-INF/tags/jquery/form/fields"
xmlns:form="urn:jsptagdir:/WEB-INF/tags/datatables" xmlns:jsp="http://java.sun.com/JSP/Page"
version="2.0">
    <jsp:directive.page contentType="text/html;charset=UTF-8"/>
    <jsp:output omit-xml-declaration="yes"/>
    <form:update id="fu_com_mediaserver_domain_security_User" modelAttribute="user"
path="/security/users" versionField="Version" z="VtfwKG+DYAKD1Qpw1/vdcRCLhDg=">
        <field:input field="firstName" id="c_com_mediaserver_domain_security_User_firstName"
min="1" required="true" z="iiIHU7n738046CRf6I6GUqLq4KI="/>
        <field:input field="lastName" id="c_com_mediaserver_domain_security_User_lastName"
min="1" required="true" z="wMMOnhCLXbhj/zvOWeaYx7VcdjE="/>
        <field:input field="emailAddress"
id="c_com_mediaserver_domain_security_User_emailAddress" min="1" required="true"
validationMessageCode="field_invalid_email" z="yFCKkoPyuzErREJwIF7Q6R/Sgu4="/>
        <field:input field="password" id="c_com_mediaserver_domain_security_User_password"
min="1" render="false" required="true" z="user-managed"/>
        <field:datetime dateTimePattern="${user_activationdate_date_format}"
field="activationDate" id="c_com_mediaserver_domain_security_User_activationDate" render="false"
z="user-managed"/>
        <field:checkbox field="enabled" id="c_com_mediaserver_domain_security_User_enabled"
z="CWMV648stFq768SI4OT2F6rfA88="/>
        <field:checkbox field="locked" id="c_com_mediaserver_domain_security_User_locked"
z="KGcH58CbqXPEJU7r7yhq7jQeCzI="/>
    </form:update>
</div>
```

13.4.13.10.5 Fichero "views.xml":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
    <definition extends="default" name="security/users/list">
        <put-attribute name="body" value="/WEB-INF/views/security/users/list.jspx"/>
    </definition>
<definition extends="default" name="security/users/show">
        <put-attribute name="body" value="/WEB-INF/views/security/users/show.jspx"/>
    </definition>
<definition extends="default" name="security/users/create">
        <put-attribute name="body" value="/WEB-INF/views/security/users/create.jspx"/>
    </definition>
<definition extends="default" name="security/users/update">
        <put-attribute name="body" value="/WEB-INF/views/security/users/update.jspx"/>
    </definition>
</tiles-definitions>
```

13.4.13.11 Directorio signup

13.4.13.11.1 Fichero "error.jspx":

```
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_sign_up" htmlEscape="false" var="title" />
        <util:panel id="title" title="${title}">
                <h2>
                        <spring:message code="header_signup_failed" />
                </h2>
                <p>
                        <spring:message code="desc_signup_failed" />
                </p>
        </util:panel>

        <script type="text/javascript">
                $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                });
        </script>
</div>
```

13.4.13.11.2 Fichero "form.jspx":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:field="urn:jsptagdir:/WEB-INF/tags/form/fields"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:fn="http://java.sun.com/jsp/jstl/functions"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_sign_up" htmlEscape="false" var="title" />
        <util:panel id="title" title="${title}">
                <form:form action="" method="POST" commandName="form">
                        <field:input field="firstName"
                                id="c_com_mediaserver_domain_security_User_firstName" min="1"
                                required="true" z="?" />
                        <field:input field="lastName"
                                id="c_com_mediaserver_domain_security_User_lastName" min="1"
                                required="true" z="?" />
                        <field:input field="emailAddress"
                                id="c_com_mediaserver_domain_security_User_emailAddress"
                                min="1" required="true"
validationMessageCode="field_invalid_email"
                                z="?" />
                        <field:input field="password"
                                id="c_com_mediaserver_domain_security_User_password" min="5"
                                required="true" type="password" z="?" />
                        <field:input field="retypePassword" id="c_retype_password"
                                type="password" z="?" />

        ${captcha_form}

        <div class="submit">
                                <script type="text/javascript">
                                        Spring.addDecoration(new Spring.ValidateAllDecoration({
                                                elementId : 'proceed',
                                                event : 'onclick'
```

```
                                        }));
                                </script>
                                <spring:message code="button_submit" htmlEscape="false"
                                        var="submit_label" />
                                <input id="proceed" type="submit"
                                        value="${fn:escapeXml(submit_label)}" />
                        </div>
                </form:form>

        </util:panel>
        <script type="text/javascript">
                $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                $("#form").attr("class","form-horizontal");
                $("#form > div").attr("class","control-group form-group");
                $("#form > div > label").attr("class","control-label col-xs-5 col-sm-4 col-md-3
col-lg-3");
                $(".dijitInputInner").attr("class","form-control input-sm");
                $("#proceed").attr("class","btn btn-primary btn-block");
                //$(".errors").attr("class","alert-danger");
        });
        </script>
        <style>
                 #form, #form >div > div { margin: auto;  width: 60%;  min-width: 400px;}
                 #recaptcha_table{margin:auto!important;}
                 .alert-danger { border-radius: 4px; }
        </style>
</div>
```

13.4.13.11.3 Fichero “thanks.jspx”:

```
<div xmlns:c="http://java.sun.com/jsp/jstl/core"
        xmlns:form="http://www.springframework.org/tags/form"
        xmlns:jsp="http://java.sun.com/JSP/Page"
        xmlns:spring="http://www.springframework.org/tags"
        xmlns:util="urn:jsptagdir:/WEB-INF/tags/util" version="2.0">
        <jsp:output omit-xml-declaration="yes" />

        <spring:message code="label_sign_up" htmlEscape="false" var="title" />
        <util:panel id="title" title="${title}">
                <h2>
                        <spring:message code="header_signup_successful" />
                </h2>
                <p>
                        <c:choose>
                                <c:when test="${accountAutoEnabled}">
                                        <spring:message code="desc_signup_success_emailed" />
                                </c:when>
                                <c:otherwise>
                                        <spring:message code="desc_signup_success_wait" />
                                </c:otherwise>
                        </c:choose>
                </p>
        </util:panel>
        <script type="text/javascript">
                $(window).load(function() {
                $("#_title_title_id").attr("class","panel panel-default");
                $(".dijitTitlePaneTextNode").attr("class","panel-title");
                $(".dijitTitlePaneTitle").attr("class","panel-heading");
                $(".dijitArrowNode").hide();
                $(".dijitTitlePaneContentOuter").attr("class","panel-body");
                });
        </script>
</div>
```

13.4.13.11.4 Fichero "views.xml":

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE tiles-definitions PUBLIC "-//Apache Software Foundation//DTD Tiles Configuration
2.1//EN" "http://tiles.apache.org/dtds/tiles-config_2_1.dtd">
<tiles-definitions>
        <definition extends="public" name="signup/form">
                <put-attribute name="body" value="/WEB-INF/views/signup/form.jspx" />
        </definition>
        <definition extends="public" name="signup/thanks">
                <put-attribute name="body" value="/WEB-INF/views/signup/thanks.jspx" />
        </definition>
        <definition extends="public" name="signup/error">
                <put-attribute name="body" value="/WEB-INF/views/signup/error.jspx" />
        </definition>
</tiles-definitions>
```

13.4.14 Fichero de dependencias de Maven

13.4.14.1 Fichero pom.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <groupId>com.mediaserver</groupId>
        <artifactId>mediaserver</artifactId>
        <packaging>war</packaging>
        <version>0.1.0.BUILD-SNAPSHOT</version>
        <name>mediaserver</name>
        <properties>
                <aspectj.version>1.7.2</aspectj.version>
                <java.version>7</java.version>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <roo.version>1.3.0.RELEASE</roo.version>
                <slf4j.version>1.7.5</slf4j.version>
                <querydsl.version>3.1.1</querydsl.version>
                <jasperreports.version>5.0.1</jasperreports.version>
                <jasperreportsfonts.version>4.0.0</jasperreportsfonts.version>
                <apachepoi.version>3.9</apachepoi.version>
                <spring-security.version>3.2.0.RELEASE</spring-security.version>

                <org.springframework-version>4.0.0.RELEASE</org.springframework-version>

                <!-- Web -->
                <jsp.version>2.2</jsp.version>
                <jstl.version>1.2</jstl.version>
                <servlet.version>3.1.0</servlet.version>

        </properties>
        <repositories>
                <repository>
                        <id>spring-maven-release</id>
                        <name>Spring Maven Release Repository</name>
                        <url>http://maven.springframework.org/release</url>
                </repository>
                <repository>
                        <id>spring-maven-milestone</id>
                        <name>Spring Maven Milestone Repository</name>
                        <url>http://maven.springframework.org/milestone</url>
                </repository>
                <repository>
                        <id>spring-roo-repository</id>
```

```
                <name>Spring Roo Repository</name>
                <url>http://spring-roo-repository.springsource.org/release</url>
        </repository>
        <repository>
                <id>gvNIX Add-on repository</id>
                <url>https://gvnix.googlecode.com/svn/repo</url>
                <name>gvNIX Add-on repository</name>
        </repository>
        <repository>
                <id>spring-snapshots</id>
                <url>http://repo.spring.io/snapshot</url>
                <snapshots>
                        <enabled>true</enabled>
                </snapshots>
                <releases>
                        <enabled>false</enabled>
                </releases>
        </repository>
        <repository>
                <id>spring-milestones</id>
                <url>http://repo.springsource.org/libs-milestone</url>
                <snapshots>
                        <enabled>false</enabled>
                </snapshots>
                <releases>
                        <enabled>true</enabled>
                </releases>
        </repository>
        <repository>
                <id>java-net</id>
                <url>https://maven.java.net/content/repositories/releases</url>
        </repository>
</repositories>
<pluginRepositories>
        <pluginRepository>
                <id>spring-maven-release</id>
                <name>Spring Maven Release Repository</name>
                <url>http://maven.springframework.org/release</url>
        </pluginRepository>
        <pluginRepository>
                <id>spring-maven-milestone</id>
                <name>Spring Maven Milestone Repository</name>
                <url>http://maven.springframework.org/milestone</url>
        </pluginRepository>
        <pluginRepository>
                <id>spring-roo-repository</id>
                <name>Spring Roo Repository</name>
                <url>http://spring-roo-repository.springsource.org/release</url>
        </pluginRepository>
</pluginRepositories>
<dependencies>

        <!-- Spring core -->
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>${org.springframework-version}</version>
                <exclusions>
                        <!-- Exclude Commons Logging in favor of SLF4j -->
                        <exclusion>
                                <groupId>commons-logging</groupId>
                                <artifactId>commons-logging</artifactId>
                        </exclusion>
                </exclusions>
        </dependency>
        <!-- Spring messaging -->
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-messaging</artifactId>
                <version>${org.springframework-version}</version>
        </dependency>
        <!-- Spring WEB MVC -->
        <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-webmvc</artifactId>
```

```
                        <version>${org.springframework-version}</version>
                </dependency>
                <!-- Spring WEBSOCKET -->
                <dependency>
                        <groupId>org.springframework</groupId>
                        <artifactId>spring-websocket</artifactId>
                        <version>${org.springframework-version}</version>
                </dependency>
                <!-- JSON: Core Jackson abstractions, basic JSON streaming API implementation --
>
                <dependency>
                        <groupId>com.fasterxml.jackson.core</groupId>
                        <artifactId>jackson-core</artifactId>
                        <version>2.3.0</version>
                </dependency>
                <!-- For SockJS -->
                <!-- http://jira.codehaus.org/browse/JACKSON-884 -->
                <!-- JSON: General data-binding functionality for Jackson: works on core
                        streaming API -->
                <dependency>
                        <groupId>com.fasterxml.jackson.core</groupId>
                        <artifactId>jackson-databind</artifactId>
                        <version>2.3.0</version>
                </dependency>

                <!-- Required when the "stomp-broker-relay" profile is enabled -->
                <dependency>
                        <groupId>org.projectreactor</groupId>
                        <artifactId>reactor-tcp</artifactId>
                        <version>1.0.0.RELEASE</version>
                </dependency>

                <dependency>
                        <groupId>javax.servlet</groupId>
                        <artifactId>javax.servlet-api</artifactId>
                        <version>${servlet.version}</version>
                        <scope>provided</scope>
                </dependency>

                <!-- JSTL tag library -->
<!--            <dependency> -->
<!--                    <groupId>javax.servlet</groupId> -->
<!--                    <artifactId>jstl</artifactId> -->
<!--                    <version>${jstl.version}</version> -->
<!--            </dependency> -->

                <!-- JSP Pages -->
                <dependency>
                        <groupId>javax.servlet.jsp</groupId>
                        <artifactId>jsp-api</artifactId>
                        <version>${jsp.version}</version>
                        <scope>provided</scope>
                </dependency>

                <!-- old -->
                <!-- General dependencies for standard applications -->
                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.11</version>
                        <scope>test</scope>
                </dependency>
                <dependency>
                        <groupId>log4j</groupId>
                        <artifactId>log4j</artifactId>
                        <version>1.2.17</version>
                </dependency>
                <dependency>
                        <groupId>org.slf4j</groupId>
                        <artifactId>slf4j-api</artifactId>
                        <version>${slf4j.version}</version>
                </dependency>
                <dependency>
                        <groupId>org.slf4j</groupId>
                        <artifactId>jcl-over-slf4j</artifactId>
```

```
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>${aspectj.version}</version>
        </dependency>
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>${aspectj.version}</version>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>servlet-api</artifactId>
            <version>2.5</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>net.sf.flexjson</groupId>
            <artifactId>flexjson</artifactId>
            <version>2.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
            <version>3.1</version>
        </dependency>
        <!-- ROO dependencies -->
        <dependency>
            <groupId>org.springframework.roo</groupId>
            <artifactId>org.springframework.roo.annotations</artifactId>
            <version>${roo.version}</version>
            <scope>provided</scope>
        </dependency>
        <!-- Spring dependencies -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${org.springframework-version}</version>
            <exclusions>
                <exclusion>
                    <groupId>commons-logging</groupId>
                    <artifactId>commons-logging</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${org.springframework-version}</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${org.springframework-version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
            <version>${org.springframework-version}</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>5.1.18</version>
        </dependency>
        <dependency>
```

```
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-core</artifactId>
		<version>4.2.2.Final</version>
	</dependency>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-entitymanager</artifactId>
		<version>4.2.2.Final</version>
		<exclusions>
			<exclusion>
				<groupId>cglib</groupId>
				<artifactId>cglib</artifactId>
			</exclusion>
			<exclusion>
				<groupId>dom4j</groupId>
				<artifactId>dom4j</artifactId>
			</exclusion>
		</exclusions>
	</dependency>
	<dependency>
		<groupId>org.hibernate.javax.persistence</groupId>
		<artifactId>hibernate-jpa-2.0-api</artifactId>
		<version>1.0.1.Final</version>
	</dependency>
	<dependency>
		<groupId>commons-collections</groupId>
		<artifactId>commons-collections</artifactId>
		<version>3.2.1</version>
	</dependency>
	<dependency>
		<groupId>org.hibernate</groupId>
		<artifactId>hibernate-validator</artifactId>
		<version>4.3.1.Final</version>
	</dependency>
	<dependency>
		<groupId>javax.validation</groupId>
		<artifactId>validation-api</artifactId>
		<version>1.0.0.GA</version>
	</dependency>
	<dependency>
		<groupId>javax.transaction</groupId>
		<artifactId>jta</artifactId>
		<version>1.1</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-jdbc</artifactId>
		<version>${org.springframework-version}</version>
	</dependency>
	<dependency>
		<groupId>org.springframework</groupId>
		<artifactId>spring-orm</artifactId>
		<version>${org.springframework-version}</version>
	</dependency>
	<dependency>
		<groupId>commons-pool</groupId>
		<artifactId>commons-pool</artifactId>
		<version>1.5.6</version>
	</dependency>
	<dependency>
		<groupId>commons-dbcp</groupId>
		<artifactId>commons-dbcp</artifactId>
		<version>1.4</version>
		<exclusions>
			<exclusion>
				<groupId>commons-logging</groupId>
				<artifactId>commons-logging</artifactId>
			</exclusion>
			<exclusion>
				<groupId>xml-apis</groupId>
				<artifactId>xml-apis</artifactId>
			</exclusion>
		</exclusions>
	</dependency>
```

```
		<dependency>
			<groupId>org.springframework.webflow</groupId>
			<artifactId>spring-js-resources</artifactId>
			<version>2.2.1.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>commons-digester</groupId>
			<artifactId>commons-digester</artifactId>
			<version>2.1</version>
			<exclusions>
				<exclusion>
					<groupId>commons-logging</groupId>
					<artifactId>commons-logging</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
		<dependency>
			<groupId>commons-fileupload</groupId>
			<artifactId>commons-fileupload</artifactId>
			<version>1.2.2</version>
		</dependency>
<!--		<dependency> -->
<!--			<groupId>javax.servlet.jsp.jstl</groupId> -->
<!--			<artifactId>jstl-api</artifactId> -->
<!--			<version>1.2</version> -->
<!--		</dependency> -->
		<dependency>
			<groupId>org.glassfish.web</groupId>
			<artifactId>jstl-impl</artifactId>
			<version>1.2</version>
		</dependency>
		<dependency>
			<groupId>javax.el</groupId>
			<artifactId>el-api</artifactId>
			<version>2.2</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>joda-time</groupId>
			<artifactId>joda-time</artifactId>
			<version>1.6</version>
		</dependency>

		<dependency>
			<groupId>commons-codec</groupId>
			<artifactId>commons-codec</artifactId>
			<version>1.5</version>
		</dependency>
		<dependency>
			<groupId>org.apache.tiles</groupId>
			<artifactId>tiles-jsp</artifactId>
			<version>2.2.2</version>
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.addon.web.mvc</artifactId>
			<version>1.3.1-RELEASE</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.addon.jpa</artifactId>
			<version>1.3.1-RELEASE</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>com.mysema.querydsl</groupId>
			<artifactId>querydsl-apt</artifactId>
			<version>${querydsl.version}</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>com.mysema.querydsl</groupId>
			<artifactId>querydsl-jpa</artifactId>
			<version>${querydsl.version}</version>
```

```
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.web.json.binding</artifactId>
			<version>1.3.1-RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.addon.datatables</artifactId>
			<version>1.3.1-RELEASE</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.datatables.tags</artifactId>
			<version>1.3.1-RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.web.datatables</artifactId>
			<version>1.3.1-RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.gvnix</groupId>
			<artifactId>org.gvnix.web.report.roo.addon</artifactId>
			<version>1.4.0.RELEASE</version>
			<scope>provided</scope>
		</dependency>
		<dependency>
			<groupId>net.sf.jasperreports</groupId>
			<artifactId>jasperreports</artifactId>
			<version>${jasperreports.version}</version>
			<exclusions>
				<exclusion>
					<groupId>commons-collections</groupId>
					<artifactId>commons-collections</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
		<dependency>
			<groupId>org.apache.poi</groupId>
			<artifactId>poi</artifactId>
			<version>${apachepoi.version}</version>
		</dependency>
		<dependency>
			<groupId>net.sf.jasperreports</groupId>
			<artifactId>jasperreports-fonts</artifactId>
			<version>${jasperreportsfonts.version}</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-core</artifactId>
			<version>${spring-security.version}</version>
			<exclusions>
				<exclusion>
					<groupId>commons-logging</groupId>
					<artifactId>commons-logging</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-config</artifactId>
			<version>${spring-security.version}</version>
			<exclusions>
				<exclusion>
					<groupId>commons-logging</groupId>
					<artifactId>commons-logging</artifactId>
				</exclusion>
			</exclusions>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-web</artifactId>
			<version>${spring-security.version}</version>
```

```
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-taglibs</artifactId>
			<version>${spring-security.version}</version>
		</dependency>

		<dependency>
	    <groupId>org.springframework</groupId>
	    <artifactId>spring-context-support</artifactId>
	    <version>${org.springframework-version}</version>
	</dependency>
		 <dependency>
	    <groupId>javax.mail</groupId>
	    <artifactId>mail</artifactId>
	    <version>1.4.7</version>
	</dependency>
	<dependency>
			<groupId>javax.mail</groupId>
			<artifactId>javax.mail-api</artifactId>
			<version>1.5.2</version>
		</dependency>
	<dependency>
	    <groupId>javax.activation</groupId>
	    <artifactId>activation</artifactId>
	    <version>1.1.1</version>
	</dependency>
			<!-- dependencia para subida de documentos -->
	<dependency>
		<groupId>commons-io</groupId>
		<artifactId>commons-io</artifactId>
		<version>2.4</version>
		</dependency>
	<dependency>
	    <groupId>net.tanesha.recaptcha4j</groupId>
	    <artifactId>recaptcha4j</artifactId>
	    <version>0.0.7</version>
	</dependency>
	<dependency>
	    <groupId>org.springframework.batch</groupId>
	    <artifactId>spring-batch-core</artifactId>
	    <version>2.1.1.RELEASE</version>
	</dependency>
	<dependency>
	    <groupId>org.springframework.batch</groupId>
	    <artifactId>spring-batch-admin-manager</artifactId>
	    <version>1.0.0.RELEASE</version>
	</dependency>
	<dependency>
	<groupId>com.google.code.gson</groupId>
	<artifactId>gson</artifactId>
	<version>2.2.4</version>
	</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-test</artifactId>
			<version>1.1.4.RELEASE</version>
		</dependency>

		<dependency>
			<groupId>org.apache.activemq</groupId>
			<artifactId>activemq-client</artifactId>
			<version>5.9.1</version>
		</dependency>
		<dependency>
			<groupId>org.scala-lang</groupId>
			<artifactId>scala-library</artifactId>
			<version>2.10.4</version>
		</dependency>
	</dependencies>
	<build>
		<plugins>
			<plugin>
				<groupId>org.apache.maven.plugins</groupId>
```

```
                                <artifactId>maven-war-plugin</artifactId>
                                <version>2.4</version>
                                <configuration>
                                        <failOnMissingWebXml>false</failOnMissingWebXml>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-compiler-plugin</artifactId>
                                <version>2.5.1</version>
                                <configuration>
                                        <source>${java.version}</source>
                                        <target>${java.version}</target>
                                        <encoding>${project.build.sourceEncoding}</encoding>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.codehaus.mojo</groupId>
                                <artifactId>aspectj-maven-plugin</artifactId>
                                <version>1.4</version>
                                <!-- NB: do not use 1.3 or 1.3.x due to MASPECTJ-90 issue -->
                                <dependencies>
                                        <!-- NB: You must use Maven 2.0.9 or above or these are
ignored (see
                                                MNG-2972) -->
                                        <dependency>
                                                <groupId>org.aspectj</groupId>
                                                <artifactId>aspectjrt</artifactId>
                                                <version>${aspectj.version}</version>
                                        </dependency>
                                        <dependency>
                                                <groupId>org.aspectj</groupId>
                                                <artifactId>aspectjtools</artifactId>
                                                <version>${aspectj.version}</version>
                                        </dependency>
                                </dependencies>
                                <executions>
                                        <execution>
                                                <phase>process-sources</phase>
                                                <goals>
                                                        <goal>compile</goal>
                                                        <goal>test-compile</goal>
                                                </goals>
                                        </execution>
                                </executions>
                                <configuration>
                                        <outxml>true</outxml>
                                        <aspectLibraries>
                                                <aspectLibrary>
                                                        <groupId>org.springframework</groupId>
                                                        <artifactId>spring-aspects</artifactId>
                                                </aspectLibrary>
                                        </aspectLibraries>
                                        <source>${java.version}</source>
                                        <target>${java.version}</target>
                                        <!-- Aspects in src/main/java and src/main/aspects are
added as default
                                                in the compile goal. Aspects in src/test/java
and src/test/aspects are added
                                                as default in the test-compile goal. Aspects in
src/main/java are added in
                                                the test-compile goal if
weaveWithAspectsInMainSourceFolder is set to true -->

                <weaveWithAspectsInMainSourceFolder>false</weaveWithAspectsInMainSourceFolder>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-resources-plugin</artifactId>
                                <version>2.6</version>
                                <configuration>
                                        <encoding>${project.build.sourceEncoding}</encoding>
                                </configuration>
                        </plugin>
```

```
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-surefire-plugin</artifactId>
                                <version>2.12</version>
                                <configuration>
                                        <printSummary>false</printSummary>

        <redirectTestOutputToFile>true</redirectTestOutputToFile>
                                        <excludes>
                                                <exclude>**/*_Roo_*</exclude>
                                        </excludes>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-assembly-plugin</artifactId>
                                <version>2.3</version>
                                <configuration>
                                        <descriptorRefs>
                                                <descriptorRef>jar-with-
dependencies</descriptorRef>
                                        </descriptorRefs>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-deploy-plugin</artifactId>
                                <version>2.7</version>
                        </plugin>
                        <!-- IDE -->
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-eclipse-plugin</artifactId>
                                <version>2.9</version>
                                <!-- Note 2.8 does not work with AspectJ aspect path -->
                                <configuration>
                                        <downloadSources>true</downloadSources>
                                        <downloadJavadocs>false</downloadJavadocs>
                                        <wtpversion>2.0</wtpversion>
                                        <additionalBuildcommands>
                                                <buildCommand>

        <name>org.eclipse.ajdt.core.ajbuilder</name>
                                                        <arguments>

        <aspectPath>org.springframework.aspects</aspectPath>
                                                        </arguments>
                                                </buildCommand>
                                                <buildCommand>

        <name>org.springframework.ide.eclipse.core.springbuilder</name>
                                                </buildCommand>
                                        </additionalBuildcommands>
                                        <additionalProjectnatures>

        <projectnature>org.eclipse.ajdt.ui.ajnature</projectnature>

        <projectnature>com.springsource.sts.roo.core.nature</projectnature>

        <projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
                                        </additionalProjectnatures>
                                </configuration>
                        </plugin>
                        <plugin>
                                <groupId>org.apache.maven.plugins</groupId>
                                <artifactId>maven-idea-plugin</artifactId>
                                <version>2.2</version>
                                <configuration>
                                        <downloadSources>true</downloadSources>
                                        <dependenciesAsLibraries>true</dependenciesAsLibraries>
                                </configuration>
                        </plugin>
                        <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
```

```
                <version>2.2</version>
            </plugin>
                        <plugin>
                                <groupId>org.codehaus.mojo</groupId>
                                <artifactId>tomcat-maven-plugin</artifactId>
                                <version>1.1</version>
                        </plugin>
                        <plugin>
                                <groupId>org.mortbay.jetty</groupId>
                                <artifactId>jetty-maven-plugin</artifactId>
                                <version>9.0.4.v20130625</version>
                                <configuration>
                                        <webAppConfig>
                                                <contextPath>/${project.name}</contextPath>
                                        </webAppConfig>
                                </configuration>
                        </plugin>
                </plugins>
        </build>
</project>
```

13.4.15 Estilos principales

13.4.15.1 Fichero "standard.css"

```
/*css reset*/
/* Eric Meyer's Reset CSS v2.0 - http://cssreset.com */
html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,a,abbr,acronym,addres
s,big,cite,code,del,dfn,em,img,ins,kbd,q,s,samp,small,strike,strong,sub,sup,tt,var,b,u,i,center,
dl,dt,dd,ol,ul,li,fieldset,form,label,legend,table,caption,tbody,tfoot,thead,tr,th,td,article,as
ide,canvas,details,embed,figure,figcaption,footer,header,hgroup,menu,nav,output,ruby,section,sum
mary,time,mark,audio,video{margin-bottom: 0px;}
/*fin css reset*/

.gsc-search-button .gsc-search-button-v2{
  background: url(http://www.google.com/uds/css/v2/search_box_icon.png) no-repeat center;
  color: #fff;  background-color: #428bca;  border-color: #357ebd;   height: 24px ;
}
.gsc-search-button:hover .gsc-search-button-v2:hover{
  background: url(http://www.google.com/uds/css/v2/search_box_icon.png) no-repeat center;
  background-color: #357ebd;  border-color: #357ebd;
}

#gsc-i-id1{line-height:normal;}
 #gs_tti50{ border = 0; }
.bplay {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
150px 0px;border-radius: 12px;}
.bpause {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
100px 0px;border-radius: 12px;}
.bstop {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
50px 0px;border-radius: 12px;}
.bnext {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
0px 50px;border-radius: 12px;}
.bprev {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
150px 50px;border-radius: 12px;}

#vlc{border: 2px solid #a1a1a1;
    padding: 15px;
    margin-top:10px;
    border-radius: 20px;
    background: rgb(114,140,89);
    background: linear-gradient(to bottom,  rgba(114,140,89,1) 0%,rgba(66,114,14,1)
50%,rgba(125,155,99,1) 100%); /* W3C */
    }

.dataTables_scrollBody{height:100%}

.dandelion_dataTables_export{display: none}
#logoBig{
        height: 13em;
```

```
        float: left;
        margin-right: 50px;
}

#ul_derecho{margin: -9px 0px;}
#Language {padding: 3px 20px; font-weight: bold;}

#textoEstado{margin:5px;}

.quicklinks{display: none;}


#datatablesRedirect{display:none}

a{color: #118802;}

.jumbotron h1 {
  text-align: center;
  background-color: rgba(122, 197, 28, 0.63);
  border-radius: 50px 2px;
}
#reproductorPrincipal{ text-align: center;
        margin-top:10%;
        margin-bottom:10%;
        tansform:scale(2.5);
        -moz-transform:scale(2.5);
        -webkit-transform:scale(2.5);
        -o-transform:scale(2.5);}
.navbar-inverse{
        background: rgb(114,140,89); /* Old browsers */
        background: -moz-linear-gradient(top,  rgba(114,140,89,1) 0%, rgba(66,114,14,1) 50%,
rgba(125,155,99,1) 100%); /* FF3.6+ */
        background: -webkit-gradient(linear, left top, left bottom, color-
stop(0%,rgba(114,140,89,1)), color-stop(50%,rgba(66,114,14,1)), color-
stop(100%,rgba(125,155,99,1))); /* Chrome,Safari4+ */
        background: -webkit-linear-gradient(top,  rgba(114,140,89,1) 0%,rgba(66,114,14,1)
50%,rgba(125,155,99,1) 100%); /* Chrome10+,Safari5.1+ */
        background: -o-linear-gradient(top,  rgba(114,140,89,1) 0%,rgba(66,114,14,1)
50%,rgba(125,155,99,1) 100%); /* Opera 11.10+ */
        background: -ms-linear-gradient(top,  rgba(114,140,89,1) 0%,rgba(66,114,14,1)
50%,rgba(125,155,99,1) 100%); /* IE10+ */
        background: linear-gradient(to bottom,  rgba(114,140,89,1) 0%,rgba(66,114,14,1)
50%,rgba(125,155,99,1) 100%); /* W3C */
        filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#728c59',
endColorstr='#7d9b63',GradientType=0 ); /* IE6-9 */
        border-color: #527711;
}

input[type="submit"].btn-block, input[type="reset"].btn-block, input[type="button"].btn-block {
  width: 100px;  margin: auto;
}

form#user, form#userRole, form#playListMedia {margin:auto;width:50%; min-width:400px}
#dataTables_scrollBody{hight:auto !important;}

.gvnix_dataTables_toolbar>a.submit_entity{
/*      background: #b1FF41; */
        height: 2em;
        width: 2em;
        background: url("../../resources/images/datatables/accept.png") no-repeat scroll center
center transparent;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        -o-background-size: cover;
        background-size: cover;
}
.gvnix_dataTables_toolbar>a.cancel_entity{
/*      background: #FFB145; */
        height: 2em ;
        width: 2em ;
        background: url("../../resources/images/datatables/cancel.png") no-repeat scroll center
center transparent;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        -o-background-size: cover;
```

```
        background-size: cover;
}
.gvnix_dataTables_toolbar>a.update_entity,.gvnix_dataTables_toolbar>a.delete_entity,.gvnix_dataT
ables_toolbar>a.create_entity{
        height: 2em;
        width: 2em;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        -o-background-size: cover;
        background-size: cover;
}

.topnav_icon{
        width: 30px;

}

#player{
        text-align:center;
}
#l_com_mediaserver_domain_PlayListMedia{
        text-align:left;
}

a.select_all {
        background: url("../../images/datatables/book_add.png") no-repeat scroll center center
transparent;
}
a.select_none {
        background: url("../../images/datatables/book_delete.png") no-repeat scroll center
center transparent;
}

a.select_toggle {
        background: url("../../images/datatables/tick.png") no-repeat scroll center center
transparent;
}



a.update_entity {
        background: url("../../images/update.png") no-repeat scroll center center transparent;
}
a.show_entity {
        background: url("../../images/show.png") no-repeat scroll center center transparent;
}
a.create_entity {
        background: url("../../images/create.png") no-repeat scroll center center transparent;
}
a.delete_entity {
        background: url("../../images/delete.png") no-repeat scroll center center transparent;
}
a.submit_entity {
        background: url("../../images/datatables/accept.png") no-repeat scroll center center
transparent;
}
a.cancel_entity {
        background: url("../../images/datatables/cancel.png") no-repeat scroll center center
transparent;
}
/* ****** */

/*** Default  gvNIX ***/

.flag {
        height: 11px;
        width: 16px;
}

body {
  background:url(../../resources/images/pattern_1.png);
  position: relative; /* For scrollyspy */
  padding-top: 70px; /* Account for fixed navbar */
  padding-bottom: 30px;
}
```

```
        @media (min-width: 768px) and (max-width: 1020px){
                body {
                  padding-top: 120px;
                }
        }

/*
 * Main navigation
 *
 * Turn the `.navbar` at the top
 */

.navbar {
  text-shadow: 0 -1px 0 rgba(0,0,0,.15);
  background-color: #563d7c;
  border-color: #463265;
  box-shadow: 0 1px 0 rgba(255,255,255,.1);
}
.navbar .navbar-collapse {
  border-color: #463265;
}
.navbar .navbar-brand {
  color: #fff;
}
.navbar .navbar-nav > li > a {
  color: #cdbfe3;
}
.navbar .navbar-nav > li > a:hover {
        -moz-transition: color 0.3s linear;
        -o-transition: color 0.3s linear;
        -webkit-transition: color 0.3s linear;
        transition: color 0.3s linear;
        background-color: rgba(92, 119, 27, 0.68);
        background: linear-gradient(to bottom, rgba(66,114,14,1) 0%,rgba(114,140,89,1) 50%,
rgba(66,114,14,1) 100%);
        text-decoration: none;
  color: #dfd;
}
.navbar .navbar-nav > .active > a,
.navbar .navbar-nav > .active > a:hover {
  color: #fff;
  background-color: #463265;
}
.navbar .navbar-toggle {
  border-color: #563d7c;
}
.navbar .navbar-toggle:hover {
  background-color: #463265;
  border-color: #463265;
}

/* Custom theme Bootstrap for gvNIX by DISID */

.navbar {
        background-color: #608224;
        border-color: #527711;
}
.navbar .navbar-toggle {
        border-color: #608224;
}
.navbar .navbar-toggle:hover {
        background-color: #608224;
        border-color: #43600E;
}
.navbar-inverse .navbar-nav>.open>a, .navbar-inverse .navbar-nav>.open>a:hover, .navbar-inverse
.navbar-nav>.open>a:focus,
.navbar .navbar-nav > .active > a, .navbar .navbar-nav > .active > a:hover {
        color: #fff;
        background-color: #43600E;
}
.navbar .navbar-nav > li > a {
color: #fff;
 font-size: 1.2em;
}
```

```
@media (max-width: 767px){
        .navbar-inverse .navbar-nav .open .dropdown-menu>li>a {
                color: #acc47f;
                white-space: normal;
        }
        .navbar-nav .open .dropdown-menu{
         max-width: 300px;
        }
        .navbar-nav .open .dropdown-menu li{
   max-width: 300px;
  }
  .col-xs-3 button[data-toggle="offcanvas"]{
    margin: 15px 0 0;
  }
}

@media (min-width: 768px){
        .navbar-nav {
        float: none;
        margin: 0 100px;

        }
}

/* Custom sidebar menu */

#sidebar-wrapper li a{
        text-decoration: none;
        }
.affix#sidebar-wrapper{
        z-index: 100;
        }


/*
 * Footer
 *
 * Separated section of content at the bottom of all pages
 */

.bs-footer {
  padding-top: 40px;
  padding-bottom: 30px;
  margin-top: 10px;
  color: #777;
  text-align: center;
  border-top: 1px solid #e5e5e5;
/*   background-color: #CDF0D7; */
}
.footer-links {
  margin: 10px 0;
  padding-left: 0;
}
.footer-links li {
  display: inline;
  padding: 0 2px;
}
.footer-links li:first-child {
  padding-left: 0;
}

@media (min-width: 768px) {
  .bs-footer {
    text-align: left;
  }
  .bs-footer p {
    margin-bottom: 0;
  }
}

/*
 * Alerts
 *
 */
```

```
.alert{
        font-size: 0.95em;
        display: block;
}

/*
 * Forms
 *
 */

 @media (min-width: 0px) {
        .form-horizontal .control-label {
                text-align: right;
        }
}
.btn {
        margin: 0px 0;
}
.btn-group{
        margin-right:5px;
}

/*
 * Custom theme Bootstrap signin.css
 *
 */

.form-signin {
  max-width: 330px;
  padding: 15px;
  margin: 0 auto;
}
.form-signin .form-signin-heading,
.form-signin .checkbox {
  margin-bottom: 10px;
}
.form-signin .checkbox {
  font-weight: normal;
}
.form-signin .form-control {
  position: relative;
  font-size: 16px;
  height: auto;
  padding: 10px;
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
          box-sizing: border-box;
}
.form-signin .form-control:focus {
  z-index: 2;
}
.form-signin input[type="text"] {
  margin-bottom: -1px;
  border-bottom-left-radius: 0;
  border-bottom-right-radius: 0;
}
.form-signin input[type="password"] {
  margin-bottom: 10px;
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}

/** Custom ui buttons **/


.ui-dialog-buttonset{
        width: 100%;
}

.ui-dialog-buttonset button {
        -moz-user-select: none;
         border: 1px solid rgba(0, 0, 0, 0);
         border-radius: 4px;
         cursor: pointer;
```

```
        display: inline-block;
        font-size: 14px;
        font-weight: normal;
        line-height: 1.42857;
        margin-bottom: 0;
        padding: 6px 12px;
        text-align: center;
        vertical-align: middle;
        white-space: nowrap;
        background-color: #428BCA;
        border-color: #357EBD;
        color: #FFFFFF;
    display: block;
        padding-left: 0;
        padding-right: 0;
        width: 100%;
}
```

13.4.15.2 *Fichero "alt.css"*

```
/*css reset*/
/* Eric Meyer's Reset CSS v2.0 - http://cssreset.com */
html,body,div,span,applet,object,iframe,h1,h2,h3,h4,h5,h6,p,blockquote,pre,a,abbr,acronym,addres
s,big,cite,code,del,dfn,em,img,ins,kbd,q,s,samp,small,strike,strong,sub,sup,tt,var,b,u,i,center,
dl,dt,dd,ol,ul,li,fieldset,form,label,legend,table,caption,tbody,tfoot,thead,tr,th,td,article,as
ide,canvas,details,embed,figure,figcaption,footer,header,hgroup,menu,nav,output,ruby,section,sum
mary,time,mark,audio,video{margin-bottom: 0px;}
/*fin css reset*/

.gsc-search-button .gsc-search-button-v2{
  background: url(http://www.google.com/uds/css/v2/search_box_icon.png) no-repeat center;
  color: #fff;  background-color: #428bca;  border-color: #357ebd;   height: 24px ;
}
.gsc-search-button:hover .gsc-search-button-v2:hover{
  background: url(http://www.google.com/uds/css/v2/search_box_icon.png) no-repeat center;
  background-color: #357ebd;  border-color: #357ebd;
}

#gsc-i-id1{line-height:normal;}
 #gs_tti50{ border = 0; }
.bplay {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
150px 0px;border-radius: 12px;}
.bpause {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
100px 0px;border-radius: 12px;}
.bstop {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
50px 0px;border-radius: 12px;}
.bnext {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
0px 50px;border-radius: 12px;}
.bprev {min-width: 50px;min-height: 50px; background: url(../../resources/images/botones.png)
150px 50px;border-radius: 12px;}

#vlc{border: 2px solid #a1a1a1;
    padding: 15px;
    margin-top:10px;
    border-radius: 20px;
    background: rgb(114,140,89);
    background: linear-gradient(to bottom,  rgba(140,89,114,1) 0%,rgba(114,14,66,1)
50%,rgba(155,99,125,1) 100%); /* W3C */
    }

.dataTables_scrollBody{height:100%}

.dandelion_dataTables_export{display: none}
#logoBig{
        height: 13em;
        float: left;
        margin-right: 50px;
}

#ul_derecho{margin: -9px 0px;}
#language {padding: 3px 20px; font-weight: bold;}
```

```
#textoEstado{margin:5px;}

.quicklinks{display: none;}


#datatablesRedirect{display:none}

a {    color: #CE44D2;}

.jumbotron h1 {
  text-align: center;
  background-color: rgba(131, 55, 190, 0.33);
  border-radius: 50px 2px;
}
#reproductorPrincipal{ text-align: center;
        margin-top:10%;
        margin-bottom:10%;
        tansform:scale(2.5);
        -moz-transform:scale(2.5);
        -webkit-transform:scale(2.5);
        -o-transform:scale(2.5);}
.navbar-inverse{
        bbackground: #cb2add; /* Old browsers */
        background: -moz-linear-gradient(top,  #cb2add 0%, #460c72 50%, #8c47e0 100%); /* FF3.6+
*/
        background: -webkit-gradient(linear, left top, left bottom, color-stop(0%,#cb2add),
color-stop(50%,#460c72), color-stop(100%,#8c47e0)); /* Chrome,Safari4+ */
        background: -webkit-linear-gradient(top,  #cb2add 0%,#460c72 50%,#8c47e0 100%); /*
Chrome10+,Safari5.1+ */
        background: -o-linear-gradient(top,  #cb2add 0%,#460c72 50%,#8c47e0 100%); /* Opera
11.10+ */
        background: -ms-linear-gradient(top,  #cb2add 0%,#460c72 50%,#8c47e0 100%); /* IE10+ */
        background: linear-gradient(to bottom,  #cb2add 0%,#460c72 50%,#8c47e0 100%); /* W3C */
        filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#cb2add',
endColorstr='#8c47e0',GradientType=0 ); /* IE6-9 */

}

input[type="submit"].btn-block, input[type="reset"].btn-block, input[type="button"].btn-block {
  width: 100px;  margin: auto;
}

form#user, form#userRole, form#playListMedia {margin:auto;width:50%; min-width:400px}
#dataTables_scrollBody{hight:auto !important;}

.gvnix_dataTables_toolbar>a.submit_entity{
/*      background: #b1FF41; */
        height: 2em;
        width: 2em;
        background: url("../../resources/images/datatables/accept.png") no-repeat scroll center
center transparent;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        -o-background-size: cover;
        background-size: cover;
}
.gvnix_dataTables_toolbar>a.cancel_entity{
/*      background: #FFB145; */
        height: 2em ;
        width: 2em ;
        background: url("../../resources/images/datatables/cancel.png") no-repeat scroll center
center transparent;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        -o-background-size: cover;
        background-size: cover;
}
.gvnix_dataTables_toolbar>a.update_entity,.gvnix_dataTables_toolbar>a.delete_entity,.gvnix_dataT
ables_toolbar>a.create_entity{
        height: 2em;
        width: 2em;
        -webkit-background-size: cover;
        -moz-background-size: cover;
        -o-background-size: cover;
```

```
        background-size: cover;
}

.topnav_icon{
        width: 30px;

}

#player{
        text-align:center;
}
#l_com_mediaserver_domain_PlayListMedia{
        text-align:left;
}

a.select_all {
        background: url("../../images/datatables/book_add.png") no-repeat scroll center center
transparent;
}
a.select_none {
        background: url("../../images/datatables/book_delete.png") no-repeat scroll center
center transparent;
}

a.select_toggle {
        background: url("../../images/datatables/tick.png") no-repeat scroll center center
transparent;
}



a.update_entity {
        background: url("../../images/update.png") no-repeat scroll center center transparent;
}
a.show_entity {
        background: url("../../images/show.png") no-repeat scroll center center transparent;
}
a.create_entity {
        background: url("../../images/create.png") no-repeat scroll center center transparent;
}
a.delete_entity {
        background: url("../../images/delete.png") no-repeat scroll center center transparent;
}
a.submit_entity {
        background: url("../../images/datatables/accept.png") no-repeat scroll center center
transparent;
}
a.cancel_entity {
        background: url("../../images/datatables/cancel.png") no-repeat scroll center center
transparent;
}
/* ****** */

/*** Default  gvNIX ***/

.flag {
        height: 11px;
        width: 16px;
}

body {
  background:url(../../resources/images/pattern_1.png);
  position: relative; /* For scrollyspy */
  padding-top: 70px; /* Account for fixed navbar */
  padding-bottom: 30px;
}
        @media (min-width: 768px) and (max-width: 1020px){
                body {
                  padding-top: 120px;
                }
        }

/*
 * Main navigation
 *
```

```
 * Turn the `.navbar` at the top
 */

.navbar {
  text-shadow: 0 -1px 0 rgba(0,0,0,.15);
  background-color: #563d7c;
  border-color: #463265;
  box-shadow: 0 1px 0 rgba(255,255,255,.1);
}
.navbar .navbar-collapse {
  border-color: #463265;
}
.navbar .navbar-brand {
  color: #fff;
}
.navbar .navbar-nav > li > a {
  color: #cdbfe3;
}
.navbar .navbar-nav > li > a:hover {
        -moz-transition: color 0.3s linear;
        -o-transition: color 0.3s linear;
        -webkit-transition: color 0.3s linear;
        transition: color 0.3s linear;
        background-color: rgba( 119, 27,92, 0.68);

        background: linear-gradient(to bottom, rgba(114,14,66,1) 0%,rgba(140,,89114,1) 50%,
rgba(114,14,66,1) 100%);
        text-decoration: none;
  color: #fdd;
}
.navbar .navbar-nav > .active > a,
.navbar .navbar-nav > .active > a:hover {
  color: #fff;
  background-color: #463265;
}
.navbar .navbar-toggle {
  border-color: #563d7c;
}
.navbar .navbar-toggle:hover {
  background-color: #463265;
  border-color: #463265;
}

/* Custom theme Bootstrap for gvNIX by DISID */

.navbar {
        background-color: #822460;
        border-color: #771152;
}
.navbar .navbar-toggle {
        border-color: #608224;
}
.navbar .navbar-toggle:hover {
        background-color: #822460;
        border-color: #600E43;
}
.navbar-inverse .navbar-nav>.open>a, .navbar-inverse .navbar-nav>.open>a:hover, .navbar-inverse
.navbar-nav>.open>a:focus,
.navbar .navbar-nav > .active > a, .navbar .navbar-nav > .active > a:hover {
        color: #fff;
        background-color: #600E43;
}
.navbar .navbar-nav > li > a {
color: #fff;
 font-size: 1.2em;
}


@media (max-width: 767px){
        .navbar-inverse .navbar-nav .open .dropdown-menu>li>a {
                color: #c47fac;
                white-space: normal;
        }
        .navbar-nav .open .dropdown-menu{
         max-width: 300px;
```

```
        }
        .navbar-nav .open .dropdown-menu li{
   max-width: 300px;
  }
  .col-xs-3 button[data-toggle="offcanvas"]{
    margin: 15px 0 0;
  }
}

@media (min-width: 768px){
        .navbar-nav {
        float: none;
        margin: 0 100px;

        }
}

/* Custom sidebar menu */

#sidebar-wrapper li a{
        text-decoration: none;
        }
.affix#sidebar-wrapper{
        z-index: 100;
        }


/*
 * Footer
 *
 * Separated section of content at the bottom of all pages
 */

.bs-footer {
  padding-top: 40px;
  padding-bottom: 30px;
  margin-top: 10px;
  color: #777;
  text-align: center;
  border-top: 1px solid #e5e5e5;
/*   background-color: #CDF0D7; */
}
.footer-links {
  margin: 10px 0;
  padding-left: 0;
}
.footer-links li {
  display: inline;
  padding: 0 2px;
}
.footer-links li:first-child {
  padding-left: 0;
}

@media (min-width: 768px) {
  .bs-footer {
    text-align: left;
  }
  .bs-footer p {
    margin-bottom: 0;
  }
}

/*
 * Alerts
 *
 */

.alert{
        font-size: 0.95em;
        display: block;
}

/*
 * Forms
```

```
 *
 */

 @media (min-width: 0px) {
        .form-horizontal .control-label {
                text-align: right;
        }
}
.btn {
        margin: 0px 0;
}
.btn-group{
        margin-right:5px;
}

/*
 * Custom theme Bootstrap signin.css
 *
 */

.form-signin {
  max-width: 330px;
  padding: 15px;
  margin: 0 auto;
}
.form-signin .form-signin-heading,
.form-signin .checkbox {
  margin-bottom: 10px;
}
.form-signin .checkbox {
  font-weight: normal;
}
.form-signin .form-control {
  position: relative;
  font-size: 16px;
  height: auto;
  padding: 10px;
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
          box-sizing: border-box;
}
.form-signin .form-control:focus {
  z-index: 2;
}
.form-signin input[type="text"] {
  margin-bottom: -1px;
  border-bottom-left-radius: 0;
  border-bottom-right-radius: 0;
}
.form-signin input[type="password"] {
  margin-bottom: 10px;
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}

/** Custom ui buttons **/


.ui-dialog-buttonset{
        width: 100%;
}

.ui-dialog-buttonset button {
        -moz-user-select: none;
         border: 1px solid rgba(0, 0, 0, 0);
         border-radius: 4px;
         cursor: pointer;
         display: inline-block;
         font-size: 14px;
         font-weight: normal;
         line-height: 1.42857;
         margin-bottom: 0;
         padding: 6px 12px;
         text-align: center;
         vertical-align: middle;
```

```
        white-space: nowrap;
        background-color: #428BCA;
        border-color: #357EBD;
        color: #FFFFFF;
    display: block;
        padding-left: 0;
        padding-right: 0;
        width: 100%;
}
```

13.5 Código Fuente de la aplicación Móvil

En esta sección se incluye el código fuente de la aplicación móvil. Se han seleccionado las clases más relevantes dejando algunas triviales, o importadas.

Las clases mostradas se organizan según los paquetes a los que pertenecen.

13.5.1 Paquete Paquete com.mediaserverRC:

13.5.1.1 Fichero "MainActivity.java":

```
package com.mediaserverRC;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.AsyncTask;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Base64;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.SeekBar.OnSeekBarChangeListener;
import android.widget.TextView;

import com.mediaserverRC.R;
import com.mediaserverRC.HttpRequest.HttpRequestException;

/**
 * Actividad principal de la aplicación
 * @author Jesus
 *
 */
public class MainActivity extends Activity {
        /**
         * Dirección del servidor que retransmitirá las ordenes de la aplicación
         */
        private static  String serverAddress = "http://156.35.98.43:8080/mediaserver";
        /**
         * nombre de usuario
         */
        private static  String user="user1";
        /**
         * password del usuario
         */
        private static  String userpass="user1";
        /**
         * volumen de reproducción
         */
        private static int volumen=100;
        /**
         * estado de progreso de la reproducción
         */
        private static int progreso=0;
        /**
         * Número de servidores a los que hay que comunicar
         */
        private static int nBackEndServers=1;
        /**
         * Etiqueta de estado
         */
    private TextView lEstado;
    /**
     * Barra de volumen
```

```
     */
    private SeekBar volumenBar;
    /**
     * Etiqueta de volumen
     */
    private TextView lVol;
    /**
     * Etiqueta de progreso
     */
    private TextView lProgreso;
    /**
     * Barra de progreso
     */
    private SeekBar progressBar;
    /**
     * Botón para cambiar la reproducción a pantalla completa/ parcial
     */
    private Button fullScreenButton;
    /**
     * Flag de estado de pantalla completa/parcial
     */
    private boolean fullScreen=false;

    /**
     * Metodo llamado al crear la actividad
     */
       @Override
       protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.activity_main);
        lEstado=(TextView)findViewById(R.id.estado);
        lVol=(TextView)findViewById(R.id.label_volumen);
        lVol.setText("Vol:"+volumen);
        volumenBar=(SeekBar)findViewById(R.id.VolumenBar);
        volumenBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
               sendMessage(seekBar);
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,boolean
fromUser) {
               volumen=progress;
               lVol.setText("Vol:"+progress+"%");

            }
        });
        lProgreso=(TextView)findViewById(R.id.labelProgreso);
        progressBar=(SeekBar)findViewById(R.id.seekBarReproductor);
        progressBar.setOnSeekBarChangeListener(new OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
               sendMessage(seekBar);
               progreso=0;
               progressBar.setProgress(0);
               lProgreso.setText("Progreso:");
            }
            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,boolean
fromUser) {
               progreso=progress;
               lProgreso.setText("Progreso:"+progress+"%");

            }
        });
        fullScreenButton = (Button)findViewById(R.id.buttonFullScreen);
        fullScreenButton.setOnClickListener(new OnClickListener() {
                       @Override
                       public void onClick(View v) {
                              sendMessage(fullScreenButton);
                       }
               });
```

```
        cargarPreferencias ();
	}
	/**
	 * Metodo llamado al crear el menú de opciones
	 */
	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}
	/**
	 * Método llamado al seleccionar un elemento de menú
	 */
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_settings) {
			Intent i = new Intent(this, Preferences.class);
			startActivity(i);
			return true;
		}
		return super.onOptionsItemSelected(item);
	}

	/**
	 * Recibe un elemento View que identfica con el botón que se presionó
	 * y envía la orden correspondiente al servidor
	 * @param view
	 */
	public void sendMessage (View view){
		String url =
serverAddress+"/playersecure?u="+user+"&p="+softEncript(userpass,user);
		switch (view.getId()){
			case R.id.bPlay:
				url+="&o=play";
				break;
			case R.id.bPause:
				url+="&o=pause";
				break;
			case R.id.bStop:
				url+="&o=stop";
				break;
			case R.id.bNext:
				url+="&o=next";
				break;
			case R.id.bPrev:
				url+="&o=prev";
				break;
			case R.id.VolumenBar:
				url+="&o=vol"+volumen;
				break;
			case R.id.seekBarReproductor:
				url+="&o=pos"+progreso;
				break;
			case R.id.buttonFullScreen:
			if(fullScreen){
				url+="&o=minsc";
				fullScreen=false;}
			else{
				url+="&o=fullsc";
				fullScreen=true;}
			break;
			default:
				return;

		}
		for(int i=0;i<nBackEndServers;i++){
			new SendMessageTask().execute(url);
		}

    }
	/**
	 * Carga las preferencias del fichero de propiedades
```

```
         * con las credenciales de usuario y la direción del servidor
         */
        public void cargarPreferencias (){
                SharedPreferences pref =
PreferenceManager.getDefaultSharedPreferences(this);
                user=pref.getString("prefUsername", "");
                userpass=pref.getString("prefUserpass", "");
                serverAddress=pref.getString("prefServerAddress", "");

        }
        /**
         * Codifica la cadena recibida con el cidificador en base 64 de Android
         * @param target
         * @param key
         * @return
         */
        private String softEncript(String target, String key){
                String ret=Base64.encodeToString( target.getBytes(), Base64.DEFAULT-
key.length());
                return ret;
        }

        /**
         * Metodo llamado al retomar la actividad
         */
        @Override
        protected void onResume() {
                super.onResume();
                cargarPreferencias();
        }

        /**
         * Clase encargada de enviar los mensajes de control como una actividad en
segundo plano
         */
        private class SendMessageTask extends AsyncTask<String, Long, String> {
           /**
            * Método de envio de los mensajes de control
            */
               protected String doInBackground(String... urls) {
                try {
                       HttpRequest request=HttpRequest.post(urls[0]).accept("text/html");
                    return request.body();
                } catch (HttpRequestException exception) {
                    return null;
                }
            }
               /**
                * Analiza la respuesta del servidor e informa al usuario de la respuesta
                */
            protected void onPostExecute(String response) {
                if (response==null){
                       response="sin conexión";
                }
                lEstado.setText(response);
                //Log.i(TAG, response);
            }
        }

}
```

13.5.1.2 Fichero Preferencias.java

```
package com.mediaserverRC;

import com.mediaserverRC.R;

import android.app.Activity;
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.os.Bundle;
import android.preference.EditTextPreference;
import android.preference.ListPreference;
```

```
import android.preference.MultiSelectListPreference;
import android.preference.Preference;
import android.preference.PreferenceFragment;
import android.preference.PreferenceGroup;
import android.preference.PreferenceManager;

/**
 * Actividad de configuración de preferencias de usuario
 * @author Jesus
 *
 */
public class Preferences extends Activity {
	/**
	 * Método llamado al crear la actividad
	 * Crea el fragment de preferencias
	 */
	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		getFragmentManager().beginTransaction()
				.replace(android.R.id.content, new
PreferenciasFragment(this))
				.commit();
	}
	/**
	 * Fragment de preferencias
	 * @author Jesus
	 *
	 */
	static class PreferenciasFragment extends PreferenceFragment implements
			OnSharedPreferenceChangeListener {
		/**
		 * Preferencias del usuario
		 */
		private Preferences preferences;
		public PreferenciasFragment(Preferences p){
			this.preferences=p;
		}
		/**
		 * Método llamado al crear el fragment
		 * ccarga las preferencias actuales
		 */
		@Override
		public void onCreate(Bundle savedInstanceState) {
			super.onCreate(savedInstanceState);
			addPreferencesFromResource(R.xml.preferencias);
			PreferenceManager.setDefaultValues(preferences,
					R.xml.preferencias, false);
			initSummary(getPreferenceScreen());
		}
		/**
		 * Método llamado al restablecer el fragment
		 */
		@Override
		public void onResume() {
			super.onResume();
			// Set up a listener whenever a key changes
			getPreferenceScreen().getSharedPreferences()
					.registerOnSharedPreferenceChangeListener(this);
		}
		/**
		 * Método llamado al pausar el frament
		 */
		@Override
		public void onPause() {
			super.onPause();
			// Unregister the listener whenever a key changes
			getPreferenceScreen().getSharedPreferences()
					.unregisterOnSharedPreferenceChangeListener(this);
		}

		public void onSharedPreferenceChanged(
				SharedPreferences sharedPreferences, String key) {
			updatePrefSummary(findPreference(key));
		}

		private void initSummary(Preference p) {
```

```
                        if (p instanceof PreferenceGroup) {
                                PreferenceGroup pGrp = (PreferenceGroup) p;
                                for (int i = 0; i < pGrp.getPreferenceCount(); i++) {
                                        initSummary(pGrp.getPreference(i));
                                }
                        } else {
                                updatePrefSummary(p);
                        }
                }
                /**
                 * Actualiza las preferencias, y guarda sus valores en las etiquetas
correspondientes
                 * @param p
                 */
                private void updatePrefSummary(Preference p) {
                        if (p instanceof ListPreference) {
                                ListPreference listPref = (ListPreference) p;
                                p.setSummary(listPref.getEntry());
                        }
                        if (p instanceof EditTextPreference) {
                                EditTextPreference editTextPref = (EditTextPreference) p;
                                if (p.getTitle().toString().contains("assword")) {
                                        p.setSummary("******");
                                } else {
                                        p.setSummary(editTextPref.getText());
                                }
                        }
                        if (p instanceof MultiSelectListPreference) {
                                EditTextPreference editTextPref = (EditTextPreference) p;
                                p.setSummary(editTextPref.getText());
                        }
                }
        }
}
```

13.6Javadoc

13.6.1 Package com.mediaserverRC

Interface Summary	
Interface	**Description**
HttpRequest.ConnectionFactory	Creates `HTTP connections` for `urls`.
HttpRequest.UploadProgress	Callback interface for reporting upload progress for a request.

Class Summary	
Class	**Description**
HttpRequest	A fluid interface for making HTTP requests using an underlying`HttpURLConnection` (or sub-class).
HttpRequest.Base64	Encodes and decodes to and from Base64 notation.
HttpRequest.CloseOperation<V>	Class that ensures a `Closeable` gets closed with proper exception handling.
HttpRequest.FlushOperation<V>	Class that and ensures a `Flushable` gets flushed with proper exception handling.
HttpRequest.Operation<V>	Operation that handles executing a callback once complete and handling nested exceptions
HttpRequest.RequestOutputStream	Request output stream
MainActivity	Actividad principal de la aplicación
Preferences	Actividad de configuración de preferencias de usuario
Preferences.PreferenciasFragment	Fragment de preferencias

Exception Summary	
Exception	**Description**

HttpRequest.HttpRequestException	HTTP request exception whose cause is always an `IOException`

13.6.1.1 Class MainActivity

java.lang.Object
  Activity
    com.mediaserverRC.MainActivity

```
public class MainActivity
extends Activity
```

Actividad principal de la aplicación

Author:

Jesus

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
private class	MainActivity.SendMessageTask Clase encargada de enviar los mensajes de control como una actividad en segundo plano

Field Summary

Fields

Modifier and Type	Field and Description
private boolean	fullScreen Flag de estado de pantalla completa/parcial
private Button	fullScreenButton Botón para cambiar la reproducción a pantalla completa/ parcial
private TextView	lEstado Etiqueta de estado
private TextView	lProgreso Etiqueta de progreso
private TextView	lVol Etiqueta de volumen
private static int	nBackEndServers Número de servidores a los que hay que comunicar
private static int	progreso estado de progreso de la reproducción

private SeekBar	**progressBar** Barra de progreso
private static java.lang.String	**serverAddress** Dirección del servidor que retransmitirá las ordenes de la aplicación
private static java.lang.String	**user** nombre de usuario
private static java.lang.String	**userpass** password del usuario
private static int	**volumen** volumen de reproducción
private SeekBar	**volumenBar** Barra de volumen

Constructor Summary

Constructors
Constructor and Description
MainActivity()

Method Summary

Methods	
Modifier and Type	**Method and Description**
void	**cargarPreferencias**() Carga las preferencias del fichero de propiedades con las credenciales de usuario y la direción del servidor
protected void	**onCreate**(Bundle savedInstanceState) Metodo llamado al crear la actividad
boolean	**onCreateOptionsMenu**(Menu menu) Metodo llamado al crear el menú de opciones
boolean	**onOptionsItemSelected**(MenuItem item) Método llamado al seleccionar un elemento de menú
protected void	**onResume**() Metodo llamado al retomar la actividad
void	**sendMessage**(View view) Recibe un elemento View que identfica con el botón que se presionó y envía la orden correspondiente al servidor
private java.lang.String	**softEncript**(java.lang.String target, java.lang.String key) Codifica la cadena recibida con el cidificador en base 64 de Android

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

serverAddress

`private static java.lang.String serverAddress`
Dirección del servidor que retransmitirá las ordenes de la aplicación

user

`private static java.lang.String user`
nombre de usuario

userpass

`private static java.lang.String userpass`
password del usuario

volumen

`private static int volumen`
volumen de reproducción

progreso

`private static int progreso`
estado de progreso de la reproducción

nBackEndServers

`private static int nBackEndServers`
Número de servidores a los que hay que comunicar

lEstado

`private TextView lEstado`
Etiqueta de estado

volumenBar

`private SeekBar volumenBar`
Barra de volumen

lVol

`private TextView lVol`
Etiqueta de volumen

lProgreso

`private TextView lProgreso`
Etiqueta de progreso

progressBar

`private SeekBar progressBar`
Barra de progreso

fullScreenButton

`private Button fullScreenButton`

Botón para cambiar la reproducción a pantalla completa/ parcial

fullScreen

```
private boolean fullScreen
```
Flag de estado de pantalla completa/parcial

Constructor Detail

MainActivity

```
public MainActivity()
```

Method Detail

onCreate

```
protected void onCreate(Bundle savedInstanceState)
```
Metodo llamado al crear la actividad

onCreateOptionsMenu

```
public boolean onCreateOptionsMenu(Menu menu)
```
Metodo llamado al crear el menú de opciones

onOptionsItemSelected

```
public boolean onOptionsItemSelected(MenuItem item)
```
Método llamado al seleccionar un elemento de menú

sendMessage

```
public void sendMessage(View view)
```
Recibe un elemento View que identfica con el botón que se presionó y envía la orden correspondiente al servidor

Parameters:

`view` -

cargarPreferencias

```
public void cargarPreferencias()
```
Carga las preferencias del fichero de propiedades con las credenciales de usuario y la direción del servidor

softEncript

```
private java.lang.String softEncript(java.lang.String target,
                                     java.lang.String key)
```
Codifica la cadena recibida con el cidificador en base 64 de Android

Parameters:

`target` -
`key` -

Returns:

onResume

```
protected void onResume()
```
Metodo llamado al retomar la actividad

13.6.1.2 Class Preferences

java.lang.Object
  Activity
    com.mediaserverRC.Preferences

`public class` **`Preferences`**
`extends Activity`
Actividad de configuración de preferencias de usuario

Author:
Jesus

Nested Class Summary

Nested Classes

Modifier and Type	Class and Description
`(package private) static class`	**`Preferences.PreferenciasFragment`** Fragment de preferencias

Constructor Summary

Constructors

Constructor and Description
`Preferences``()`

Method Summary

Methods

Modifier and Type	Method and Description
`protected void`	**`onCreate`**`(Bundle savedInstanceState)` Método llamado al crear la actividad Crea el fragment de preferencias

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Preferences

`public Preferences()`

Method Detail

onCreate

`protected void onCreate(Bundle savedInstanceState)`
Método llamado al crear la actividad Crea el fragment de preferencias

13.6.1.3 Class Preferences.PreferenciasFragment

java.lang.Object
  PreferenceFragment
    com.mediaserverRC.Preferences.PreferenciasFragment

Enclosing class:
Preferences

```
static class Preferences.PreferenciasFragment
extends PreferenceFragment
```

Fragment de preferencias

Author:
Jesus

Field Summary

Fields

Modifier and Type	Field and Description
`private Preferences`	**preferences** Preferencias del usuario

Constructor Summary

Constructors

Constructor and Description
Preferences.PreferenciasFragment(Preferences p)

Method Summary

Methods

Modifier and Type	Method and Description
private void	**initSummary**(Preference p)
void	**onCreate**(Bundle savedInstanceState) Método llamado al crear el fragment ccarga las preferencias actuales
void	**onPause**() Método llamado al pausar el frament
void	**onResume**() Método llamado al restablecer el fragment
void	**onSharedPreferenceChanged**(SharedPreferences sharedPreferences, java.lang.String key)
private void	**updatePrefSummary**(Preference p) Actualiza las preferencias, y guarda sus valores en las etiquetas

correspondientes

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

preferences

```
private Preferences preferences
```
Preferencias del usuario

Constructor Detail

Preferences.PreferenciasFragment

```
public Preferences.PreferenciasFragment(Preferences p)
```

Method Detail

onCreate

```
public void onCreate(Bundle savedInstanceState)
```
Método llamado al crear el fragment ccarga las preferencias actuales

onResume

```
public void onResume()
```
Método llamado al restablecer el fragment

onPause

```
public void onPause()
```
Método llamado al pausar el frament

onSharedPreferenceChanged

```
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
                                      java.lang.String key)
```

initSummary

```
private void initSummary(Preference p)
```

updatePrefSummary

```
private void updatePrefSummary(Preference p)
```
Actualiza las preferencias, y guarda sus valores en las etiquetas correspondientes
Parameters:

`p` -

13.6.2 Package com.mediaserver.domain

Class Summary	
Class	**Description**
Media	Esta clase representara cada uno de los archivos multimedia de la aplicación, compuestos por un nombre y una ubicacion
MediaBatchService	
PlayList	Esta clase representa las listas de reproducción de la aplicación Se identifican por un nombre
PlayListBatchService	
PlayListMedia	Clase que relaciona una PlayList un elemento Media para evitar una asociación múltiple por ambos extremos
PlayListMediaBatchService	

13.6.2.1 Class Media

java.lang.Object
    com.mediaserver.domain.Media

`public class` **Media**
`extends java.lang.Object`

Esta clase representará cada uno de los archivos multimedia de la aplicación, compuestos por un nombre y una ubicacion

Author:

UO180258

Field Summary

Fields	
Modifier and Type	**Field and Description**
`private java.lang.String`	**name** Nombre del medio
`private User`	**owner** Propietario o creador del medio
`private java.lang.String`	**source** dirección en la que se ubica el medio
`private MediaType`	**type** Tipo de medio

Constructor Summary

Constructors
Constructor and Description
`Media()`

Method Summary

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

name

```
@Size(min=3,
      max=30)
private java.lang.String name
```

Nombre del medio

source

```
private java.lang.String source
```

dirección en la que se ubica el medio

type

```
private MediaType type
```

Tipo de medio

owner

```
private User owner
```

Propietario o creador del medio

Constructor Detail

Media

```
public Media()
```

13.6.2.2 Class PlayList

java.lang.Object
  com.mediaserver.domain.PlayList

```
public class PlayList
extends java.lang.Object
```

Esta clase representa las listas de reproducción de la aplicación Se identifican por un nombre

Author:

UO180258

Field Summary

Fields	
Modifier and Type	**Field and Description**
`private User`	**`owner`** Creador o propietario de la lista de reproducción
`private java.lang.String`	**`playListName`** Nombre de la lista de reproducción

Constructor Summary

Constructors	
Constructor and Description	
`PlayList``()`	

Method Summary

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

playListName

```
@Size(min=3,
      max=30)
private java.lang.String playListName
```

Nombre de la lista de reproducción

owner

```
private User owner
```

Creador o propietario de la lista de reproducción

Constructor Detail

PlayList

```
public PlayList()
```

13.6.2.3 Class PlayListMedia

java.lang.Object
  com.mediaserver.domain.PlayListMedia

All Implemented Interfaces:

java.lang.Comparable<PlayListMedia>

```
public class PlayListMedia
extends java.lang.Object
implements java.lang.Comparable<PlayListMedia>
```

Clase que relaciona una PlayList un elemento Media para evitar una asociación múltiple por ambos extremos

Author:
UO180258

Field Summary

Fields

Modifier and Type	Field and Description
private Media	**media** Medio
private PlayList	**playList** Lista de reproducción

Constructor Summary

Constructors

Constructor and Description
PlayListMedia()

Method Summary

Methods

Modifier and Type	Method and Description
int	**compareTo**(**PlayListMedia** o) Comparación para ordenación ascendente de elemenots de la clase según el nombre del medio

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

playList

```
@NotNull
private PlayList playList
```
Lista de reproducción

media

```
@NotNull
private Media media
```
Medio

Constructor Detail

PlayListMedia

```
public PlayListMedia()
```

Method Detail

compareTo

```
public int compareTo(PlayListMedia o)
```

Comparación para ordenación ascendente de elemenots de la clase según el nombre del medio

Specified by:

`compareTo` in interface `java.lang.Comparable<PlayListMedia>`

Parameters:

`o` -

Returns:

13.6.3 Package com.mediaserver.domain.security

Class Summary

Class	Description
Role	Esta clase representa los roles de los usuarios de la aplicación Según los dinstintos roles los usuarios tendrán una serie de permisios y acciones posibles u otras
RoleBatchService	
User	Esta clase representa a los usuarios de la aplicación
UserBatchService	
UserRole	Clase que relaciona un usuario con un rol para evitar la relacion múltiple por ambos extremos
UserRoleBatchService	

13.6.3.1 Class Role

java.lang.Object
  com.mediaserver.domain.security.Role

```
public class Role
extends java.lang.Object
```

Esta clase representa los roles de los usuarios de la aplicación Según los dinstintos roles los usuarios tendrán una serie de permisios y acciones posibles u otras

Author:

UO180258

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	**description** Descripción del rol
private java.lang.String	**name** Nombre del rol

Constructor Summary

Constructors

Constructor and Description
Role()
Role(java.lang.String name, java.lang.String description) Constructor de la clase, a partir de su nombre y descripción

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

name

```
@NotNull
@Size(min=1)
private java.lang.String name
```

Nombre del rol

description

```
@NotNull
@Size(max=200)
private java.lang.String description
```

Descripción del rol

Constructor Detail

Role

```
public Role()
```

Role

```
public Role(java.lang.String name,
    java.lang.String description)
```

Constructor de la clase, a partir de su nombre y descripción

Parameters:

`name` - nombre del rol
`description` - descripción del rol

13.6.3.2 Class User

java.lang.Object
com.mediaserver.domain.security.User

`public class` **`User`**
`extends java.lang.Object`
Esta clase representa a los usuarios de la aplicación

Author:

UO180258

Field Summary

Fields

Modifier and Type	Field and Description
`private java.util.Date`	**`activationDate`** Fecha de activación del usuario en el sistema
`private java.lang.String`	**`emailAddress`** Dirección de email del usuario, será unica para cada usuario
`private java.lang.Boolean`	**`enabled`** Estado de habilitación, permite activar a los usuarios para que puedan acceder al sistema una vez se han registrado
`private java.lang.String`	**`firstName`** Nombre del usuario
`private java.lang.String`	**`lastName`** Apellido del usuario
`private java.lang.Boolean`	**`locked`** Estado de bloqueo, permite bloquar el acceso a los usuarios al sistema
`private java.lang.String`	**`password`** password de acceso del usuario

Constructor Summary

Constructors

Constructor and Description
`User``()`

Method Summary

Methods

Modifier and Type	Method and Description

static java.lang.Long	**countFindUsersByEmailAddress**(java.lang.String emailAddress) Encuenta un usuario identificado por su id a partir de su dirección de email
static javax.persistence.TypedQuery<**User**>	**findUsersByEmailAddress**(java.lang.String emailAddress) Devuelve una TypeddQuery de usuarios a partir de su email
static javax.persistence.TypedQuery<**User**>	**findUsersByEmailAddress**(java.lang.String emailAddress, java.lang.String sortFieldName, java.lang.String sortOrder) Devuelve una TypeddQuery de usuarios a partir de su email ordenados según el parametro sortOrder
static **User**	**getContextUser**() Este metodo devuelve el usuario que está utilizando el sistema en la sesión actual obteniendolo sus datos del contexto y completando los datos del objeto con los de la base de datos

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

firstName

```
@NotNull
@Size(min=1)
private java.lang.String firstName
```

Nombre del usuario

lastName

```
@NotNull
@Size(min=1)
private java.lang.String lastName
```

Apellido del usuario

emailAddress

```
@NotNull
@Size(min=1)
private java.lang.String emailAddress
```

Dirección de email del usuario, será unica para cada usuario

password

```
@NotNull(groups=CreateAttributes.class)
@Size(min=1)
```

```
private java.lang.String password
```
password de acceso del usuario

activationDate

```
private java.util.Date activationDate
```
Fecha de activación del usuario en el sistema

enabled

```
private java.lang.Boolean enabled
```
Estado de habilitación, permite activar a los usuarios para que puedan acceder al sistema una vez se han registrado

locked

```
private java.lang.Boolean locked
```
Estado de bloqueo, permite bloquar el acceso a los usuarios al sistema

Constructor Detail

User

```
public User()
```

Method Detail

getContextUser

```
public static User getContextUser()
```
Este metodo devuelve el usuario que está utilizando el sistema en la sesión actual obteniendolo sus datos del contexto y completando los datos del objeto con los de la base de datos

Returns:

countFindUsersByEmailAddress

```
public
static java.lang.Long countFindUsersByEmailAddress(java.lang.
String emailAddress)
```
Encuenta un usuario identificado por su id a partir de su dirección de email

Parameters:

`emailAddress -`

Returns:

findUsersByEmailAddress

```
public
static javax.persistence.TypedQuery<User> findUsersByEmailAdd
ress(java.lang.String emailAddress)
```
Devuelve una TypeddQuery de usuarios a partir de su email

Parameters:

`emailAddress -`

Returns:

findUsersByEmailAddress

```
public
static javax.persistence.TypedQuery<User> findUsersByEmailAdd
ress(java.lang.String emailAddress,
```

```
java.lang.String sortFieldName,

java.lang.String sortOrder)
```

Devuelve una TypeddQuery de usuarios a partir de su email ordenados según el parametro sortOrder

Parameters:

`emailAddress` -
`sortFieldName` -
`sortOrder` - puede tener los valores ASC para orden ascendente o DESC para orden descendente

Returns:

13.6.3.3 Class UserRole

java.lang.Object
com.mediaserver.domain.security.UserRole

```
public class UserRole
extends java.lang.Object
```

Clase que relaciona un usuario con un rol para evitar la relacion múltiple por ambos extremos

Author:

UO180258

Field Summary

Constructor Summary

Constructors
Constructor and Description
`UserRole()`

Method Summary

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

user

```
@NotNull
```

```
private User user
```
Usuario objetivo de la relación

role

```
@NotNull
private Role role
```
Rol objetivo de la relación

Constructor Detail

UserRole

```
public UserRole()
```

13.6.4 Package com.mediaserver.reference

Enum Summary

Enum	Description
MediaType	Enumerado de tipos de archivos multimedia (clase Media)

13.6.4.1 Enum MediaType

java.lang.Object
  java.lang.Enum<MediaType>
    com.mediaserver.reference.MediaType

All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable<MediaType>

```
public enum MediaType
extends java.lang.Enum<MediaType>
```
Enumerado de tipos de archivos multimedia (clase Media)

Author:

UO180258

Enum Constant Summary

Enum Constants

Enum Constant and Description
Audio
Video

Method Summary

Methods

Modifier and Type	Method and Description
static MediaType	valueOf(java.lang.String name) Returns the enum constant of this type with the specified name.

`static MediaType[] values()`	Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

`clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf`

Methods inherited from class java.lang.Object

`getClass, notify, notifyAll, wait, wait, wait`

Enum Constant Detail

Video

`public static final MediaType Video`

Audio

`public static final MediaType Audio`

Method Detail

values

`public static MediaType[] values()`

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (MediaType c : MediaType.values())
    System.out.println(c);
```

Returns:

an array containing the constants of this enum type, in the order they are declared

valueOf

`public static MediaType valueOf(java.lang.String name)`

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

`name` - the name of the enum constant to be returned.

Returns:

the enum constant with the specified name

Throws:

`java.lang.IllegalArgumentException` - if this enum type has no constant with the specified name

`java.lang.NullPointerException` - if the argument is null

13.6.5 Package com.mediaserver.security

Class Summary

Class	Description
JpaUserDetailService	Load user and his/her roles using JPA
RoleBasedAuthenticationSuccessHandler	

13.6.5.1 Class JpaUserDetailService

java.lang.Object
  com.mediaserver.security.JpaUserDetailService
All Implemented Interfaces:
    org.springframework.security.core.userdetails.UserDetailsServi

---

```
public class JpaUserDetailService
extends java.lang.Object
implements
org.springframework.security.core.userdetails.UserDetailsSe
rvice
```

Load user and his/her roles using JPA

Constructor Summary

Constructors

Constructor and Description
JpaUserDetailService()

Method Summary

Methods

Modifier and Type	Method and Description
private java.util.List<org.springframework.security.core.GrantedAuthority>	loadGrantedAuthorities(User targetUser)
org.springframework.security.core.userdetails.UserDetails	loadUserByUsername(java.lang.String username)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JpaUserDetailService

```
public JpaUserDetailService()
```

Method Detail

loadUserByUsername

```
public org.springframework.security.core.userdetails.UserDetails loadUserByUsername(java.lang.String username)

throws
org.springframework.security.core.userdetails.UsernameNotFoundException
```

Specified by:

`loadUserByUsername` in interface `org.springframework.security.core.userdetails.UserDetailsService`

Throws:

`org.springframework.security.core.userdetails.UsernameNotFoundException`

loadGrantedAuthorities

```
private java.util.List<org.springframework.security.core.GrantedAuthority> loadGran
```

13.6.5.2 Class RoleBasedAuthenticationSuccessHandler

java.lang.Object
  com.mediaserver.security.RoleBasedAuthenticationSuccessHandler

All Implemented Interfaces:

org.springframework.security.web.authentication.AuthenticationSuccessHandler

---

```
public class RoleBasedAuthenticationSuccessHandler
extends java.lang.Object
implements
org.springframework.security.web.authentication.AuthenticationSuccessHandler
```

Field Summary

Fields

Modifier and Type	Field and Description
private java.util.Map<java.lang.String,java.lang.String>	roleUrlMap

Constructor Summary

Constructors

Constructor and Description
RoleBasedAuthenticationSuccessHandler()

Method Summary

Methods

Modifier and Type	Method and Description
void	**onAuthenticationSuccess**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response, org.springframework.security.core.Authentication authentication)
void	**setRoleUrlMap**(java.util.Map<java.lang.String,java.lang.String> roleUrlMap)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

roleUrlMap

```
private java.util.Map<java.lang.String,java.lang.String> roleUrlMap
```

Constructor Detail

RoleBasedAuthenticationSuccessHandler

```
public RoleBasedAuthenticationSuccessHandler()
```

Method Detail

onAuthenticationSuccess

```
public void onAuthenticationSuccess(javax.servlet.http.HttpServletRequest request,

javax.servlet.http.HttpServletResponse response,

org.springframework.security.core.Authentication authentication)
                                throws java.io.IOException,

javax.servlet.ServletException
```

Specified by:

onAuthenticationSuccess in interface org.springframework.security.web.authentication.AuthenticationSuccessHandler

Throws:

java.io.IOException
javax.servlet.ServletException

setRoleUrlMap

```
public void setRoleUrlMap(java.util.Map<java.lang.String,java
.lang.String> roleUrlMap)
```

13.6.6 Package com.mediaserver.util

Class Summary	
Class	**Description**
EncryptionUtil	

13.6.6.1 Class EncryptionUtil

java.lang.Object
  com.mediaserver.util.EncryptionUtil

```
public class EncryptionUtil
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields	
Modifier and Type	**Field and Description**
private static java.lang.String	UTF_8

Constructor Summary

Constructors	
Constructor and Description	
EncryptionUtil()	

Method Summary

Methods	
Modifier and Type	**Method and Description**
static void	decrypt(java.lang.String encryptionKey, java.io.InputStream is, java.io.OutputStream os)
static java.lang.String	decrypt(java.lang.String encryptionKey, java.lang.String encryptedText)
private static void	doCopy(java.io.InputStream is,

	java.io.OutputStream os)
static void	**encrypt**(java.lang.String encryptionKey, java.io.InputStream is, java.io.OutputStream os)
static java.lang.String	**encrypt**(java.lang.String encryptionKey, java.lang.String plainText)
private static void	**encryptOrDecrypt**(java.lang.String encryptionKey, int mode, java.io.InputStream is, java.io.OutputStream os)
private static byte[]	**toBytes**(java.lang.String string)
private static java.lang.String	**toString**(byte[] bytes)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

UTF_8

```
private static final java.lang.String UTF_8
```

See Also:

Constant Field Values

Constructor Detail

EncryptionUtil

```
public EncryptionUtil()
```

Method Detail

encrypt

```
public static void encrypt(java.lang.String encryptionKey,
           java.io.InputStream is,
           java.io.OutputStream os)
                    throws java.security.InvalidKeyException,
                           java.io.IOException
```

Throws:

java.security.InvalidKeyException
java.io.IOException

decrypt

```
public static void decrypt(java.lang.String encryptionKey,
           java.io.InputStream is,
           java.io.OutputStream os)
                   throws java.security.InvalidKeyException,
```

```
                                java.io.IOException
```

Throws:

```
java.security.InvalidKeyException
java.io.IOException
```

encryptOrDecrypt

```
private
static void encryptOrDecrypt(java.lang.String encryptionKey,
                     int mode,
                     java.io.InputStream is,
                     java.io.OutputStream os)
                              throws
java.security.InvalidKeyException,
                                       java.io.IOException
```

Throws:

```
java.security.InvalidKeyException
java.io.IOException
```

toBytes

```
private static byte[] toBytes(java.lang.String string)
```

toString

```
private static java.lang.String toString(byte[] bytes)
```

doCopy

```
private static void doCopy(java.io.InputStream is,
          java.io.OutputStream os)
                    throws java.io.IOException
```

Throws:

```
java.io.IOException
```

encrypt

```
public
static java.lang.String encrypt(java.lang.String encryptionKe
y,
                        java.lang.String plainText)
                                 throws
java.security.InvalidKeyException
```

Throws:

```
java.security.InvalidKeyException
```

decrypt

```
public
static java.lang.String decrypt(java.lang.String encryptionKe
y,
                        java.lang.String encryptedText)
                                 throws
java.security.InvalidKeyException
```

Throws:

```
java.security.InvalidKeyException
```

13.6.7 Package com.mediaserver.web

Class Summary	
Class	**Description**
ApplicationConversionServiceFactoryBean	A central place to register application converters and formatters.
MediaController	Controlador de la clase Media
PlayListController	Controlador de la clase PlayList
PlayListMediaController	Controlador de la clase PlayListMedia

13.6.7.1 Class ApplicationConversionServiceFactoryBean

java.lang.Object
  org.springframework.format.support.FormattingConversionServiceFactoryBean
    com.mediaserver.web.ApplicationConversionServiceFactoryBean
    All Implemented Interfaces:
      org.springframework.beans.factory.Aware,
      org.springframework.beans.factory.FactoryBean<org.springframework.format.support.FormattingConversionService>, org.springframework.beans.factory.InitializingBean,
      org.springframework.context.EmbeddedValueResolverAware

```
public class ApplicationConversionServiceFactoryBean
extends
org.springframework.format.support.FormattingConversionServiceFactoryBean
```

A central place to register application converters and formatters.

Constructor Summary

Constructors	
Constructor and Description	
ApplicationConversionServiceFactoryBean()	

Method Summary

Methods	
Modifier and Type	**Method and Description**
org.springframework.core.convert.converter.Converter<Media,java.lang.String>	getMediaToStringConverter() Convierte un medio en una cadea de texo mostrable por la interfaz
org.springframework.core.convert.converter.Converter<PlayList,java.lang.String>	getPlayListToStringConverter() Convierte una lista de reproducción en una cadea de texo mostrable por la interfaz
org.springframework.core.c	getRoleToStringConverter()

onvert.converter.Converter<Role,java.lang.String>	Convierte un rol en una cadea de texo mostrable por la interfaz
org.springframework.core.convert.converter.Converter<User,java.lang.String>	getUserToStringConverter() Convierte un usuario en una cadea de texo mostrable por la interfaz
protected void	installFormatters(org.springframework.format.FormatterRegistry registry)

Methods inherited from class org.springframework.format.support.FormattingConversionServiceFactoryBean

afterPropertiesSet, getObject, getObjectType, isSingleton, setConverters, setEmbeddedValueResolver, setFormatterRegistrars, setFormatters, setRegisterDefaultFormatters

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ApplicationConversionServiceFactoryBean

```
public ApplicationConversionServiceFactoryBean()
```

Method Detail

installFormatters

```
protected void installFormatters(org.springframework.format.FormatterRegistry registry)
```

Overrides:

`installFormatters` in class `org.springframework.format.support.FormattingConversionServiceFactoryBean`

getUserToStringConverter

```
public org.springframework.core.convert.converter.Converter<User,java.lang.String> getUserToStringConverter()
```

Convierte un usuario en una cadea de texo mostrable por la interfaz

Returns:

getRoleToStringConverter

```
public org.springframework.core.convert.converter.Converter<Role,java.lang.String> getRoleToStringConverter()
```

Convierte un rol en una cadea de texo mostrable por la interfaz

Returns:

getMediaToStringConverter

```
public org.springframework.core.convert.converter.Converter<Media,java.lang.String> getMediaToStringConverter()
```
Convierte un medio en una cadea de texo mostrable por la interfaz
Returns:

getPlayListToStringConverter

```
public org.springframework.core.convert.converter.Converter<PlayList,java.lang.String> getPlayListToStringConverter()
```
Convierte una lista de reproducción en una cadea de texo mostrable por la interfaz
Returns:

13.6.7.2 Class MediaController

java.lang.Object
com.mediaserver.web.MediaController

```
@RequestMapping(value="/medias")
@Controller
public class MediaController
extends java.lang.Object
```
Controlador de la clase Media
Author:
UO180258

Constructor Summary

Constructors	
Constructor and Description	
`MediaController()`	

Method Summary

Methods	
Modifier and Type	**Method and Description**
`java.lang.String`	`create(Media media, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel, javax.servlet.http.HttpServletRequest httpServletRequest)` Crea un nuevo medio en el sistema y devuelve la vista del elemento creado
`org.gvnix.web.json.JsonResponse<java.util.List<Media>>`	`createBatch(java.util.List<Media> medias, org.springframework.validation.BindingResult bindingResult, javax.servlet.http.HttpServletRequest request)` Creción AJAX de nuevos medios.
`org.springframework.ht`	`deleteBatch(boolean all,`

tp. ResponseEntity<java.lang.Object>	java.lang.Boolean deleteIn, java.util.List<java.lang.Long> idList, **Media** media, org.springframework.web.context.request.WebRequest request) Elimina un medio mediante una petición AJAX y devuelve el resultado para mostrarlo sin necisidad de recargar la página
com.github.dandelion.datatables. core.ajax. DatatablesResponse <java.util.Map<java.lang.String,java.lang.String>>	**findAllMedias**(com.github.dandelion.datatables.core.ajax.DatatablesCriterias criterias,**Media** media, javax.servlet.http.HttpServletRequest request) Encuentra los medios de la aplicación, todos en caso de que el usuario del contexto sea administrador y solo los del usuario concreto en otro caso
com.github.dandelion.datatables.core. ajax.DatatablesResponse <java.util.Map<java.lang.String,java.lang.String>>	**findMediasByType**(com.github.dandelion.datatables.core.ajax.DatatablesCriterias criterias,**MediaType** type) Encuenta los medios del sistema por tipo, todas para administradores, solo las del usuario en otro caso
java.util.Map <java.lang.String, java.lang.Object>	**getPropertyMap**(**Media** media, javax.servlet.http.HttpServletRequest request)
java.lang.String	**list**(java.lang.Integer page, java.lang.Integer size, java.lang.String sortFieldName, java.lang.String sortOrder, org.springframework.ui.Model uiModel) Produce una vista con el listado de medios
private void	**unBind**(**Media** media)
java.lang.String	**update**(**Media** media, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel, javax.servlet.http.HttpServletRequest httpServletRequest) Modifica un medio del sistema
org.gvnix.web.json. JsonResponse<java.util.List<**Media**>>	**updateBatch**(java.util.List<**Media**> medias, org.springframework.validation.BindingResult bindingResult, javax.servlet.http.HttpServletRequest request) Actualiza mediante petición AJAx un medio y devuelve el

resultado para mostrar sin necesidad de actualizar la página

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MediaController

```
public MediaController()
```

Method Detail

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.Object> getPr
opertyMap(Media media,

javax.servlet.http.HttpServletRequest request)
```

list

```
@RequestMapping(produces="text/html")
public java.lang.String list(@RequestParam(value="page",requi
red=false)
                                    java.lang.Integer page,

@RequestParam(value="size",required=false)
                                    java.lang.Integer size,

@RequestParam(value="sortFieldName",required=false)

java.lang.String sortFieldName,

@RequestParam(value="sortOrder",required=false)

java.lang.String sortOrder,

org.springframework.ui.Model uiModel)
```

Produce una vista con el listado de medios

Parameters:

`page` - número de pagnia si es necesario paginar los resultados
`sizetamaÃ±o` -
`sortFieldName` - Nombre del campo por el que se ordenan las filas mostradas
`sortOrder` - Orden ascenente o descendente
`uiModel` - modelos de interfaz

Returns:

create

```
@RequestMapping(method=POST,
                produces="text/html")
```

```
public java.lang.String create(Media media,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel,

javax.servlet.http.HttpServletRequest httpServletRequest)
```

Crea un nuevo medio en el sistema y devuelve la vista del elemento creado

Parameters:

```
media -
bindingResult -
uiModel -
httpServletRequest -
```

Returns:

findAllMedias

```
@RequestMapping(headers="Accept=application/json",
                value="/datatables/ajax",
                produces="application/json")
@ResponseBody
public com.github.dandelion.datatables.core.ajax.DatatablesRe
sponse<java.util.Map<java.lang.String,java.lang.String>> find
AllMedias(@DatatablesParams

com.github.dandelion.datatables.core.ajax.DatatablesCriterias
 criterias,

@ModelAttribute

Media media,

javax.servlet.http.HttpServletRequest request)
```

Encuentra los medios de la aplicación, todos en caso de que el usuario del contexto sea administrador y solo los del usuario concreto en otro caso

Parameters:

```
criterias -
media -
request -
```

Returns:

createBatch

```
@RequestMapping(produces="application/json",
                consumes="application/json",
                method=POST,
                headers="Accept=application/json")
@ResponseBody
public org.gvnix.web.json.JsonResponse<java.util.List<Media>>
 createBatch(@RequestBody

java.util.List<Media> medias,

org.springframework.validation.BindingResult bindingResult,
```

```
javax.servlet.http.HttpServletRequest request)
```

Creción AJAX de nuevos medios. Se aÃ±aden a la lista de medios y se les asigna el usuario que solicitó su creación como propietario. Devuelve una respuesta jon para mostrar los resultados sin necesidad de recargar la página.

Parameters:

```
medias -
bindingResult -
request -
```

Returns:

update

```
@RequestMapping(method=PUT,
                produces="text/html")
public java.lang.String update(Media media,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel,

javax.servlet.http.HttpServletRequest httpServletRequest)
```

Modifica un medio del sistema

Parameters:

```
media -
bindingResult -
uiModel -
httpServletRequest -
```

Returns:

updateBatch

```
@RequestMapping(produces="application/json",
                consumes="application/json",
                method=PUT,
                headers="Accept=application/json")
@ResponseBody
public org.gvnix.web.json.JsonResponse<java.util.List<Media>>
 updateBatch(@RequestBody

java.util.List<Media> medias,

org.springframework.validation.BindingResult bindingResult,

javax.servlet.http.HttpServletRequest request)
```

Actualiza mediante petición AJAx un medio y devuelve el resultado para mostrar sin necesidad de actualizar la página

Parameters:

```
medias -
bindingResult -
request -
```

Returns:

deleteBatch

```
@RequestMapping(value="/delete",
                produces="application/json",
                method=POST)
public org.springframework.http.ResponseEntity<java.lang.Obje
ct> deleteBatch(@RequestParam(value="all",required=false)

boolean all,

@RequestParam(value="deleteIn",required=false)

java.lang.Boolean deleteIn,

@RequestParam(value="idList[]",required=false)

java.util.List<java.lang.Long> idList,

@ModelAttribute

Media media,

org.springframework.web.context.request.WebRequest request)
```

Elimina un medio mediante una petición AJAX y devuelve el resultado para mostrarlo sin necisidad de recargar la página

Parameters:

```
all -
deleteIn -
idList -
media -
request -
```

Returns:

unBind

```
private void unBind(Media media)
```

findMediasByType

```
@RequestMapping(headers="Accept=application/json",
                value="/datatables/ajax",
                params="ajax_find=ByType",
                produces="application/json")
@ResponseBody
public com.github.dandelion.datatables.core.ajax.DatatablesRe
sponse<java.util.Map<java.lang.String,java.lang.String>> find
MediasByType(@DatatablesParams

com.github.dandelion.datatables.core.ajax.DatatablesCriterias
 criterias,

@RequestParam(value="type")

MediaType type)
```

Encuenta los medios del sistema por tipo, todas para administradores, solo las del usuario en otro caso

Parameters:

```
criterias -
```

type -

Returns:

13.6.7.3 Class PlayListController

java.lang.Object
com.mediaserver.web.PlayListController

```
@RequestMapping(value="/playlists")
@Controller
public class PlayListController
extends java.lang.Object
```

Controlador de la clase PlayList

Constructor Summary

Constructors

Constructor and Description
PlayListController()

Method Summary

Methods

Modifier and Type	Method and Description
java.lang.String	create(PlayList playList, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel, javax.servlet.http.HttpServletRequest httpServletRequest) crea una nueva PlayList en el sistema
org.gvnix.web.json.JsonResponse<java.util.List<PlayList>>	createBatch(java.util.List<PlayList> playLists, org.springframework.validation.BindingResult bindingResult, javax.servlet.http.HttpServletRequest request) Crea una nueva PlayList en el sistema mediante petición AJAX para incorporarla al listado completo y que se muestre sin necesidad de recargar la página. asigna el usuario actual como su propietario
org.springframework.http.ResponseEntity<java.lang.Object>	deleteBatch(boolean all, java.lang.Boolean deleteIn, java.util.List<java.lang.Long> idList, PlayList playList, org.springframework.web.context.request.WebRequest request) Elimina una PlayList o conjunto de ellas mediante

	petición AJAx para poder actualizar la vista sin necesidad de recargar la página
com.github.dandelion.datatables.core.ajax.DatatablesResponse<java.util.Map<java.lang.String,java.lang.String>>	**findAllPlayLists**(com.github.dandelion.datatables.core.ajax.DatatablesCriterias criterias,**PlayList** playList, javax.servlet.http.HttpServletRequest request) Busca las PlayLsit de la aplicación y las devuelve mediante AJAX para que no sea necesario recargar la página Mostrará todas para los administradores, y solo las del usuario en otro caso
java.util.Map<java.lang.String,java.lang.Object>	**getPropertyMap**(**PlayList** playList, javax.servlet.http.HttpServletRequest request)
java.lang.String	**list**(java.lang.Integer page, java.lang.Integer size, java.lang.String sortFieldName, java.lang.String sortOrder, org.springframework.ui.Model uiModel) Lista las PlayList de la aplicación, todas para administrador, y solo las del usuario en otro caso
java.lang.String	**show**(java.lang.Long id, org.springframework.ui.Model uiModel) Muestra una PlayList se le especifica su id como parámetro
private void	**unBind**(**PlayList** playlist) Elimina los vinculos de la Playlist, todos los objetos PlayListMedia relacionados
java.lang.String	**update**(**PlayList** playList, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel, javax.servlet.http.HttpServletRequest httpServletRequest) Actualiza una PlayList
org.gvnix.web.json.JsonResponse<java.util.List<**PlayList**>>	**updateBatch**(java.util.List<**PlayList**> playLists, org.springframework.validation.BindingResult bindingResult, javax.servlet.http.HttpServletRequest request) Actualiza una PlayList mediante petición AJAx para poder mostrar los resultados sin necesidad de actualizar la página

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PlayListController

```
public PlayListController()
```

Method Detail

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.Object> getPr
opertyMap(PlayList playList,

javax.servlet.http.HttpServletRequest request)
```

list

```
@RequestMapping(produces="text/html")
public java.lang.String list(@RequestParam(value="page",requi
red=false)
                                    java.lang.Integer page,

@RequestParam(value="size",required=false)
                                    java.lang.Integer size,

@RequestParam(value="sortFieldName",required=false)

java.lang.String sortFieldName,

@RequestParam(value="sortOrder",required=false)

java.lang.String sortOrder,

org.springframework.ui.Model uiModel)
```

Lista las PlayList de la aplicación, todas para administrador, y solo las del usuario en otro caso

Parameters:

page -
size -
sortFieldName -
sortOrder -
uiModel -

Returns:

create

```
@RequestMapping(method=POST,
                produces="text/html")
public java.lang.String create(PlayList playList,

org.springframework.validation.BindingResult bindingResult,
```

```
org.springframework.ui.Model uiModel,

javax.servlet.http.HttpServletRequest httpServletRequest)
```

crea una nueva PlayList en el sistema

Parameters:

```
playList -
bindingResult -
uiModel -
httpServletRequest -
```

Returns:

findAllPlayLists

```
@RequestMapping(headers="Accept=application/json",
                value="/datatables/ajax",
                produces="application/json")
@ResponseBody
public com.github.dandelion.datatables.core.ajax.DatatablesRe
sponse<java.util.Map<java.lang.String,java.lang.String>> find
AllPlayLists(@DatatablesParams

com.github.dandelion.datatables.core.ajax.DatatablesCriterias
 criterias,

@ModelAttribute

PlayList playList,

javax.servlet.http.HttpServletRequest request)
```

Busca las PlayLsit de la aplicación y las devuelve mediante AJAX para que no sea necesario recargar la página Mostrará todas para los administradores, y solo las del usuario en otro caso

Parameters:

```
criterias -
playList -
request -
```

Returns:

createBatch

```
@RequestMapping(produces="application/json",
                consumes="application/json",
                method=POST,
                headers="Accept=application/json")
@ResponseBody
public org.gvnix.web.json.JsonResponse<java.util.List<PlayLis
t>> createBatch(@RequestBody

java.util.List<PlayList> playLists,

org.springframework.validation.BindingResult bindingResult,

javax.servlet.http.HttpServletRequest request)
```

Crea una nueva PlayList en el sistema mediante petición AJAX para incorporarla al listado completo y que se muestre sin necesidad de recargar la página. asigna el usuario actual como su propietario

Parameters:

playLists -
bindingResult -
request -

Returns:

show

```
@RequestMapping(value="/{id}",
                produces="text/html")
public java.lang.String show(@PathVariable(value="id")
                                    java.lang.Long id,

org.springframework.ui.Model uiModel)
```

Muestra una PlayList se le especifica su id como parámetro

Parameters:

id -
uiModel -

Returns:

deleteBatch

```
@RequestMapping(value="/delete",
                produces="application/json",
                method=POST)
public org.springframework.http.ResponseEntity<java.lang.Obje
ct> deleteBatch(@RequestParam(value="all",required=false)

boolean all,

@RequestParam(value="deleteIn",required=false)

java.lang.Boolean deleteIn,

@RequestParam(value="idList[]",required=false)

java.util.List<java.lang.Long> idList,

@ModelAttribute

PlayList playList,

org.springframework.web.context.request.WebRequest request)
```

Elimina una PlayList o conjunto de ellas mediante petición AJAx para poder actualizar la vista sin necesidad de recargar la página

Parameters:

all -
deleteIn -
idList -
playList -
request -

Returns:

unBind

```
private void unBind(PlayList playlist)
```
Elimina los vinculos de la Playlist, todos los objetos PlayListMedia relacionados

Parameters:

```
playlist -
```

update

```
@RequestMapping(method=PUT,
                produces="text/html")
public java.lang.String update(PlayList playList,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel,

javax.servlet.http.HttpServletRequest httpServletRequest)
```
Actualiza una PlayList

Parameters:

```
playList -
bindingResult -
uiModel -
httpServletRequest -
```

Returns:

updateBatch

```
@RequestMapping(produces="application/json",
                consumes="application/json",
                method=PUT,
                headers="Accept=application/json")
@ResponseBody
public org.gvnix.web.json.JsonResponse<java.util.List<PlayLis
t>> updateBatch(@RequestBody

java.util.List<PlayList> playLists,

org.springframework.validation.BindingResult bindingResult,

javax.servlet.http.HttpServletRequest request)
```
Actualiza una PlayList mediante petición AJAx para poder mostrar los resultados sin necesidad de actualizar la página

Parameters:

```
playLists -
bindingResult -
request -
```

Returns:

13.6.7.4 Class PlayListMediaController

java.lang.Object
  com.mediaserver.web.PlayListMediaController

```
@RequestMapping(value="/playlistmedias")
@Controller
```

```
public class PlayListMediaController
extends java.lang.Object
```

Controlador de la clase PlayListMedia

Author:

UO180258

Constructor Summary

Constructors	
Constructor and Description	
`PlayListMediaController()`	

Method Summary

Methods	
Modifier and Type	**Method and Description**
`private java.lang.Long`	`comprobarIdPlayList(java.lang.String referer)` Encuenta el indice de la playlist que originó la peticion de inserción con objetivo de poder crear un PlayListMedia asociado a al lista correcta
`org.gvnix.web.json.JsonResponse<java.util.List<PlayListMedia>>`	`createBatch(java.util.List<PlayListMedia> playListMedias, org.springframework.validation.BindingResult bindingResult, javax.servlet.http.HttpServletRequest request)` Crea un playListMedia mediante petición AJAx para poder reflejar el resultado sin necesidad de recargar la página
`com.github.dandelion.datatables.core.ajax.DatatablesResponse<java.util.Map<java.lang.String,java.lang.String>>`	`findAllPlayListMedias(com.github.dandelion.datatables.core.ajax.DatatablesCriterias criterias,PlayListMedia PlayListMedia, javax.servlet.http.HttpServletRequest request)` Busca todos los PlayListMedia de la aplicación mediante AJAx para poder mostrarlos sin necesidad de actualizar la página
`java.util.Map<java.lang.String,java.lang.Object>`	`getPropertyMap(PlayListMedia playListMedia, javax.servlet.http.HttpServletRequest request)`
`(package private) void`	`populateEditForm(org.springframework.ui.Model uiModel, PlayListMedia playListMedia)`

	Rellena el formulario de edición de PlayListMedias, con odos los medios y PlayLists para administraador, y solo los del usuario en otro caso

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

PlayListMediaController

```
public PlayListMediaController()
```

Method Detail

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.Object> getPr
opertyMap(PlayListMedia playListMedia,

javax.servlet.http.HttpServletRequest request)
```

populateEditForm

```
void populateEditForm(org.springframework.ui.Model uiModel,
                      PlayListMedia playListMedia)
```

Rellena el formulario de edición de PlayListMedias, con odos los medios y PlayLists para administraador, y solo los del usuario en otro caso

Parameters:

`uiModel` -
`playListMedia` -

findAllPlayListMedias

```
@RequestMapping(headers="Accept=application/json",
                value="/datatables/ajax",
                produces="application/json")
@ResponseBody
public com.github.dandelion.datatables.core.ajax.DatatablesRe
sponse<java.util.Map<java.lang.String,java.lang.String>> find
AllPlayListMedias(@DatatablesParams

com.github.dandelion.datatables.core.ajax.DatatablesCriterias
 criterias,

@ModelAttribute

PlayListMedia PlayListMedia,

javax.servlet.http.HttpServletRequest request)
```

Busca todos los PlayListMedia de la aplicación mediante AJAx para poder mostrarlos sin necesidad de actualizar la página

Parameters:

criterias -
PlayListMedia -
request -

Returns:

createBatch

```
@RequestMapping(produces="application/json",
                consumes="application/json",
                method=POST,
                headers="Accept=application/json")
@ResponseBody
public org.gvnix.web.json.JsonResponse<java.util.List<PlayLis
tMedia>> createBatch(@RequestBody

java.util.List<PlayListMedia> playListMedias,

org.springframework.validation.BindingResult bindingResult,

javax.servlet.http.HttpServletRequest request)
```

Crea un playListMedia mediante petición AJAx para poder reflejar el resultado sin necesidad de recargar la página

Parameters:

playListMedias -
bindingResult -
request -

Returns:

comprobarIdPlayList

```
private java.lang.Long comprobarIdPlayList(java.lang.String r
eferer)
```

Encuenta el indice de la playlist que originó la peticion de inserción con objetivo de poder crear un PlayListMedia asociado a al lista correcta

Parameters:

referer -

Returns:

13.6.8 Package com.mediaserver.web.security

Class Summary

Class	Description
ChangePasswordController	
ChangePasswordForm	
ForgotPasswordController	
ForgotPasswordForm	
NewPasswordForm	

RoleController
SignUpController
SignUpForm
UserController
UserRoleController

13.6.8.1 Class ChangePasswordController

java.lang.Object
  com.mediaserver.web.security.ChangePasswordController

```
@RequestMapping(value="/passwd")
@Controller
public class ChangePasswordController
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields	
Modifier and Type	**Field and Description**
private org.springframework.security.crypto.password.PasswordEncoder	passwordEncoder

Constructor Summary

Constructors	
Constructor and Description	
ChangePasswordController()	

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	createForm(org.springframework.ui.Model uiModel)
private void	populateEditForm(org.springframework.ui.Model uiModel,ChangePasswordForm form)
java.lang.String	submit(ChangePasswordForm form, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel)

java.lang.String	**thanks**()
private void	**validateMore**(**ChangePasswordForm** form, org.springframework.validation.BindingResult bindingResult)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

passwordEncoder

```
@Autowired
private org.springframework.security.crypto.password.PasswordEncoder passwordEncoder
```

Constructor Detail

ChangePasswordController

```
public ChangePasswordController()
```

Method Detail

createForm

```
@RequestMapping
public java.lang.String createForm(org.springframework.ui.Model uiModel)
```

submit

```
@RequestMapping(method=POST)
public java.lang.String submit(@ModelAttribute(value="form")
                               ChangePasswordForm form,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel)
```

thanks

```
@RequestMapping(value="/thanks")
public java.lang.String thanks()
```

validateMore

```
private void validateMore(ChangePasswordForm form,

org.springframework.validation.BindingResult bindingResult)
```

populateEditForm

```
private void populateEditForm(org.springframework.ui.Model uiModel,
```

```
ChangePasswordForm form)
```

13.6.8.2 Class ChangePasswordForm

java.lang.Object
  com.mediaserver.web.security.ChangePasswordForm

```
public class ChangePasswordForm
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	**newPassword**
private java.lang.String	**oldPassword**
private java.lang.String	**retypeNewPassword**

Constructor Summary

Constructors

Constructor and Description
ChangePasswordForm()

Method Summary

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait
```

Field Detail

oldPassword

```
@NotNull
private java.lang.String oldPassword
```

newPassword

```
@NotNull
```

```
@Size(min=5,
      max=100)
private java.lang.String newPassword
```

retypeNewPassword

```
private java.lang.String retypeNewPassword
```

Constructor Detail

ChangePasswordForm

```
public ChangePasswordForm()
```

13.6.8.3 Class ForgotPasswordController

java.lang.Object
  com.mediaserver.web.security.ForgotPasswordController

```
@RequestMapping(value="/forgot")
@Controller
public class ForgotPasswordController
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	encryptionKey
private int	hoursToExpire
private org.springframework.mail .javamail.JavaMailSenderImpl	mailSender
private org.springframework.mail .SimpleMailMessage	mailTemplate
private org.springframework. security.crypto.password .PasswordEncoder	passwordEncoder

Constructor Summary

Constructors

Constructor and Description
ForgotPasswordController()

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	createForm(org.springframework.ui.Model uiModel)
java.lang.String	createFormSetNewPassword(java.lang.String key, org.springframework.ui.Model uiModel)
private void	populateEditForm(org.springframework.ui.Model uiModel, ForgotPasswordForm form)
private void	populateEditForm(org.springframework.ui.Model uiModel, NewPasswordForm form)
private void	sendAccountRecoveryRequestMail(User user, java.lang.String key)
void	setEncryptionKey(java.lang.String encryptionKey)
void	setHoursToExpire(int hoursToExpire)
java.lang.String	submit(ForgotPasswordForm form, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel)
java.lang.String	submitSetNewPassword(java.lang.String key, NewPasswordForm form, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel)
java.lang.String	thanks()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

mailSender

```
@Autowired
```

```
private
transient org.springframework.mail.javamail.JavaMailSenderImp
l mailSender
```

mailTemplate

```
@Autowired
@Qualifier(value="accountRecoveryMailTemplate")
private org.springframework.mail.SimpleMailMessage
mailTemplate
```

encryptionKey

```
private java.lang.String encryptionKey
```

hoursToExpire

```
private int hoursToExpire
```

passwordEncoder

```
@Autowired
private org.springframework.security.crypto.password.Password
Encoder passwordEncoder
```

Constructor Detail

ForgotPasswordController

```
public ForgotPasswordController()
```

Method Detail

createForm

```
@RequestMapping
public java.lang.String createForm(org.springframework.ui.Mod
el uiModel)
```

submit

```
@RequestMapping(method=POST)
public java.lang.String submit(@ModelAttribute(value="form")
                                        ForgotPasswordForm form,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel)
                          throws
java.io.UnsupportedEncodingException,

java.security.InvalidKeyException
```

Throws:

```
java.io.UnsupportedEncodingException
java.security.InvalidKeyException
```

createFormSetNewPassword

```
@RequestMapping(value="/set",
```

```
                params="k")
public java.lang.String createFormSetNewPassword(@RequestPara
m(value="k")

java.lang.String key,

org.springframework.ui.Model uiModel)
```

submitSetNewPassword

```
@RequestMapping(value="/set",
                params="k",
                method=POST)
public java.lang.String submitSetNewPassword(@RequestParam(va
lue="k")

java.lang.String key,

@ModelAttribute(value="form")

NewPasswordForm form,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel)
```

thanks

```
@RequestMapping(value="/thanks")
public java.lang.String thanks()
```

populateEditForm

```
private void populateEditForm(org.springframework.ui.Model ui
Model,
                      ForgotPasswordForm form)
```

populateEditForm

```
private void populateEditForm(org.springframework.ui.Model ui
Model,
                      NewPasswordForm form)
```

sendAccountRecoveryRequestMail

```
private void sendAccountRecoveryRequestMail(User user,
                                    java.lang.String key)
                                       throws
java.io.UnsupportedEncodingException
```

Throws:

java.io.UnsupportedEncodingException

setHoursToExpire

```
public void setHoursToExpire(int hoursToExpire)
```

setEncryptionKey

```
public void setEncryptionKey(java.lang.String encryptionKey)
```

13.6.8.4 Class ForgotPasswordForm

java.lang.Object
  com.mediaserver.web.security.ForgotPasswordForm

```
public class ForgotPasswordForm
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	**emailAddress**

Constructor Summary

Constructors

Constructor and Description
ForgotPasswordForm()

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

emailAddress

```
@NotNull
@Email
private java.lang.String emailAddress
```

Constructor Detail

ForgotPasswordForm

```
public ForgotPasswordForm()
```

13.6.8.5 Class NewPasswordForm

java.lang.Object
  com.mediaserver.web.security.NewPasswordForm

```
public class NewPasswordForm
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	**newPassword**
private java.lang.String	**retypeNewPassword**

Constructor Summary

Constructors

Constructor and Description
NewPasswordForm()

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

newPassword

```
@NotNull
@Size(min=5)
private java.lang.String newPassword
```

retypeNewPassword

```
private java.lang.String retypeNewPassword
```

Constructor Detail

NewPasswordForm

```
public NewPasswordForm()
```

13.6.8.6 Class RoleController

java.lang.Object
  com.mediaserver.web.security.RoleController

```
@RequestMapping(value="/security/roles")
@Controller
public class RoleController
extends java.lang.Object
```

Constructor Summary

Constructors	
Constructor and Description	
RoleController()	

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.util.Map<java.lang.String,java.lang.Object>	getPropertyMap(Role role, javax.servlet.http.HttpServletRequest request)

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait
```

Constructor Detail

RoleController

```
public RoleController()
```

Method Detail

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.Object> getPr
opertyMap(Role role,

javax.servlet.http.HttpServletRequest request)
```

13.6.8.7 Class SignUpController

java.lang.Object
com.mediaserver.web.security.SignUpController

```
@RequestMapping(value="/signup")
@Controller
public class SignUpController
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields

Modifier and Type	Field and Description
private boolean	**isAccountAutoEnabled**
private org.springframework.mail.javamail.JavaMailSenderImpl	**mailSender**
private org.springframework.mail.SimpleMailMessage	**mailTemplate**
private org.springframework.security.crypto.password.PasswordEncoder	**passwordEncoder**

Constructor Summary

Constructors

Constructor and Description
SignUpController()

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**activateUser**(java.lang.String activationKey, java.lang.String emailAddress, org.springframework.ui.Model uiModel)
private void	**activateUserNoKey**(**User** user)
private java.lang.String	**createActivationKey**(**User** user)
java.lang.String	**createSignUpForm**(org.springframework.ui.Model uiModel)
private void	**populateEditForm**(org.springframework.ui.Model uiModel, **SignUpForm** form)
private void	**sendActivationMail**(java.lang.String activationKey, **User** user)
void	**setAccountAutoEnabled**(boolean isAccountAutoEnabled)
static void	**setUserRole**(**User** user)
java.lang.String	**submitSignUpForm**(**SignUpForm** form, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel)
java.lang.String	**thanks**(org.springframework.ui.Model uiModel)
private void	**validateMore**(**SignUpForm** form, org.springframework.validation.BindingResult bindingResult)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

passwordEncoder

```
@Autowired
private org.springframework.security.crypto.password.PasswordEncoder passwordEncoder
```

mailSender

```
@Autowired
private
transient org.springframework.mail.javamail.JavaMailSenderImp
l mailSender
```

mailTemplate

```
@Autowired
@Qualifier(value="accountActivationMailTemplate")
private org.springframework.mail.SimpleMailMessage
mailTemplate
```

isAccountAutoEnabled

```
private boolean isAccountAutoEnabled
```

Constructor Detail

SignUpController

```
public SignUpController()
```

Method Detail

createSignUpForm

```
@RequestMapping(method=GET)
public java.lang.String createSignUpForm(org.springframework.
ui.Model uiModel)
```

submitSignUpForm

```
@RequestMapping(method=POST)
public java.lang.String submitSignUpForm(@ModelAttribute(valu
e="form")

SignUpForm form,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel)
```

setUserRole

```
public static void setUserRole(User user)
```

validateMore

```
private void validateMore(SignUpForm form,

org.springframework.validation.BindingResult bindingResult)
```

activateUser

```
@RequestMapping(params="k",
                method=GET)
public java.lang.String activateUser(@RequestParam(value="k",
required=true)
```

```
java.lang.String activationKey,

@RequestParam(value="e",required=true)

java.lang.String emailAddress,

org.springframework.ui.Model uiModel)
```

activateUserNoKey

```
private void activateUserNoKey(User user)
```

thanks

```
@RequestMapping(value="/thanks")
public java.lang.String thanks(org.springframework.ui.Model u
iModel)
```

sendActivationMail

```
private void sendActivationMail(java.lang.String activationKe
y,
                        User user)
                           throws javax.mail.MessagingException
```

Throws:

```
javax.mail.MessagingException
```

createActivationKey

```
private java.lang.String createActivationKey(User user)
```

populateEditForm

```
private void populateEditForm(org.springframework.ui.Model ui
Model,
                      SignUpForm form)
```

setAccountAutoEnabled

```
public void setAccountAutoEnabled(boolean isAccountAutoEnable
d)
```

13.6.8.8 Class SignUpForm

java.lang.Object
  com.mediaserver.web.security.SignUpForm

```
public class SignUpForm
extends java.lang.Object
```

Author:

rohit

Field Summary

Fields	
Modifier and Type	**Field and Description**
private java.lang.String	**emailAddress**
private java.lang.String	**firstName**
private java.lang.String	**lastName**
private java.lang.String	**password**
private java.lang.String	**recaptchaChallenge**
private java.lang.String	**reCaptchaErrorMessage**
private java.lang.String	**recaptchaResponse**
private net.tanesha.recaptcha.ReCaptcha	**reCatcpha**
private java.lang.String	**retypePassword**

Constructor Summary

Constructors	
Constructor and Description	
SignUpForm()	

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	**getReCaptchaHtml**()
boolean	**isValidCaptcha**()
void	**setRecaptcha_challenge_field**(java.lang.String recaptchaChallenge)
void	**setRecaptcha_response_field**(java.lang.String recaptchaResponse)

Methods inherited from class java.lang.Object

```
clone, equals, finalize, getClass, hashCode, notify, notifyAll,
toString, wait, wait, wait
```

Field Detail

firstName

```
@NotNull
@Size(min=1,
      max=100)
private java.lang.String firstName
```

lastName

```
@NotNull
@Size(min=1,
      max=100)
private java.lang.String lastName
```

emailAddress

```
@NotNull
@Size(min=1,
      max=100)
@Email
private java.lang.String emailAddress
```

password

```
@NotNull
@Size(min=5,
      max=100)
private java.lang.String password
```

retypePassword

```
private java.lang.String retypePassword
```

recaptchaChallenge

```
private java.lang.String recaptchaChallenge
```

recaptchaResponse

```
private java.lang.String recaptchaResponse
```

reCatcpha

```
private net.tanesha.recaptcha.ReCaptcha reCatcpha
```

reCaptchaErrorMessage

```
private java.lang.String reCaptchaErrorMessage
```

Constructor Detail

SignUpForm

```
public SignUpForm()
```

Method Detail

setRecaptcha_challenge_field

```
public void setRecaptcha_challenge_field(java.lang.String rec
aptchaChallenge)
```

setRecaptcha_response_field

```
public void setRecaptcha_response_field(java.lang.String reca
ptchaResponse)
```

getReCaptchaHtml

```
public java.lang.String getReCaptchaHtml()
```

isValidCaptcha

```
public boolean isValidCaptcha()
```

13.6.8.9 Class UserController

java.lang.Object
  com.mediaserver.web.security.UserController

```
@RequestMapping(value="/security/users")
@Controller
public class UserController
extends java.lang.Object
```

Author:

'Muhammad Ichsan'

Field Summary

Fields

Modifier and Type	Field and Description
private org.springframework.mail.MailSender	mailSender
private org.springframework.mail.SimpleMailMessage	mailTemplate
private org.springframework.security.crypto.password.PasswordEncoder	passwordEncoder
private javax.validation.Validator	validator

Constructor Summary

Constructors
Constructor and Description
`UserController()`

Method Summary

Methods	
Modifier and Type	**Method and Description**
java.lang.String	`create(User user, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel, javax.servlet.http.HttpServletRequest httpServletRequest)`
java.lang.String	`delete(java.lang.Long id, java.lang.Integer page, java.lang.Integer size, org.springframework.ui.Model uiModel)`
org.springframework.http.ResponseEntity<java.lang.Object>	`deleteBatch(boolean all, java.lang.Boolean deleteIn, java.util.List<java.lang.Long> idList, User user, org.springframework.web.context.request.WebRequest request)`
java.util.Map<java.lang.String,java.lang.Object>	`getPropertyMap(User user, javax.servlet.http.HttpServletRequest request)`
private void	`sendActivationMail(User user)`
private void	`unBind(User user)`
java.lang.String	`update(User user, org.springframework.validation.BindingResult bindingResult, org.springframework.ui.Model uiModel, javax.servlet.http.HttpServletRequest httpServletRequest)`

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

passwordEncoder

```
@Autowired
private org.springframework.security.crypto.password.Password
Encoder passwordEncoder
```

mailSender

```
@Autowired
private transient org.springframework.mail.MailSender
mailSender
```

mailTemplate

```
@Autowired
@Qualifier(value="accountActivationMailTemplate")
private org.springframework.mail.SimpleMailMessage
mailTemplate
```

validator

```
@Autowired
private javax.validation.Validator validator
```

Constructor Detail

UserController

```
public UserController()
```

Method Detail

create

```
@RequestMapping(method=POST, produces="text/html")
public java.lang.String create(User user,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel,

javax.servlet.http.HttpServletRequest httpServletRequest)
```

update

```
@RequestMapping(method=PUT, produces="text/html")
public java.lang.String update(User user,

org.springframework.validation.BindingResult bindingResult,

org.springframework.ui.Model uiModel,

javax.servlet.http.HttpServletRequest httpServletRequest)
```

sendActivationMail

```
private void sendActivationMail(User user)
```

delete

```
@RequestMapping(value="/{id}",
                method=DELETE,
                produces="text/html")
public java.lang.String delete(@PathVariable(value="id")
                                    java.lang.Long id,

@RequestParam(value="page",required=false)
                                    java.lang.Integer page,

@RequestParam(value="size",required=false)
                                    java.lang.Integer size,

org.springframework.ui.Model uiModel)
```

deleteBatch

```
@RequestMapping(value="/delete",
                produces="application/json",
                method=POST)
public org.springframework.http.ResponseEntity<java.lang.Obje
ct> deleteBatch(@RequestParam(value="all",required=false)

boolean all,

@RequestParam(value="deleteIn",required=false)

java.lang.Boolean deleteIn,

@RequestParam(value="idList[]",required=false)

java.util.List<java.lang.Long> idList,

@ModelAttribute

User user,

org.springframework.web.context.request.WebRequest request)
```

unBind

```
private void unBind(User user)
```

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.Object> getPr
opertyMap(User user,

javax.servlet.http.HttpServletRequest request)
```

13.6.8.10 Class UserRoleController

java.lang.Object
  com.mediaserver.web.security.UserRoleController

```
@RequestMapping(value="/security/assignrole")
@Controller
public class UserRoleController
extends java.lang.Object
```

Constructor Summary

Constructors
Constructor and Description
`UserRoleController()`

Method Summary

Methods	
Modifier and Type	**Method and Description**
`java.util.Map<java.lang.String,java.lang.Object>`	`getPropertyMap(UserRole userRole, javax.servlet.http.HttpServletRequest request)`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

UserRoleController

```
public UserRoleController()
```

Method Detail

getPropertyMap

```
public java.util.Map<java.lang.String,java.lang.Object> getPropertyMap(UserRole userRole,

javax.servlet.http.HttpServletRequest request)
```

13.6.9 Package com.websocket.config

Class Summary	
Class	**Description**
CORSFilter	Enabling CORS support - Access-Control-Allow-Origin
WebSocketConfig	

13.6.9.1 Class CORSFilter

java.lang.Object
  org.springframework.web.filter.GenericFilterBean
    org.springframework.web.filter.OncePerRequestFilter
      com.websocket.config.CORSFilter

All Implemented Interfaces:

javax.servlet.Filter, org.springframework.beans.factory.Aware, org.springframework.beans.factory.BeanNameAware, org.springframework.beans.factory.DisposableBean, org.springframework.beans.factory.InitializingBean, org.springframework.context.EnvironmentAware, org.springframework.web.context.ServletContextAware

---

```
public class CORSFilter
extends org.springframework.web.filter.OncePerRequestFilter
```

Enabling CORS support - Access-Control-Allow-Origin

Author:

zeroows@gmail.com `cors com.elm.mb.rest.filters.CORSFilter cors /*`

Field Summary

Fields	
Modifier and Type	**Field and Description**
`private static org.apache.commons.logging.Log`	LOG

Fields inherited from class org.springframework.web.filter.OncePerRequestFilter

`ALREADY_FILTERED_SUFFIX`

Fields inherited from class org.springframework.web.filter.GenericFilterBean

`logger`

Constructor Summary

Constructors
Constructor and Description
`CORSFilter()`

Method Summary

Methods	
Modifier and Type	**Method and Description**
`protected void`	`doFilterInternal(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response, javax.servlet.FilterChain filterChain)`

Methods inherited from class org.springframework.web.filter.OncePerRequestFilter

`doFilter, getAlreadyFilteredAttributeName, isAsyncDispatch, isAsyncStarted, shouldNotFilter, shouldNotFilterAsyncDispatch, shouldNotFilterErrorDispatch`

Methods inherited from class org.springframework.web.filter.GenericFilterBean

`addRequiredProperty, afterPropertiesSet, destroy, getFilterConfig, getFilterName, getServletContext, init, initBeanWrapper, initFilterBean, setBeanName, setEnvironment, setServletContext`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

LOG

```
private static final org.apache.commons.logging.Log LOG
```

Constructor Detail

CORSFilter

```
public CORSFilter()
```

Method Detail

doFilterInternal

```
protected void doFilterInternal(javax.servlet.http.HttpServle
tRequest request,

javax.servlet.http.HttpServletResponse response,
                    javax.servlet.FilterChain filterChain)
                         throws
javax.servlet.ServletException,
                                  java.io.IOException
```

Specified by:

`doFilterInternal` in class `org.springframework.web.filter.OncePerRequestFilter`

Throws:

`javax.servlet.ServletException`
`java.io.IOException`

13.6.9.2 Class WebSocketConfig

java.lang.Object
  com.websocket.config.WebSocketConfig

All Implemented Interfaces:

org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer

---

```
@Profile(value="default")
@EnableWebSocketMessageBroker
@EnableScheduling
@Controller
public class WebSocketConfig
extends java.lang.Object
implements
org.springframework.web.socket.config.annotation.WebSocketM
essageBrokerConfigurer
```

Constructor Summary

Constructors

Constructor and Description
`WebSocketConfig()`

Method Summary

Methods

Modifier and Type	Method and Description
`void`	`configureClientInboundChannel(org.springframework.messaging.simp.config.ChannelRegistration registrati`

	on)
void	**configureClientOutboundChannel**(org.springframework.messaging.simp.config.ChannelRegistration registration)
void	**configureMessageBroker**(org.springframework.messaging.simp.config.MessageBrokerRegistry config) Configure message broker options.
void	**registerStompEndpoints**(org.springframework.web.socket.config.annotation.StompEndpointRegistry registry) Configure STOMP

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

WebSocketConfig

```
public WebSocketConfig()
```

Method Detail

registerStompEndpoints

```
public void registerStompEndpoints(org.springframework.web.socket.config.annotation.StompEndpointRegistry registry)
```

Configure STOMP

Specified by:

registerStompEndpoints in interface org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer

configureMessageBroker

```
public void configureMessageBroker(org.springframework.messaging.simp.config.MessageBrokerRegistry config)
```

Configure message broker options.

Specified by:

configureMessageBroker in interface org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer

configureClientInboundChannel

```
public void configureClientInboundChannel(org.springframework.messaging.simp.config.ChannelRegistration registration)
```

Specified by:

configureClientInboundChannel in interface org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer

configureClientOutboundChannel

```
public void configureClientOutboundChannel(org.springframework.messaging.simp.config.ChannelRegistration registration)
```

Specified by:

`configureClientOutboundChannel` in interface `org.springframework.web.socket.config.annotation.WebSocketMessageBrokerConfigurer`

13.6.10 Package com.websocket.controllers

Class Summary

Class	Description
WebSocketBroadcastController	

13.6.10.1 Class WebSocketBroadcastController

java.lang.Object
  com.websocket.controllers.WebSocketBroadcastController

```
@Controller
public class WebSocketBroadcastController
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
`private org.springframework.security.crypto.password.StandardPasswordEncoder`	**passwordEncoder**
`private org.springframework.messaging.simp.SimpMessagingTemplate`	**template**

Constructor Summary

Constructors

Constructor and Description
`WebSocketBroadcastController()`

Method Summary

Methods	
Modifier and Type	**Method and Description**
`java.lang.String`	`CambiarReproductorSecure(java.lang.String order, java.lang.String userName, java.lang.String userPass)` Metodo de control remoto, destinado a test Paso de parametros por get
`java.lang.String`	`CambiarReproductorSecurePost(java.lang.String order, java.lang.String userName, java.lang.String userPass)` Reciabe ordenes de control remoto por POST
`private boolean`	`checkUser(java.lang.String userName)`
`java.lang.String`	`handleException(java.lang.Throwable exception)`
`void`	`processMessageFromClient(SimpleMessage message, java.security.Principal principal)`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

template

```
@Autowired
private org.springframework.messaging.simp.SimpMessagingTemplate template
```

passwordEncoder

```
@Autowired
private org.springframework.security.crypto.password.StandardPasswordEncoder passwordEncoder
```

Constructor Detail

WebSocketBroadcastController

```
public WebSocketBroadcastController()
```

Method Detail

processMessageFromClient

```
@MessageMapping(value="/simplemessages")
public void processMessageFromClient(SimpleMessage message,

java.security.Principal principal)
                                throws java.lang.Exception
```

Throws:

`java.lang.Exception`

handleException

```
@MessageExceptionHandler
@SendToUser(value="/queue/errors")
public java.lang.String handleException(java.lang.Throwable e
xception)
```

CambiarReproductorSecure

```
@RequestMapping(value="/playersecure",
                method=GET)
@ResponseBody
public java.lang.String CambiarReproductorSecure(@RequestPara
m(value="o",required=true)

java.lang.String order,

@RequestParam(value="u",required=true)

java.lang.String userName,

@RequestParam(value="p",required=true)

java.lang.String userPass)
                                                throws
java.lang.Exception
```

Metodo de control remoto, destinado a test Paso de parametros por get

Parameters:

`order` -
`userName` -
`userPass` -

Returns:

Throws:

`java.lang.Exception`

CambiarReproductorSecurePost

```
@RequestMapping(value="/playersecure",
                method=POST)
@ResponseBody
public java.lang.String CambiarReproductorSecurePost(@Request
Param(value="o",required=true)
```

```
java.lang.String order,

@RequestParam(value="u",required=true)

java.lang.String userName,

@RequestParam(value="p",required=true)

java.lang.String userPass)
                                                    throws
java.lang.Exception
```

Reciabe ordenes de control remoto por POST

Parameters:

order -
userName -
userPass -

Returns:

Throws:

java.lang.Exception

checkUser

```
private boolean checkUser(java.lang.String userName)
```

13.6.11 Package com.websocket.model

Class Summary	
Class	**Description**
MessageBroadcast	Clase que representa el mensaje básico de multidifusión
SimpleMessage	

13.6.11.1 Class MessageBroadcast

java.lang.Object
  com.websocket.model.MessageBroadcast

```
public class MessageBroadcast
extends java.lang.Object
```

Clase que representa el mensaje básico de multidifusión

Author:

UO180258

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	**messageContent**

Constructor Summary

Constructors

Constructor and Description
MessageBroadcast()
MessageBroadcast(java.lang.String messageContent)

Method Summary

Methods

Modifier and Type	Method and Description
java.lang.String	**getMessageContent**()
void	**setMessageContent**(java.lang.String messageContent)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

messageContent

```
private java.lang.String messageContent
```

Constructor Detail

MessageBroadcast

```
public MessageBroadcast()
```

MessageBroadcast

```
public MessageBroadcast(java.lang.String messageContent)
```

Method Detail

getMessageContent

```
public java.lang.String getMessageContent()
```

setMessageContent

```
public void setMessageContent(java.lang.String messageContent
)
```

13.6.11.2 Class SimpleMessage

java.lang.Object
  com.websocket.model.SimpleMessage

```
public class SimpleMessage
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
private java.lang.String	message

Constructor Summary

Constructors

Constructor and Description
SimpleMessage()

Method Summary

Methods

Modifier and Type	Method and Description

java.lang.String	**getMessage**()
void	**setMessage**(java.lang.String message)

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

message

private java.lang.String message

Constructor Detail

SimpleMessage

public SimpleMessage()

Method Detail

getMessage

public java.lang.String getMessage()

setMessage

public void setMessage(java.lang.String message)

13.6.12 Package com.websocket.util

Class Summary

Class	Description
Base64	Utilities for encoding and decoding the Base64 representation of binary data.
Base64.Coder	
Base64.Decoder	
Base64.Encoder	
Util	

13.6.12.1 Class Util

java.lang.Object
  com.websocket.util.Util

```
public final class Util
extends java.lang.Object
```

`Util` is for some utility methods.

Since:
1.0

Version:
1.0.0.1

Author:
Sunit Katkar

Constructor Summary

Constructors	
Constructor and Description	
`Util()`	

Method Summary

Methods	
Modifier and Type	**Method and Description**
`static java.lang.String`	`getSimpleDate()` Utility method to get a simple human readable date and time using the default locale set for the JVM on the server is used.
`static java.lang.String`	`getSimpleDate(java.util.Locale locale)` Utility method to get a simple human readable date and time

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Util

```
public Util()
```

Method Detail

getSimpleDate

```
public static java.lang.String getSimpleDate()
```

Utility method to get a simple human readable date and time using the default locale set for the JVM on the server is used.

Returns:

- Simple date and time

getSimpleDate

```
public
static java.lang.String getSimpleDate(java.util.Locale locale
)
```

Utility method to get a simple human readable date and time

Parameters:

`locale` - - If null then the default locale set for the JVM on the server is used.

Returns:

- Simple date and time