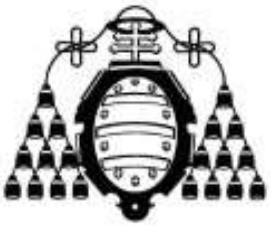
UNIVERSIDAD DE OVIEDO



ESCUELA DE INGENIERÍA INFORMÁTICA DE OVIEDO

PROYECTO FIN DE MÁSTER

PORTAL PARA LA GESTIÓN DE DOMINIOS WEB

DIRECTOR: Fernando Álvarez García

AUTOR: Borja Garrido Bear

Vº Bº del Director del Proyecto

Agradecimientos

El desarrollo completo de este proyecto debe su ser a gran serie de personas que han ido apareciendo durante el mismo y a muchas otras que simplemente han estado allí siempre y el desarrollo de este proyecto no ha supuesto una excepción para recibir su ayuda.

Primeramente agradecer a mi familia y amigos por el apoyo y los ánimos recibidos durante todo el desarrollo del proyecto.

A Ecocomputer S.L., en especial a Myrtha y Agustín, por su colaboración e interés en el proyecto y su desarrollo.

A todos los compañeros que han utilizado la base de este proyecto para hacer funcionar los suyos y gracias a los cuales se ha conseguido perfilar tanto su funcionamiento, en especial a Mario Rey y a David Cabañeros pues también aportaron ideas sobre cómo debería funcionar.

A mi compañero de trabajo David Moreno por su aportación de ideas para mejorar la configuración de las conexiones.

Y a Fernando Álvarez, mi director de Trabajo Fin de Máster, por su aportación de ideas sobre cómo sobrellevar muchos de los inconvenientes que el proyecto ha tenido y aquellos muchos detalles del desarrollo que se me escapaban.

Resumen

La aplicación desarrollada permite a los empleados de una empresa gestionar en primera instancia los permisos que los usuarios tendrán sobre los distintos módulos de la Intranet y en menor medida gestionar gran parte del proceso de adquisición de dominios para clientes externos. La aplicación ha sido desarrollada a petición de la empresa Ecocomputer S.L., bajo su supervisión y en colaboración con ellos.

Concretamente la aplicación proporciona la base para el resto de módulos que se incluirán sobre la Intranet (dos de ellos abarcados por este proyecto), encargándose de la identificación de los usuarios, proporcionar un sistema de log a medida de la empresa y una pequeña guía de trabajo para que estos sean lo más homogéneo posible.

El módulo de control de usuarios proporcionará soporte de permisos de lectura y escritura a nivel de perfil y de usuario para todos los módulos de la Intranet.

El módulo de gestión de dominios proporcionará una pequeña interfaz para interconectar la intranet de la empresa con el servidor ISPConfig, el cual será el encargado en última instancia de administrar los dominios.

La aplicación se desplegará en un servidor Apache Tomcat, desarrollada bajo las tecnologías Java y la base de datos estará bajo SQL Server 2005.

Palabras Clave

Intranet, Gestión de usuarios, Gestión de dominios, Java, Struts2, Tiles.

Abstract

The developed project allows the company workers to manage firstly the level access that users will have over the different Intranet's modules, and secondly manage a big part of the domain acquisition process for external clients. The application had been developed as a request from Ecocomputer S.L and under its supervision and collaboration.

Specifically, the application provides a base for all the other modules that will be included within the intranet (two of them covered by this project), handling the user login process, providing a custom log system for the company and a little baseline for those project to be as homogeneous as possible.

The user management module provides write and read level access for roles and users in all the intranet modules.

The domain management module provides a little interface in order to interconnect the company intranet with the ISPConfig server, which at the end, will be in charge for the complete domain management.

The application will be deployed over an Apache Tomcat server, developed under Java technologies and a SQL Server 2005 database.

Keywords

Intranet, Users management, domain management, Java, Struts2, Tiles.

Índice General

CAPÍTULO 1. MEMORIA DEL PROYECTO 25

1.1 RESUMEN DE LA MOTIVACIÓN, OBJETIVOS Y ALCANCE DEL PROYECTO 25
1.2 RESUMEN DE TODOS LOS ASPECTOS 25
1.2.1 Capítulo 1: Memoria del Proyecto 25
1.2.2 Capítulo 2: Introducción 25
1.2.3 Capítulo 3: Aspectos teóricos 26
1.2.4 Capítulo 4: Planificación del Proyecto 26
1.2.5 Capítulo 5: Análisis 26
1.2.6 Capítulo 6: Diseño del Sistema 26
1.2.7 Capítulo 7: Implementación del Sistema 26
1.2.8 Capítulo 8: Desarrollo de Pruebas 26
1.2.9 Capítulo 9: Manuales del Sistema 26
1.2.10 Capítulo 10: Conclusiones y ampliaciones 27
1.2.11 Capítulo 11: Presupuesto 27
1.2.12 Capítulo 12: Referencias Bibliográficas 27
1.2.13 Capítulo 13: Apéndice 27

CAPÍTULO 2. INTRODUCCIÓN 29

2.1 JUSTIFICACIÓN DEL PROYECTO 29
2.2 OBJETIVOS DEL PROYECTO 30
2.2.1 Creación de la infraestructura necesaria para la intranet 30
2.2.2 Módulo de gestión de dominios 30
2.3 ESTUDIO DE LA SITUACIÓN ACTUAL 32
2.3.1 Evaluación de Alternativas 33

CAPÍTULO 3. ASPECTOS TEÓRICOS 35

3.1 DOMINIOS DE INTERNET 35
3.1.1 Sintaxis 35
3.1.2 Dominios de nivel más alto 35
3.1.3 Registro de dominios 36
3.2 SERVLET 37
3.2.1 Definición e historia 37
3.2.2 Características 37
3.2.3 Aplicación en el proyecto 37
3.3 JAVASCRIPT 38
3.3.1 Definición e historia 38
3.3.2 Características 38
3.3.3 Aplicación en el proyecto 38
3.4 SQL 39
3.4.1 Definición e historia 39
3.4.2 Características 39
3.4.3 Aplicación en el proyecto 39
3.5 SQL SERVER 40
3.5.1 Definición e historia 40
3.5.2 Características 40
3.5.3 Aplicación en el proyecto 40

3.6 JDBC....41
3.6.1 Definición e historia....41
3.6.2 Características....41
3.6.3 Aplicación en el proyecto....41
3.7 APACHE TOMCAT....42
3.7.1 Definición e historia....42
3.7.2 Características....42
3.7.3 Aplicación en el proyecto....42
3.8 APACHE STRUTS....43
3.8.1 Definición e historia....43
3.8.2 Características....43
3.8.3 Aplicación en el proyecto....43
3.9 QMAIL....44
3.9.1 Definición e historia....44
3.9.2 Características....44
3.9.3 Aplicación en el proyecto....44
3.10 ANDROID....45
3.10.1 Definición e historia....45
3.10.2 Características....45
3.10.3 Aplicación en el proyecto....45
3.11 SCRUM....46
3.11.1 Definición e historia....46
3.11.2 Características....46
3.11.3 Aplicación en el proyecto....47

CAPÍTULO 4. PLANIFICACIÓN DEL PROYECTO....48

4.1 PLANIFICACIÓN....48
4.1.1 Diagramas de Gantt....48
4.1.2 Planificación final extendida....50

CAPÍTULO 5. ANÁLISIS....53

5.1 SPRINT 0....53
5.1.1 Definición del Sistema....53
5.1.2 Requisitos del Sistema....54
5.1.3 Identificación de actores del sistema....58
5.2 RESTO DEL PROYECTO....59
5.2.1 Definición del sistema....59
5.2.2 Requisitos del sistema....60
5.2.3 Identificación de actores del sistema....64
5.2.4 Especificación de Casos de Uso....66
5.2.5 Identificación de los Subsistemas en la Fase de Análisis....91
5.2.6 Análisis de Interfaces de Usuario....104
5.2.7 Especificación del plan de pruebas....107

CAPÍTULO 6. DISEÑO....112

6.1 ARQUITECTURA DEL SISTEMA....112
6.1.1 Diagrama de paquetes....112
6.1.2 Diagrama de componentes....114
6.1.3 Diagrama de despliegue....116
6.2 DISEÑO DE CLASES....118
6.2.1 Diagramas de Clases....118
6.3 DIAGRAMAS DE INTERACCIÓN Y ESTADOS....133

6.3.1 Caso de uso: Iniciar sesión *134*
6.3.2 Añadir objeto a base de datos: Añadir usuario *136*
6.3.3 Añadir objeto al servidor ISP: Dominio *138*
6.4 Diseño de la Base de Datos 139
6.4.1 Descripción del SGBD Usado *139*
6.4.2 Integración del SGBD en Nuestro Sistema *139*
6.4.3 Diagrama de la base de datos *139*
6.5 Diseño de la Interfaz 145
6.5.1 Intranet *146*
6.5.2 Módulo de gestión de usuarios *147*
6.5.3 Módulo de gestión de usuarios *149*
6.6 Especificación Técnica del Plan de Pruebas 152
6.6.1 Pruebas Unitarias *152*
6.6.2 Pruebas de Integración y del Sistema *152*
6.6.3 Pruebas de Usabilidad y Accesibilidad *152*
6.6.4 Pruebas de Rendimiento *153*

CAPÍTULO 7. IMPLEMENTACIÓN DEL SISTEMA 155

7.1 Estándares y Normas Seguidos 155
7.1.1 XTML 1.1 ..... *155*
7.1.2 CSS 2 ..... *155*
7.1.3 Estándares de programación *156*
7.1.4 Modelo Vista Controlador *156*
7.1.5 Patrón fachada *156*
7.1.6 Objeto de Acceso a Datos (DAO) *156*
7.1.7 Método de Factorías *157*
7.2 Lenguajes de Programación 158
7.2.1 Java *158*
7.2.2 JavaScript *160*
7.2.3 PHP *161*
7.3 Herramientas y Programas Usados para el Desarrollo 162
7.3.1 Eclipse Juno *162*
7.3.2 SQL Server Management Studio Express *162*
7.3.3 Enterprise Architect *162*
7.3.4 Microsoft Office Project *163*
7.4 Creación del Sistema 164
7.4.1 Problemas Encontrados *164*
7.4.2 Descripción Detallada de las Clases *166*

CAPÍTULO 8. DESARROLLO DE LAS PRUEBAS 218

8.1 Pruebas Unitarias 218
8.1.1 Insertar usuario *218*
8.1.2 Obtener usuario por identificador *218*
8.1.3 Obtener todos los usuarios *218*
8.1.4 Obtener usuario por criterio *218*
8.1.5 Modificar usuario *219*
8.1.6 Eliminar usuario *219*
8.1.7 Obtener cliente por identificador *219*
8.1.8 Obtener todos los clientes *219*
8.1.9 Insertar grupo *219*
8.1.10 Obtener grupo por identificador *219*

8.1.11 Obtener todos los grupos ... 220
8.1.12 Obtener grupo por descripción ... 220
8.1.13 Modificar grupo ... 220
8.1.14 Eliminar grupo ... 220
8.1.15 Insertar perfil ... 220
8.1.16 Obtener perfil por identificador ... 220
8.1.17 Obtener todos los perfiles ... 221
8.1.18 Obtener perfil por descripción ... 221
8.1.19 Modificar perfil ... 221
8.1.20 Eliminar perfil ... 221
8.1.21 Obtener todos los módulos ... 221
8.1.22 Obtener módulo por nombre ... 222
8.1.23 Insertar dominio ... 222
8.1.24 Obtener dominio por identificador ... 222
8.1.25 Obtener todos los dominios ... 222
8.1.26 Obtener dominios por cliente ... 222
8.1.27 Activar/Desactivar Dominio ... 222
8.1.28 Eliminar dominio ... 223
8.1.29 Insertar buzón ... 223
8.1.30 Obtener buzón por identificador ... 223
8.1.31 Obtener todos los buzones ... 223
8.1.32 Insertar cuenta ... 223
8.1.33 Obtener cuenta por identificador ... 223
8.1.34 Obtener cuenta por nombre ... 224
8.1.35 Aplicar una autorespuesta ... 224
8.1.36 Insertar clienteISP ... 224
8.1.37 Obtener clienteISP por identificador ... 224
8.1.38 Obtener todos los clientes ... 224
8.1.39 Eliminar cliente ... 225
8.1.40 Obtener servidor por identificador ... 225
8.1.41 Obtener todos los servidores ... 225
8.2 PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA ... 226
8.2.1 Intranet ... 226
8.2.2 Gestión de usuarios ... 227
8.2.3 Gestión de dominios ... 231
8.3 PRUEBAS DE RENDIMIENTO ... 236
8.4 PRUEBAS DE ACCESIBILIDAD ... 236

CAPÍTULO 9. MANUALES DEL SISTEMA ... 241

9.1 MANUAL DE INSTALACIÓN ... 241
9.1.1 Servidor de base de datos ... 241
9.1.2 Sistema de gestión de base de datos ... 243
9.1.3 Base de datos ... 244
9.1.4 Servidor Web ... 260
9.1.5 Instalación del servidor ISPConfig ... 262
9.2 MANUAL DE EJECUCIÓN ... 263
9.2.1 Inicio del servidor web ... 263
9.2.2 Desplegar la aplicación ... 263
9.2.3 Parada del servidor web ... 263
9.3 MANUAL DE USUARIO ... 264
9.3.1 Inicio de sesión en la aplicación ... 264

9.3.2 Opciones de usuario 264
9.3.3 Opciones de cliente 265
9.3.4 Manejo de usuarios 266
9.3.5 Manejo de clientes 268
9.3.6 Manejo de buzones 269
9.4 MANUAL DEL PROGRAMADOR 272

CAPÍTULO 10. CONCLUSIONES Y AMPLIACIONES 277

10.1 CONCLUSIONES 277
10.2 AMPLIACIONES 278
10.2.1 Usabilidad y accesibilidad 278
10.2.2 Utilización de otra tecnología más acorde 278
10.2.3 Mayor soporte al servidor ISPConfig 278
10.2.4 Aplicación para Android 279

CAPÍTULO 11. PRESUPUESTO 281

11.1 PRESUPUESTO DE LA EMPRESA 281
11.2 PRESUPUESTO DESARROLLADO POR EL ALUMNO 282

CAPÍTULO 12. REFERENCIAS BIBLIOGRÁFICAS 283

12.1 REFERENCIAS EN INTERNET 283
12.2 TUTORIALES Y GUÍAS VARIAS 283
12.3 FOROS DE AYUDA 284

CAPÍTULO 13. APÉNDICES 285

13.1 CONTENIDO ENTREGADO EN EL CD-ROM 285

Índice de Figuras

Estado del arte 2.1: QMail admin **33**
Estado del arte 2.2: ISPConfig **34**
Aspectos teóricos 3.1 : Módelo de funcionamiento DNS **35**
Aspectos teóricos 3.2: Utilización del S.O Android **45**
Aspectos teóricos 3.3: Fases de la metodología SCRUM **46**
Diagrama 4.1: Gantt correspondiente a la planificación inicial **48**
Diagrama 4.2: Gantt correspondiente a la planificación final **49**
Diagrama 5.1 : Casos de uso-Usuario resgistrado **66**
Diagrama 5.2: Robustez-Iniciar sesión **66**
Diagrama 5.3: Robustez-Cerrar sesión **67**
Diagrama 5.4 : Casos de uso-Usuario con permisos en el módulo de usuarios **69**
Diagrama 5.5: Robustez-Crear usuario **70**
Diagrama 5.6: Robustez-Editar usuario **71**
Diagrama 5.7: Robustez-Ver usuarios **72**
Diagrama 5.8: Robustez-Eliminar usuario **73**
Diagrama 5.9: Casos de uso-Usuario con permisos en el módulo de dominios **79**
Diagrama 5.10: Robustez-Crear cliente **79**
Diagrama 5.11: Robustez-Editar cliente **81**
Diagrama 5.12: robustez-Ver clientes **82**
Diagrama 5.13: Casos de uso-Cliente **91**
Diagrama 5.14: Clases-Intranet **93**
Diagrama 5.15: Clases-Módulo de usuarios **99**
Diagrama 5.16: Clases-Módulo gestión de dominios **100**
Interfaz 5.1: Descripción de la interfaz **104**
Diagrama 6.1: Paquetes **112**
Diagrama 6.2: Componentes **114**
Diagrama 6.3: Despliegue **116**
Diagrama 6.4: Clases-Modelo intranet **118**
Diagrama 6.5: Clases-Persistencia intranet **119**
Diagrama 6.6: Clases-Negocio intranet **120**
Diagrama 6.7: Clases-Presentación intranet **120**
Diagrama 6.8: Clases-Seguridad intranet **121**
Diagrama 6.9: Clases-Persistencia usuarios **122**
Diagrama 6.10: Clases-Negocio usuarios **123**
Diagrama 6.11: Clases-Presentación usuarios (grupos) **124**
Diagrama 6.12: Clases-Presentación usuarios (perfiles) **125**
Diagrama 6.13: Clases-Presentación usuarios (usuarios) **126**
Diagrama 6.14: Clases-Modelo dominios **127**
Diagrama 6.15: Clases-Persistencia dominios **128**
Diagrama 6.16: Clases-Negocio dominios **129**
Diagrama 6.17: Clases-Presentación dominios (buzones) **130**
Diagrama 6.18: Clases-Presentación dominios (clientes) **131**
Diagrama 6.19: Clases-Presentación dominios (dominios) **132**
Diagrama 6.20: Interacción-Iniciar sesión **134**
Diagrama 6.21: Estado-Usuario **135**
Diagrama 6.22: Interacción-Insertar usuario **136**

Diagrama 6.23: Estado-Conexión **137**
Diagrama 6.24: Interacción-Insertar dominio **138**
Diagrama 6.25: Entidad-Relación **140**
Diagrama 6.26: Entidad-Relación (ISPConfig) **144**
Interfaz 6.1: Página personal **146**
Interfaz 6.2: Listado de usuarios **147**
Interfaz 6.3: añadir usuario **148**
Interfaz 6.4: Listado dominios **149**
Interfaz 6.5: Ver cliente **150**
Interfaz 6.6: Configurar autorespuesta **151**
Manual instalación 9.1: SQLServer-Información de registro **241**
Manual instalación 9.2: SQLServer-Componentes **242**
Manual instalación 9.3: SQLServer-Finalización **243**
Manual instalación 9.4: SQLServer management studio **244**
Manual instalación 9.5: Conexión base de datos **244**
Manual instalación 9.6: Creación de certificado autofirmado **261**
Manual usuario 9.1: Iniciar sesión **264**
Manual usuario 9.2: Opciones de usuario **264**
Manual usuario 9.3: Cambio de contraseña **265**
Manual usuario 9.4: Opciones del cliente **265**
Manual usuario 9.5: Consultar dominios propios **266**
Manual usuario 9.6: Visualización de usuarios **266**
Manual usuario 9.7: Añadir nuevo usuario **267**
Manual usuario 9.8: Modificar usuario **267**
Manual usuario 9.9: Eliminar usuario **268**
Manual usuario 9.10: Insertar cliente **269**
Manual usuario 9.11: Añadir buzón **269**
Manual usuario 9.12: Cuentas de correo **270**
Manual usuario 9.13: Activar autorespuesta **270**

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La empresa Ecocomputer, para la cual se realiza este proyecto, ofrece servicios de contratación y gestión de dominios a sus clientes, por ello, surge la necesidad de una herramienta que automatice dichos procesos en lo máximo posible y que proporcione a los clientes posibilidades de administración extra que antes no tenían. Por todo esto, surge el proyecto de un gestor de dominios para la empresa, el cual permita a grandes rasgos, asociar dominios con los clientes y generar de manera automática facturas y avisos relativos a estos. Por otro lado, se permite en ella que los clientes puedan administrar y gestionar sus cuentas de correo dentro del dominio.

Se pretende pues, con todo esto, que el proyecto sirva de ayuda dentro de la empresa y que sea puesto en explotación por la misma. Consiguiendo con ello haber puesto en producción en un sistema real el proyecto.

Para todo lo anterior, es necesaria a su vez una intranet básica que permita gestionar los usuarios y clientes de la empresa, de la cual no se disponía. Por ello, como parte del proyecto y en colaboración con otros compañeros, se desarrollará una infraestructura básica, en mi caso, todo lo relacionado a la gestión de usuarios y permisos de acceso para los distintos módulos que se irán acoplando.

1.2 Resumen de Todos los Aspectos

Este documento contiene los siguientes capítulos:

1.2.1 Capítulo 1: Memoria del Proyecto

En esta sección se resume en general el contenido del proyecto, para quienes no posean un conocimiento avanzado y técnico de las diferentes tecnologías empleadas. El público objetivo es cualquier persona que quiera saber a grandes rasgos y sin profundizar demasiado lo que se pretende conseguir con la realización de este proyecto.

1.2.2 Capítulo 2: Introducción

Aquí se explica el propósito por el que se ha desarrollado el proyecto. Se detallan las necesidades que se pretenden satisfacer, así como qué se pretende conseguir con ello.

1.2.3 Capítulo 3: Aspectos teóricos

En este apartado describiremos los conceptos relevantes y necesarios para el correcto desarrollo del proyecto. Se trata de una pequeña descripción de las diferentes tecnologías o herramientas empleadas durante todo el proceso de creación.

1.2.4 Capítulo 4: Planificación del Proyecto

Este capítulo contiene la planificación temporal del proyecto: el tiempo que ha llevado desarrollarlo desde la concepción de la idea hasta el último aspecto antes de la entrega definitiva.

1.2.5 Capítulo 5: Análisis

Esta sección abarca todo lo concerniente a los límites del desarrollo de la aplicación. Todas las funcionalidades que ofrecerá son aquí desglosadas, estudiando los diferentes elementos necesarios para el correcto cumplimiento de los requisitos listados. Finalmente, el lector podrá encontrar un primer boceto de la interfaz de usuario y las diferentes pruebas que la aplicación deberá superar para cumplir con lo requerido.

1.2.6 Capítulo 6: Diseño del Sistema

Se explica la arquitectura de la aplicación desarrollada con la ayuda de diferentes diagramas que faciliten su entendimiento. También se incluye el resultado final de la interfaz de usuario y se define cómo se harán las diferentes pruebas.

1.2.7 Capítulo 7: Implementación del Sistema

En este capítulo el lector podrá conocer los distintos lenguajes de programación empleados para construir la aplicación, así como normas o patrones de desarrollo y los programas externos que han sido empleados para ello. También se desglosan los diferentes problemas surgidos durante el desarrollo del proyecto, además de una descripción detallada de las clases del código fuente del mismo.

1.2.8 Capítulo 8: Desarrollo de Pruebas

Aquí podremos encontrar los resultados de las pruebas realizadas a la aplicación y las conclusiones obtenidas tras su estudio.

1.2.9 Capítulo 9: Manuales del Sistema

Bajo este apartado se agrupan los diferentes manuales o guías que se necesitan para instalar la aplicación en el entorno de explotación. También se encuentra el manual de usuario que

explica cómo se debe usar la aplicación, además de una guía para otros desarrolladores que pueda ayudarles a ampliar, modificar o entender diferentes conceptos del proyecto.

1.2.10 Capítulo 10: Conclusiones y ampliaciones

En esta sección se localiza la opinión personal sobre lo que ha supuesto la creación de este proyecto para su desarrollador, así como un desglose de las diferentes posibles mejoras y ampliaciones a las que la aplicación podría ser sometida en un futuro.

1.2.11 Capítulo 11: Presupuesto

Aquí podrá encontrarse el presupuesto detallado de lo que este proyecto podría haber costado financieramente.

1.2.12 Capítulo 12: Referencias Bibliográficas

Bajo este capítulo se muestra una lista de toda la documentación consultada para la realización del proyecto, ya sean libros divulgativos, explicativos o enlaces de Internet con dicha información.

1.2.13 Capítulo 13: Apéndice

Se detalla el contenido del CD entregado y las tareas que deben realizarse para su correcta instalación, ejecución y configuración.

Capítulo 2. Introducción

2.1 Justificación del Proyecto

Dentro de la empresa Ecocomputer, surge la necesidad de automatizar la gestión de los dominios que ofrecen a los clientes. Por esto, se pide el desarrollo de una aplicación que sea capaz de vincular los dominios con sus respectivos clientes. Esta aplicación deberá constar de una interfaz web, la cual será utilizada por los trabajadores de la empresa a modo de administrador general y por los clientes, a modo de consultor de información de sus dominios y administrador de las cuentas de correo que estos tengan asociadas.

De igual manera, la aplicación automatizará varios puntos de esta gestión, entre ellos la generación automática de facturas en función a la ley vigente, la generación de avisos de caducidad del dominio y la posibilidad de renovar los contratos. Por otro lado, de cara al cliente deberá permitirle gestionar sus cuentas de correo, reenvíos, altas nuevas, etc. Y consultar sus contratos con la empresa.

De forma análoga, se generará una pequeña aplicación Android, que permita a los clientes gestionar los reenvíos de sus cuentas de correo y consultarlos y a los técnicos de la empresa consultar los datos de los dominios que se encuentren registrados en el sistema, así como las incidencias asociadas a dichos dominios.

Como ya se dijo en un punto anterior, este proyecto se justifica por sí solo, ya que se trata de un proyecto real, que será puesto en explotación dentro de la empresa y con el cual trabajarán.

2.2 Objetivos del Proyecto

Los objetivos que se persiguen con la realización del proyecto pueden dividirse en dos apartados:

2.2.1 Creación de la infraestructura necesaria para la intranet

Estos objetivos están enfocados a la creación de la intranet de la empresa, abarcando con ella las funciones más básicas utilizadas por todos los módulos que serán implementados e integrados posteriormente sobre ella, de manera que el código sea lo menos diverso posible, facilitando su posterior mantenimiento.

1. Creación de una base de datos común para todos los módulos que surjan por encima.
2. Será desarrollada íntegramente sobre un entorno Web con gestión de permisos de usuario y niveles de acceso.
3. Constará de un interfaz web, que será accesible desde cualquier dispositivo.
4. Centralización de acceso a todas las herramientas que se incluyan sobre la intranet.
5. Gestión completa de usuarios, permisos, perfiles y grupos para su posterior utilización dentro de los módulos que sean incluidos.
6. Especificación de unas pautas de programación y utilización de los distintos servicios que ofrece la base desarrollada.
7. Especificación de una interfaz genérica que deba ser acogida por todos los módulos que se añadan a la intranet.

2.2.2 Módulo de gestión de dominios

Los objetivos que se recogen en esta sección se aplican al módulo de gestión de dominios, el cual es el objetivo principal del proyecto.

1. Será desarrollada íntegramente sobre un entorno Web con gestión de permisos de usuario y niveles de acceso.
2. La aplicación constará de un interfaz web, que será accesible desde cualquier dispositivo.
3. Dicha interfaz, será tan intuitiva como sea posible y deberá resultar agradable de cara al usuario.
4. Se integrará con la intranet de la empresa, siendo accesible desde ella y conllevando un estilo de codificación lo más similar posible.
5. La aplicación deberá permitir el registro de varios dominios para un mismo cliente y gestionar ambos en conjunto o de forma separada, según preferencia del mismo.
6. Permitirá consultar por parte de los trabajadores de la empresa, datos relativos a los dominios y los clientes que estén registrados en la base de datos.
7. Los clientes tendrán cierto control sobre sus cuentas de correo asociadas, permitiéndoles especificar auto respuestas y crear nuevas cuentas para sus dominios.

8. Finalmente, el proyecto constará de un pequeño añadido en forma de aplicación Android, la cual deberá permitir consultar datos de clientes e incidencias, en el caso de trabajadores de la empresa y gestionar las cuentas de correo, en el caso de clientes de la misma.

2.3 Estudio de la Situación Actual

Inicialmente en la empresa, el tema relativo a la gestión de dominios está encargado por completo a una persona, la cual tiene que utilizar varias herramientas para poder llevar a cabo todas las acciones necesarias para gestionar y mantener el registro del dominio. Aunque en la actualidad existen una infinidad de herramientas que proporcionan la posibilidad de gestionar todo lo relativo a dominios web, lo que se pretende con esta aplicación es ir un poco más allá y generar un proyecto que no solo permita gestionar información relativa a los dominios y a los clientes que tenga registrados, si no también, que se adapte a la base de datos existente en la empresa, utilizando en todo momento información específica que la empresa ha requerido. Además, ofrecerá un sistema personalizado para que interiormente se puedan generar incidencias de cara a dichos dominios, permitiendo un control mucho mayor de todos los aspectos relativos a las mismas. Además, la posibilidad de adaptar dicho sistema para que funcione con los módulos indicados, como QMailAdmin, ofreciendo a su vez una gestión de buzones de correo interna.

En resumen, está aplicación se trata de un software hecho a medida, con el cual se pretende cubrir todas las necesidades de la empresa en el campo de la gestión y mantenimiento de los dominios que tengan registrados, facilitando así la labor del administrador, pues se le ofrecen todas las funcionalidades necesarias dentro de la misma herramienta.

2.3.1 Evaluación de Alternativas

2.3.1.1 *QMailAdmin*

QMailAdmin es la primera alternativa a evaluar, pues es el sistema de gestión de correos electrónicos que está utilizando la empresa en este momento.

Se trata de una interfaz web para gestionar sistemas QMail con dominios virtuales. Existe una versión que puede ser usada con vpopmail y permite añadir, eliminar usuarios, especificar alias, reenvíos, listas de correo y autorespuestas.

El único inconveniente que esto le plantea a la empresa es que no se integra con ninguna otra herramienta y que el diseño es bastante sencillo, produciendo una mala imagen cuando es necesario darle acceso a los clientes al mismo. Por esto se plantea cambiarlo por una interfaz para QMail integrada con el resto de la intranet de la empresa sobre la que se tenga posibilidad de edición.

Estado del arte 2.1: QMail admin

2.3.1.2 *ISPConfig*

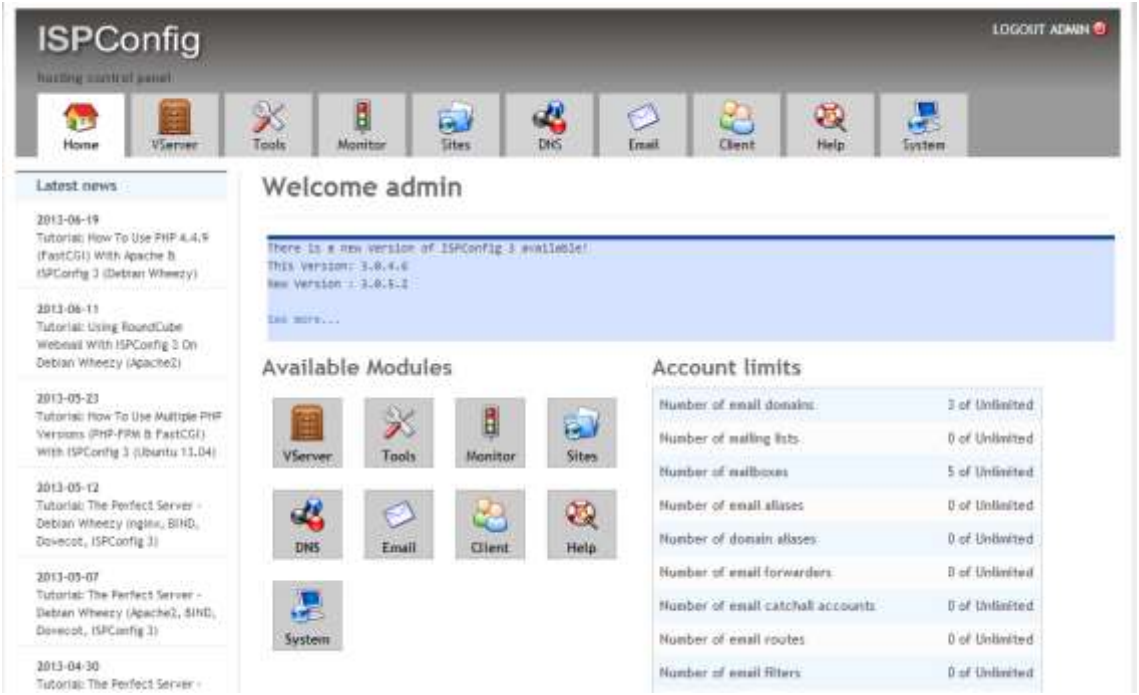
ISPConfig es un software libre de control para Linux capaz de gestionar múltiples servidores desde un único panel de control.

Este software se acerca y mejora el módulo de gestión de dominios que se planteaba en este proyecto pues proporciona la capacidad de registrar y configurar dominios, buzones, cuentas de correo. En resumen, todo lo que se pretende con la parte relacionada de este proyecto.

Además proporciona módulos externos que permiten realizar la facturación y sustituir la interfaz de usuario para los correos.

En última instancia proporciona una capa de servicios web SOAP para ser consumidos por otras aplicaciones.

Dados todos los puntos resumidos anteriormente se planteará a la empresa la utilización de este software para realizar la gestión de dominios, lo que en último punto reducirá el alcance del proyecto, pero en lo cual no tiene sentido invertir horas de desarrollo para conseguir un resultado posiblemente peor. No se pretende reinventar la rueda.



Estado del arte 2.2: ISPConfig

Como vemos la interfaz ofrece gestión de muchos tipos de objetos relacionados con los dominios antes mencionados y monitorización de algunos de ellos.

Capítulo 3. Aspectos Teóricos

3.1 Dominios de Internet

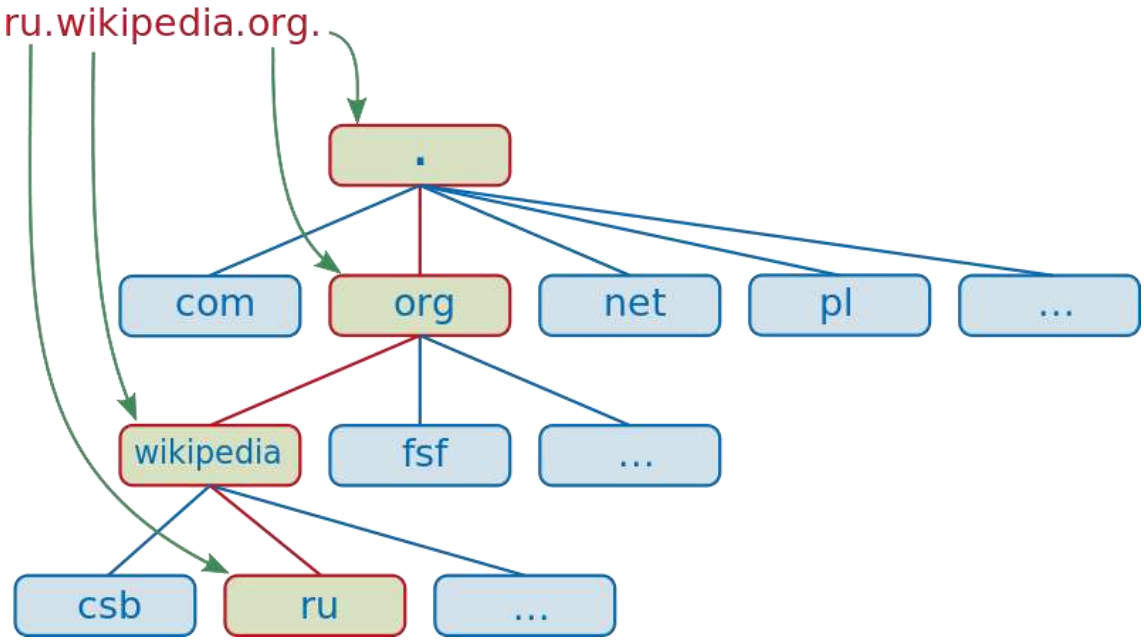
Los dominios de Internet son cadenas utilizadas para la identificación de terminales dentro de Internet. Están siempre formados por las reglas y procedimientos especificados en el estándar DNS (Domain Name System).

Estos nombres se utilizan como referencia recordable para una dirección IP a la cual están vinculados. El registro de un dominio por parte de una persona le otorga el derecho de uso exclusivo del mismo durante la duración del contrato.

3.1.1 Sintaxis

Los nombres de dominio están conformados por etiquetas, separadas mediante puntos. La etiqueta que aparece a la derecha del todo indica el dominio de nivel más alto.

El árbol de etiquetas de un dominio puede consistir de 127 niveles, casa uno con un máximo de 63 octetos no excediendo el nombre del dominio los 255 caracteres.



Aspectos teóricos 3.1 : Módelo de funcionamiento DNS

3.1.2 Dominios de nivel más alto

Los niveles de dominio más alto o TLD (Top-Level Domains) son los niveles que se encuentran registrados en la zona DNS más alta, de manera que son los primeros sobre los que se consultan las direcciones de nombres.

La división más básica de estos niveles consta de dos partes, por países, que se basan en dos caracteres identificativos de los mismos (es, uk, fr…). Y siete dominios generales que se utilizan para la representación de un conjunto de categorías y nombres de organizaciones.

3.1.3 Registro de dominios

El derecho de utilización de un dominio web debe ser delegado por una registradora de dominios web, la que a su vez debe estar acreditada por la ICANN (Internet Corporation for Assigned Names and Numbers). Cuando se registra un nuevo dominio, este es automáticamente actualizado en los servidores de nivel más alto, referenciando la dirección IP a la que ha sido vinculado.

Normalmente, el registro de un dominio conlleva un pago anual por el servicio derivado de la delegación del nombre.

3.1.3.1 *Requisitos técnicos*

Para la realización del registro de un dominio, es necesario cumplir con una serie de requisitos, los cuales se detallan a continuación.

- Contacto administrativo: Normalmente, son los encargados de administrar un nombre de dominio. Tiene el máximo control sobre el mismo y es el encargado de manejar toda la información relativa al negocio y comprobar que se cumplen todos los requerimientos relativos al registro de dominios.
- Contacto técnico: Son los encargados de los servidores que son la ruta final de un nombre de servicio. Deben proporcionar un servicio de mantenimiento que implique que el servidor esté funcionando de manera continua y sea seguro.
- Contacto de facturación: Son los encargados de llevar todo el tema relativo a las facturas y pagos asociados a un dominio.

3.2 Servlet

3.2.1 Definición e historia

Un Servlet es una clase de programación Java, utilizada para extender el funcionamiento de un servidor, al contrario que JavaScript, estos funcionan en modo "server-side", lo que permite un mayor control de los datos que manejan.

De manera resumida, un Servlet es un objeto que recibe una petición y genera una respuesta basada en la misma. Los paquetes básicos de Java dotan a los Servlets de objetos que representan las peticiones y las respuestas, así como otros objetos con información sobre la configuración y el entorno de ejecución de los mismos. En el caso de los Servlets HTTP, se incluyen objetos manejadores de sesión que permiten la ejecución de peticiones y respuestas entre el cliente y el servidor.

La especificación de Servlets fue creada en 1997 por Sun MicroSystems y fue desarrollada bajo la Comunidad de Procesos de Java. Actualmente, se encuentran en la versión 3.0.

3.2.2 Características

La utilización de Servlets, ofrece una serie de ventajas con respecto a otras tecnologías similares, las cuales serán detalladas a continuación.

- Cada petición generada, es manejada por un hilo Java separado de los procesos del servidor.
- Al contrario que con CGI, los Servlets no se duplican en memoria, si no que al ejecutarse en hilos solo transmiten su información de funcionamiento.
- Una sola instancia se encarga del manejo de todas las peticiones del sistema, facilitando el manejo de la persistencia y reduciendo la utilización de memoria.

3.2.3 Aplicación en el proyecto

Al tratarse de una aplicación que estará alojada en un servidor y sobre la cual se accederá mediante una interfaz Web, la utilización de Servlets nos permite realizar la interacción entre ambas partes utilizando la tecnología Java. Lo cual debido a la formación de la que se consta resulta muy útil.

Se utilizarán estos Servlets como ya se ha dicho para la comunicación entre la interfaz web y la aplicación desarrollada, sin embargo, para facilitar el trabajo con ellos se utilizarán tecnologías como Struts y Spring, que ofrecen implementaciones robustas de los mismos.

3.3 JavaScript

3.3.1 Definición e historia

El lenguaje JavaScript aparece en 1995 como una evolución del lenguaje hasta entonces conocido como LiveScript. Debido a sus características se comenzó a usar como lenguaje de interpretación de ordenes junto HTML en modo "client-side", al contrario que PHP que se utilizaba en modo "server-side" siendo así su complementario, en 1997 fue adoptado como un estándar ECMA, pasando a tomar el nombre de ECMAScript y posteriormente convirtiéndose en un estándar ISO.

3.3.2 Características

El lenguaje JavaScript es un lenguaje de programación interpretado que posee características tales como:

- Orientación a objetos
- Está basado en prototipos
- Es imperativo
- Es débilmente tipificado
- Es dinámico

Posee además, una gran portabilidad, pudiendo ser interpretado por la mayoría de los navegadores web. Esto es debido a que se suele usar en conjunto con HTML para conseguir aplicaciones web dinámicas, que puedan hacer uso de los recursos del ordenador cliente para ejecutar funciones establecidas por un servidor.

3.3.3 Aplicación en el proyecto

Debido a la necesidad de la aplicación de tener un interfaz Web, JavaScript se vuelve una tecnología casi fundamental para su desarrollo, permitiendo un gran dinamismo dentro de la misma. Sin embargo, por temas de seguridad, se utilizará principalmente para alterar la interfaz, de manera que pueda servir como avisos para el cliente y de manera visual. Ya que toda la comunicación con el servidor se realizará mediante Servlets.

3.4 SQL

3.4.1 Definición e historia

El lenguaje SQL (Lenguaje de consulta estructurado) es un lenguaje de programación con un propósito específico, diseñado para el manejo de datos en sistemas de bases de datos relacionales. Sus funcionalidades más utilizadas incluyen la inserción, consulta, modificación y eliminación de datos.

SQL fue uno de los primeros sistemas comerciales creados para trabajar con el modelo relacional de datos, descrito por Edgar F. Codd en 1970. Se convirtió en un estándar de la ANSI en 1986 y de ISO en 1987. Aunque las versiones actuales han sido modificadas varias veces desde entonces, añadiendo funcionalidades nuevas.

3.4.2 Características

El lenguaje SQL está subdividido en varios elementos del lenguaje, entre los que se incluyen:

- **Clausulas:** Son condiciones de modificación que indican los datos que se quieren manipular.
- **Expresiones:** Pueden ser o producir datos que serán almacenados en la base de datos.
- **Predicados:** Especifican condiciones que pueden ser evaluadas por SQL.
- **Consultas:** Devuelven los datos basándose en un criterio específico.
- **Declaraciones:** Instrucciones que producen efectos y modificaciones en los datos o el esquema de los mismos y controlan transiciones del programa.

3.4.3 Aplicación en el proyecto

Al tratarse de un proyecto con una base de datos relacional, es necesaria la utilización de un lenguaje de programación que nos proporcione el acceso a los datos y la posibilidad de alterarlos, de manera que queden reflejados para futuras consultas. Dado que SQL es el lenguaje más extendido para este tipo de operaciones será el que se utilice en el proyecto.

3.5 SQL Server

3.5.1 Definición e historia

Microsoft SQL Server es un sistema para la gestión de bases de datos producido por Microsoft, basado en el modelo relacional. Como base de datos, está basado en la funcionalidad de guardar y recuperar datos pedidos por otras aplicaciones. Los lenguajes de consulta principales para esta aplicación son T-SQL y ANSI-SQL.

La primera versión de Microsoft SQL Server fue lanzada en 1989, bajo el nombre de SQL Server 1.0. Desde entonces ha sufrido varias reediciones, pasándose a llamar SQL Server 2000 en el año cuyo nombre indica y hasta la versión actual SQL Server 2012.

3.5.2 Características

La versión que se utilizará para la aplicación por petición de la empresa será SQL Server Express 2005, una versión de gama baja del software de Microsoft, que ofrece solo limitaciones Hardware, solo pudiendo usar un procesador, 1 GB de memoria y 4GB de archivos de bases de datos.

3.5.3 Aplicación en el proyecto

SQL Server se utilizará en el proyecto para la creación de la base de datos y como fuente para realizar las consultas necesarias desde la capa de persistencia.

3.6 JDBC

3.6.1 Definición e historia

JDBC (Java Database Connectivity), es una tecnología de acceso a base de datos basada en Java y desarrollada por Sun Microsystems. Define como un cliente debe realizar el acceso a la base de datos, proveyendo métodos para consultar y actualizar dichas bases de datos.

Fue lanzado por primera vez en 1997 y desde entonces forma parte de la Edición Estándar de Java. Desde su versión 3.1, fue desarrollada por la Java Community Process, siendo su última versión la 4.1 y estando incluida junto con Java 7.

3.6.2 Características

JDBC da soporte a conexiones que permitan ejecutar sentencias, tanto de actualización como de consulta. De manera adicional, permite la invocación de procedimientos almacenados.

En el caso de operaciones de actualización, JDBC retorna el número de filas que han sido modificadas por la sentencia, por otro lado, en el caso de consultas, retorna un "Result set" o conjunto de resultados, el cual se utilizará para iterar sobre el mismo y obtener los datos necesarios.

3.6.3 Aplicación en el proyecto

Al ser un proyecto basado en Java y con una base de datos proporcionada por SQL Server, se hace necesaria la aplicación de un mecanismo de conexión entre ambos. JDBC ofrece bibliotecas para realizar dicha conexión con bases SQL Server y además permite que sea relativamente sencilla la inclusión de una base de datos distinta en un futuro, combinando esto con el desarrollo de la aplicación en módulos independientes, conseguimos que la base de datos tenga un nivel de acoplamiento muy débil con el resto de la aplicación, lo que proporciona varias ventajas añadidas al proyecto.

3.7 Apache Tomcat

3.7.1 Definición e historia

Apache Tomcat es un servidor web de código abierto que también actúa como un contenedor de Servlets, desarrollado por la Fundación de Software de Apache. Comenzó como una implementación de referencia para Servlets, creada por James Duncan Davidson, un arquitecto de software en Sun Microsystems y posteriormente se donó como código abierto en 1999 a la Fundación de Software de Apache. Desde esta primera versión (3.0.x) se han realizado y relanzado varias revisiones del mismo, hasta la última, 7.0.34.

3.7.2 Características

Tomcat ofrece implementación para los Servlet de Java y para JSP (Java Servlet Page), cumpliendo la especificación de Oracle y ofreciendo un servidor web "puro" para desplegar aplicaciones Java.

Se divide en cuatro componentes principales, cada uno con una funcionalidad propia:

- **Catalina:** Es un contenedor de Servlets que implementa las especificaciones dadas por Sun Microsystems.
- **Coyote:** Funciona como conector de HTTP, dando soporte a la versión 1.1 del protocolo. Es el componente encargado de escuchar por peticiones TCP y transmitírselas al núcleo de Tomcat para procesarlas y devolver la respuesta indicada al cliente.
- **Jasper:** Es un convertidor de archivos JSP, compilándolos en código de Java y convirtiéndolos en Servlets, que puedan ser utilizados por Catalina. Detecta los cambios en "caliente" que se realicen sobre las JSPs.
- **Clúster:** Es un componente que se añadió para manejar aplicaciones muy grandes, se utiliza principalmente para realizar balanceos de carga.

3.7.3 Aplicación en el proyecto

Dentro del proyecto, Tomcat será utilizado como servidor de aplicaciones, puesto que permite la utilización de Servlets Java para realizar la ejecución de código remoto.

3.8 Apache Struts

3.8.1 Definición e historia

Apache Struts es un marco de trabajo para aplicaciones Web de código abierto, utiliza y extienda la API de Servlets de Java y trata de conseguir que los desarrolladores utilicen el Modelo Vista Controlador (MVC) para sus aplicaciones. Originalmente fue creado por Craig McClanahan y donado a la Fundación de Software de Apache en el año 2000.

3.8.2 Características

Una de las principales características de Struts es que permite separar por completo la lógica de negocio de la capa de presentación, además ofrece un mecanismo de conexión entre ambas por medio de "Actions" que generan respuestas para el cliente.

Entre sus características más importantes encontramos:

- Acciones simples basadas en POJO
- Capacidad de pruebas simplificada
- Manejo de hilos seguro
- Soporte para AJAX
- Soporte para plantillas
- Soporte para distintos tipos de resultados
- Facilidad para extenderlo con plug-ins

3.8.3 Aplicación en el proyecto

Dadas las características del proyecto, Struts nos viene como anillo al dedo, permitiendo separar en gran medida la lógica del negocio de la presentación y ayudándonos durante todo el desarrollo con las funcionalidades extendidas sobre los Servlets de Java.

3.9 QMail

3.9.1 Definición e historia

QMail es un agente de transferencia de correos (MTA) usado en Unix. Fue escrito en 1995 por Daniel J. Bernstein con la idea de ser un reemplazo más seguro para Sendmail.

Debido a las por entonces deficiencias que presentaba Sendmail, entre otras su falta de seguridad y las declaraciones de su autor, QMail se convirtió en un objetivo directo de críticas y ataques.

3.9.2 Características

La principal característica que ofrece QMail es la seguridad, utilizando una arquitectura modular con componentes sin confianza entre ellos, lo que supone que se necesiten diferencias credenciales para el acceso a distintos módulos. Además, la implementación sobre un reemplazo más seguro de la biblioteca estándar de C consigue que no sea vulnerable a desbordamientos de pila, ataques con cadenas formateadas, etc.

Cuando se lanzó, era significantemente más rápido que Sendmail y estaba pensado para el manejo de listas de correo exageradamente grandes. Además, resulta más sencillo de configurar que los agentes de la época.

3.9.3 Aplicación en el proyecto

QMail se utiliza como componente en el proyecto para la gestión de correos electrónicos, su uso viene dado obligatoriamente por la empresa, pero se realizará el desarrollo de manera que la aplicación sea lo menos dependiente posible de dicho agente, debido entre otras cosas a la posible decisión de la empresa de reemplazar QMail por otro servidor de correo distinto.

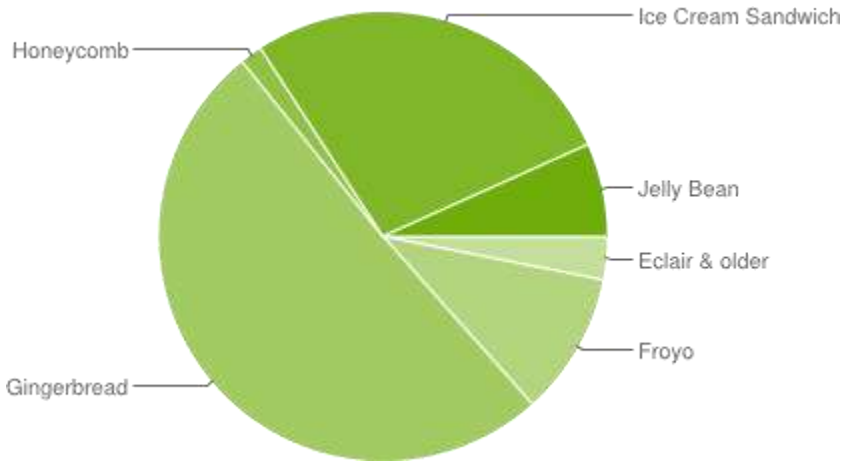
3.10 Android

3.10.1 Definición e historia

Android es un sistema operativo basado en Linux diseñado principalmente para dispositivos móviles con soporte táctil. Inicialmente fue desarrollado por Android, Inc., fundada en 2003. Que fue finalmente absorbida por Google en 2005. El primer S.O de Android fue desvelado en 2007. Google libera el código de Android bajo la licencia de Apache, lo que le convierte en una atracción para desarrolladores.

Desde su nacimiento, Android ha recibido numerosas actualizaciones, siendo la última estable la versión 4.2 Jelly Bean, lanzada en Noviembre del 2012.

A continuación se muestra un gráfico con el uso de las diferentes versiones de Android a 3 de Diciembre del 2012.



Aspectos teóricos 3.2: Utilización del S.O Android

3.10.2 Características

La principal característica de Android y que lo convierte en un punto fundamental para el desarrollo del proyecto es la posibilidad de ejecutarse en dispositivos móviles y la API que nos ofrece, ya que nos dota de la posibilidad consumir servicios web y programar con Java.

3.10.3 Aplicación en el proyecto

Dentro del proyecto, Android se utilizará para desarrollar una pequeña aplicación a parte con funcionalidad limitada, y que pueda ser utilizada desde cualquier dispositivo Android con conexión a Internet.

3.11 Scrum

3.11.1 Definición e historia

Scrum es una metodología de desarrollo de software ágil, iterativa e incremental. Utilizada en la gestión de proyectos software y al desarrollo de aplicaciones. Es una metodología enfocada a la gestión de proyectos en los que resulta difícil planificar todo desde un principio.

Surge en 1986 de la mano de Hirotaka Takeuchi, como una nueva aproximación que incrementaría la velocidad y flexibilidad de los desarrollos. La forma principal de trabajar es mediante la división del proyecto en "Sprints" que se irán afianzando en un tiempo prudencial, normalmente de dos a cinco semanas y tras los cuales se realizará una revisión del módulo desarrollado, una fase de cambios y una revisión de los requisitos para la nueva fase.

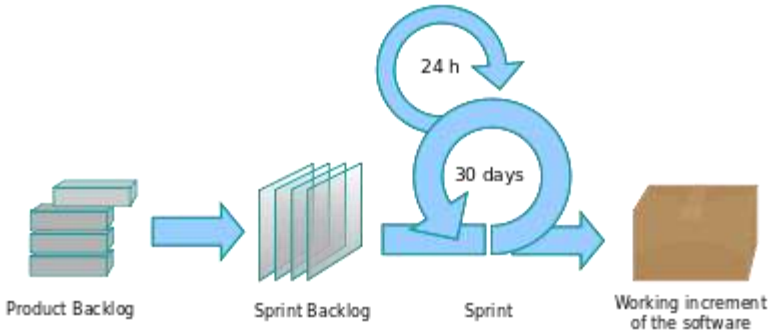
3.11.2 Características

En este modelo se contemplan tres roles diferentes:

- El maestro de la melé (ScrumMaster): Es el encargado de mantener los procesos.
- El propietario del producto o cliente.
- El equipo: Un equipo que se encarga de la realización de las distintas fases.

3.11.2.1 *FASES DEL MODELO*

1. Análisis de los requisitos.
2. Análisis de las funcionalidades cumplidas.
3. Creación de un "sprint": Diseño e implementación de una funcionalidad.
4. Fijado de la ampliación en el software final.



Aspectos teóricos 3.3: Fases de la metodología SCRUM

El equipo tiene siempre un "Product Backlog" que es un documento con todos los requisitos del cliente y en el cual se indican los que se van solventando. Este log se complementa con el "Sprint Backlog" que es un registro con todas las ampliaciones del software producidas. Sobre

este registro se eligen una serie de requisitos a cumplir con la siguiente ampliación y se trabaja sobre ello. Se evalúa si la ampliación es correcta y se añade al producto software.

3.11.3 Aplicación en el proyecto

Debido a que es un proyecto desarrollado para un cliente externo a la universidad, resulta del todo imposible tener una lista de requisitos inicial que resulte completa y sobre la que se pueda trabajar. Por ello, la metodología Scrum nos ofrece la posibilidad de ir trabajando sobre módulos del proyecto, los cuales se definirán de una forma un poco vaga al comienzo y más específica en reuniones anteriores a comenzar a trabajar sobre el mismo.

La mayor ventaja que esto nos ofrece es que la empresa puede disponer de unos requisitos flexibles hasta cierto punto y por nuestra parte que no se realizará trabajo que sea desestimado o quede fuera de la idea final de la empresa. Se intentará trabajar sobre una bolsa de horas fijas, dentro de la cual la empresa podrá pedir todos los cambios que considere oportunos para satisfaces sus requisitos, siempre y cuando las horas de dicha bolsa no varíen.

Capítulo 4. Planificación del Proyecto

4.1 Planificación

4.1.1 Diagramas de Gantt

El proyecto deberá desarrollarse en un total de 550 horas, con un plazo fijo entre el mes de enero de 2013 y Julio de ese mismo año. En un principio se decidió dividir el ciclo de desarrollo de dicho proyecto en un total de 9 sprints correspondientes, teniendo 7 de estos una duración fija de 12 días laborales cada uno, más uno inicial de solo 6 días y otro final de tiempo variable hasta la culminación del proyecto, teniendo en cuenta que la carga de trabajo diaria ascendiese a cinco horas de lunes a sábado. Tal y como se puede ver en el siguiente diagrama de Gantt.

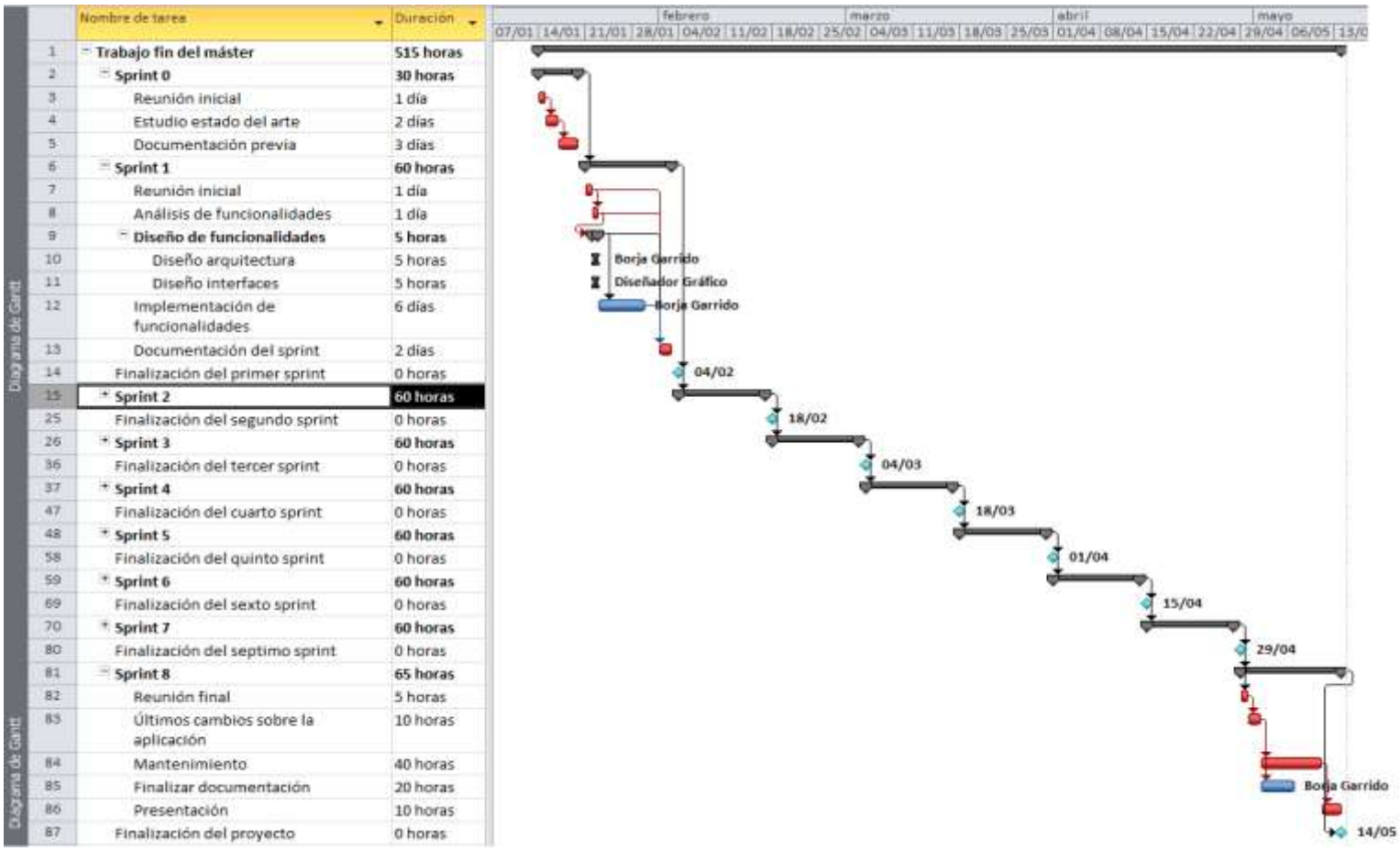


Diagrama 4.1: Gantt correspondiente a la planificación inicial

Sin embargo, la planificación expuesta arriba es totalmente idílica y debido a los problemas surgidos durante la realización de un proyecto, la planificación final del mismo se vio turbada pasando a convertirse en algo más similar a lo ofrecido en el siguiente diagrama de Gantt. En el cual se recoge la planificación real.

En este caso se puede observar como el proyecto comienza con una planificación similar, cumpliendo el sprint 0 y de una u otra manera el sprint 1, sin embargo en febrero debido al comienzo de las prácticas de empresa y posteriormente a la aparición de un trabajo a tiempo parcial, el calendario cambia convirtiéndose a una jornada laboral de 17:00 a 20:00 de lunes a viernes, de 9:00 a 14:00 los sábados y de 10:00 a 14:00 los domingos, es decir aproximadamente 25 horas semanales, viendo que el primer sprint de 60 horas (correspondientes a dos semanas) se quedó algo corto y a la imposibilidad de reunirse con los clientes cada dos semanas se decidió alargar los sprints a 3 semanas o sobre 70-75 horas cada sprint. Cubriendo en total 450 horas de proyecto sin contar el mantenimiento.

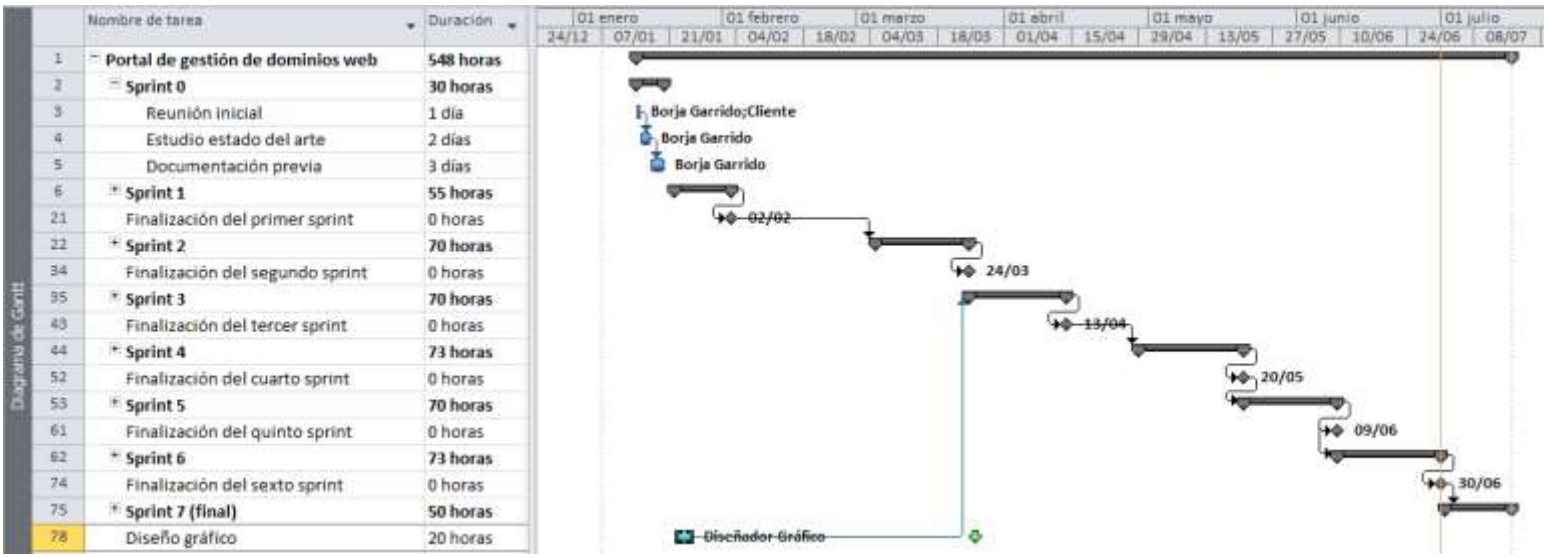


Diagrama 4.2: Gantt correspondiente a la planificación final

4.1.2 Planificación final extendida

A continuación se ofrece una tabla con las tareas extendidas para poder contemplar mejor su planificación.

Cabe destacar que la utilización de un repositorio de software se amolda perfectamente a esta planificación pues combinado con una metodología ágil permite tener todos los sprints que se van desarrollando de manera separada.

Número de esquema	Nombre de tarea	Duración	Comienzo	Fin
1	**Portal de gestión de dominios web**	**548 horas**	**14/01/13**	**15/07/13**
1.1	**Sprint 0**	**30 horas**	**14/01/13**	**19/01/13**
1.1.1	Reunión inicial	5 horas	14/01/13	14/01/13
1.1.2	Estudio estado del arte	10 horas	15/01/13	16/01/13
1.1.3	Documentación previa	15 horas	17/01/13	19/01/13
1.2	**Sprint 1**	**55 horas**	**22/01/13**	**02/02/13**
1.2.1	**Análisis de funcionalidades**	**6 horas**	**22/01/13**	**23/01/13**
1.2.1.1	Decisión de las funcionalidades a implementar	0,5 horas	22/01/13	22/01/13
1.2.1.2	Asignación de prioridades	0,5 horas	22/01/13	22/01/13
1.2.1.3	Análisis de las funcionalidades decididas	5 horas	22/01/13	23/01/13
1.2.2	**Diseño de funcionalidades**	**5 horas**	**23/01/13**	**24/01/13**
1.2.2.1	Diseño arquitectura	5 horas	23/01/13	24/01/13
1.2.3	**Implementación de funcionalidades**	**31 horas**	**24/01/13**	**31/01/13**
1.2.3.1	Preparación del entorno	5 horas	24/01/13	25/01/13
1.2.3.2	Creación de la base de datos	5 horas	25/01/13	26/01/13
1.2.3.3	Implementación del módulo de usuarios	6 horas	26/01/13	28/01/13
1.2.3.4	Implementación del módulo de perfiles	4 horas	29/01/13	29/01/13
1.2.3.5	Implementación del módulo de grupos	4 horas	30/01/13	30/01/13
1.2.3.6	Implementación sistema de login	2 horas	31/01/13	31/01/13
1.2.4	Documentación del sprint	10 horas	01/02/13	02/02/13
1.3	Finalización del primer sprint	0 horas	02/02/13	02/02/13
1.4	**Sprint 2**	**70 horas**	**04/03/13**	**24/03/13**
1.4.1	Reunión inicial	3 horas	04/03/13	04/03/13
1.4.2	Análisis de funcionalidades	5 horas	05/03/13	06/03/13
1.4.3	**Diseño de funcionalidades**	**5 horas**	**06/03/13**	**08/03/13**
1.4.3.1	Diseño arquitectura	5 horas	06/03/13	08/03/13
1.4.4	**Implementación de funcionalidades**	**47 horas**	**08/03/13**	**21/03/13**
1.4.4.1	**Cambios sobre el sprint anterior**	**47 horas**	**08/03/13**	**21/03/13**
1.4.4.1.1	Cambios en la base de datos	2 horas	08/03/13	08/03/13
1.4.4.1.2	Nueva configuración del entorno de trabajo	10 horas	09/03/13	11/03/13
1.4.4.1.3	Cambios en los módulos ya desarrollados	25 horas	11/03/13	18/03/13
1.4.4.1.4	Nuevo sistema de login	10 horas	18/03/13	21/03/13
1.4.5	Cambios en la documentación inicial	10 horas	22/03/13	24/03/13
1.5	Finalización del segundo sprint	0 horas	24/03/13	24/03/13
1.6	**Sprint 3**	**70 horas**	**24/03/13**	**13/04/13**

1.6.1	Reunión inicial	3 horas	24/03/13	25/03/13
1.6.2	Análisis de funcionalidades	7 horas	25/03/13	27/03/13
1.6.3	Diseño arquitectura	5 horas	27/03/13	29/03/13
1.6.4	**Implementación de funcionalidades**	**40 horas**	**29/03/13**	**09/04/13**
1.6.4.1	Implementación del sistema de control de permisos	20 horas	29/03/13	03/04/13
1.6.4.2	Acoplamiento con la interfaz	20 horas	04/04/13	09/04/13
1.6.5	Documentación del sprint 2	15 horas	09/04/13	13/04/13
1.7	Finalización del tercer sprint	0 horas	13/04/13	13/04/13
1.8	**Sprint 4**	**73 horas**	**29/04/13**	**20/05/13**
1.8.1	Reunión inicial	3 horas	29/04/13	29/04/13
1.8.2	**Análisis de funcionalidades**	**15 horas**	**07/05/13**	**11/05/13**
1.8.2.1	Familiarización con ISPConfig	15 horas	07/05/13	11/05/13
1.8.3	Diseño arquitectura	5 horas	13/05/13	14/05/13
1.8.4	**Implementación de funcionalidades**	**73 horas**	**29/04/13**	**20/05/13**
1.8.4.1	Cambios sobre la interfaz del módulo de usuarios	25 horas	29/04/13	06/05/13
1.8.4.2	Módulo de clientes de dominios	20 horas	14/05/13	20/05/13
1.9	Finalización del cuarto sprint	0 horas	20/05/13	20/05/13
1.10	**Sprint 5**	**70 horas**	**20/05/13**	**09/06/13**
1.10.1	**Implementación de funcionalidades**	**50 horas**	**20/05/13**	**03/06/13**
1.10.1.1	Finalización del módulo de clientes de dominios	10 horas	20/05/13	23/05/13
1.10.1.2	Módulo de dominios	20 horas	23/05/13	29/05/13
1.10.1.3	Módulo de buzones de correo	20 horas	29/05/13	03/06/13
1.10.2	**Documentación previa**	**20 horas**	**04/06/13**	**09/06/13**
1.10.2.1	Documentación del sprint 3	15 horas	04/06/13	08/06/13
1.10.2.2	Documentación del sprint 4	5 horas	08/06/13	09/06/13
1.11	Finalización del quinto sprint	0 horas	09/06/13	09/06/13
1.12	**Sprint 6**	**73 horas**	**09/06/13**	**30/06/13**
1.12.1	Reunión inicial	3 horas	09/06/13	10/06/13
1.12.2	**Implementación de funcionalidades**	**30 horas**	**10/06/13**	**19/06/13**
1.12.2.1	Cambios sobre el módulo de clientes	10 horas	10/06/13	13/06/13
1.12.2.2	Cambios sobre el módulo de dominios	10 horas	14/06/13	16/06/13
1.12.2.3	Cambios sobre el módulo de Correos	10 horas	16/06/13	19/06/13
1.12.3	**Documentación**	**40 horas**	**19/06/13**	**30/06/13**
1.12.3.1	Documentación del sprint 4	10 horas	19/06/13	22/06/13
1.12.3.2	Documentación del sprint 5	15 horas	22/06/13	26/06/13
1.12.3.3	**Manuales**	**15 horas**	**27/06/13**	**30/06/13**
1.12.3.3.1	Manuales de programador	10 horas	27/06/13	29/06/13
1.12.3.3.2	Manuales de usuario	5 horas	29/06/13	30/06/13
1.13	Finalización del sexto sprint	0 horas	30/06/13	30/06/13
1.14	**Sprint 7 (final)**	**50 horas**	**01/07/13**	**15/07/13**
1.14.1	Finalizar documentación	40 horas	01/07/13	13/07/13
1.14.2	Preparar presentación	10 horas	13/07/13	15/07/13
1.15	Diseño gráfico	20 horas	22/01/13	25/01/13

4.1.2.1 *Explicación de la planificación final*

La planificación de un proyecto software nunca es una tarea trivial, si bien es cierto que gracias a la utilización de una metodología ágil esto se vuelve bastante más sencillo, pues la planificación nunca es en un espacio temporal tan amplio como el del propio proyecto. Además errores de planificación sobre un primer sprint se puede utilizar para corregir estimaciones en los siguientes, ocasionando que cada vez las estimaciones sean más acertadas y reales.

Existen así pues varios motivos que pueden ocasionar rechazos y que aún tras una planificación cuidadosa no pueden ser previstos. A continuación se listarán los más importantes relacionados con este proyecto en concreto:

- **Indisponibilidad del desarrollador:** Debido a varios motivos, principalmente relacionados con las prácticas en empresa y el haber encontrado trabajo no fue posible dedicarle el tiempo planificado al proyecto y por lo tanto hubo que cambiarla casi por completo, provocando que el mes completo de febrero se perdiera.
- **Curva de aprendizaje:** Pese a haberse desarrollado proyectos de menor complejidad con las mismas tecnologías, el enfrentarse a un proyecto más complejo con todo lo que esto acaece (configuraciones, módulos) ocasiona que la curva de aprendizaje se dispare, más teniendo en cuenta que nunca se termina de aprender cómo utilizar correctamente una tecnología.
- **Cambios en los requisitos:** El trabajar para un cliente real con sus dudas y decisiones conlleva que los requisitos sean dinámicos o al menos que deba tenerse en cuenta la posibilidad de cambio, esto se pudo sobrellevar bastante bien gracias a la metodología ágil, sin embargo a veces estos cambios resultaban ser muy pesados y aunque en mayor o menor medida se llevaba una negociación con el cliente estos ocasionaban cambios en el alcance final del proyecto. Haciendo inevitable la necesidad de eliminar cierta funcionalidad del proyecto.
    - **Aplicación para dispositivos móviles Android:** Desde un principio se trato está funcionalidad como algo secundario y la falta del tiempo para la realización del proyecto la relego aún más, llegando a ser descartada del mismo.
    - **Módulo de facturación para dominios:** Uno de mis compañeros ya se encargaba de un módulo relacionado con la gestión de contratos y la facturación de los mismos, al ser necesario realizar la contratación de dominios mediante este no se vio necesario replicar la funcionalidad.

Capítulo 5. **Análisis**

El análisis del proyecto se divide en dos secciones, Sprint 0 correspondiente al análisis realizado tras la primera reunión con el cliente y el análisis realizado durante el resto de sprints, esto es debido a que después de esta primera reunión se decidió un cambio total sobre el enfoque del proyecto que conllevó bastante trabajo perdido, de esta manera se puede apreciar uno de los problemas más comunes en la realización de proyectos para un cliente.

5.1 Sprint 0

En este sprint se hace una primera aproximación al proyecto en su todo, no como en otros sprints en los cuales se harán las aproximaciones semana a semana, se evita en este apartado la definición de clases y se centra en los requisitos facilitados por el cliente al inicio del proyecto. Se ofrece está sección para hacer notar el gran cambio en alcance que surgió el proyecto inicial con el producto final.

5.1.1 Definición del Sistema

5.1.1.1 *Determinación del Alcance del Sistema*

La aplicación consistirá en una herramienta para gestionar dominios contratados por clientes dentro de la empresa Ecocomputer. Se describirá el alcance inicial en función de módulos que posteriormente serán identificados como subsistemas.

La aplicación se desarrollará en un entorno web y contará con una gestión de permisos de acceso y perfiles de usuario, de manera que cada perfil o permiso tendrá acceso a opciones propias del mismo. Los perfiles de usuario definidos inicialmente son los siguientes.

- Administrador
- Técnico de Ecocomputer
- Cliente de mantenimiento

Los administradores serán los encargados de gestionar la página web, podrá dar de alta nuevos usuarios y modificar los niveles de acceso de los mismos.

El personal de Ecocomputer tendrá acceso a toda la información de los clientes con los que trabaja y un control total sobre la documentación asociada a los proyectos.

Los clientes, estarán en posesión de uno o varios dominios, a los cuales tendrá asignados uno o más contactos, los cuales se encargará el de administrar. Por último, los clientes tendrán acceso a una interfaz desde la que podrán gestionar sus buzones de correo.

La aplicación, deberá permitir dar de alta nuevos dominios y vincularlos con un cliente, para lo cual se generará un contrato nuevo con el mismo, el sistema deberá de igual manera controlar

la fecha de vencimiento de esos contratos y avisará al usuario de que la fecha de renovación se acerca mediante correo electrónico. Deberá a su vez, permitir la generación de facturas de manera automática, basándose en la periodicidad acordada con cada cliente. Finalmente constará de una interfaz de correo electrónico basada en QMailAdmin que permita una gestión completa sobre las cuentas de correo electrónico y un buscador de clientes en el back-end de la aplicación, de manera que no sea visible para los usuarios.

Se presentará a los usuarios mediante una interfaz web, cuya funcionalidad estará dividida en distintos módulos, a los cuales se tendrá acceso o no dependiendo de los perfiles y grupos de usuario a los que se pertenezca, en este caso existirá un conjunto de grupos iniciales, que podrá ser posteriormente modificado, estos grupos son.

- Técnico
- Desarrollo
- Administración
- Laboratorio informático

Análogamente, se desarrollará una pequeña aplicación basada en Android, que permita realizar la consulta de datos relativos a un dominio o cliente por parte de los técnicos de Ecocomputer, la gestión de los buzones de correo y realizar búsqueda de incidencias por unos campos específicos.

5.1.2 Requisitos del Sistema

5.1.2.1 *Requisitos funcionales*

5.1.2.1.1 Requisitos del módulo de usuarios

Código	*Nombre*	*Descripción*
RF-1.1	Gestión de perfiles	La aplicación deberá constar de varios perfiles de usuario y gestionarlos. Estos perfiles serán (Administrador, Técnico de Ecocomputer y cliente de mantenimiento).
RF-1.2	Gestión de permisos de acceso	La aplicación garantizará distintos permisos de acceso en función del perfil de usuario que acceda a la aplicación.
RF-1.3	Gestión de clientes	La aplicación tendrá una base de datos con datos fiscales de los clientes y las condiciones de facturación asociadas.
RF-1.4	Identificación de usuarios	La aplicación deberá permitir a los usuarios identificarse en el sistema mediante un login y una contraseña.
RF-1.5	Asociación de usuarios	Los usuarios estarán contenidos en distintos grupos (Técnico, Desarrollo, Administración, Laboratorio informático), pudiendo formar parte de varios.
RF-1.6	Soporte CRUD usuarios	Se podrán dar de alta, baja, modificar y consultar los usuarios de la base de datos.
RF-1.7	Soporte CRUD grupos	Se podrán dar de alta, baja, modificar y consultar los grupos de usuarios de la base de datos.
RF-1.8	Asociación de perfiles	Cada perfil de usuario tendrá asociados unos permisos propios, que determinarán a que módulo y funcionalidades del mismo, podrán o no acceder.

RF-1.9	Herencia de permisos	Los usuarios nuevos que se den de alta en el sistema heredarán los permisos del grupo al que pertenecen.
RF-1.10	Modificación de permisos	La aplicación permitirá la modificación de los permisos de usuario en el sistema.
RF-1.11	Soporte CRUD perfiles	Se podrán dar de alta, baja, modificar y consultar los perfiles de usuario de la base de datos.
RF-1.12	Administrador	Existirá un usuario administrador encargado de la gestión de la página web. Será el único con permisos para realizar operaciones CRUD con usuarios y sus niveles de acceso.
RF-1.13	Acceso a la información	El personal de la empresa deberá ser tener acceso a toda la información de los clientes con los que trabajen.
RF-1.14	Modificación de la documentación	El personal de la empresa deberá ser capaz de consultar, descargar y modificar la documentación desde su cuenta de usuario.
RF-1.15	Búsqueda de clientes	La aplicación permitirá realizar una búsqueda de clientes en el "back-end".
RF-1.16	Recordar contraseña	El usuario podrá pedir a la aplicación que le recuerde su contraseña mediante un correo electrónico.

5.1.2.1.1.1 *Requisitos del módulo de correo electrónico*

Código	***Nombre***	***Descripción***
RF-2.1	Interfaz QMailAdmin	La aplicación constará de una interfaz con QMailAdmin.
RF-2.2	Gestión de cuentas de correo	Mediante la aplicación del requisito RF-2.1 se desarrollará un sistema que permita gestionar todos los aspectos relativos a las cuentas de correo.
RF-2.3	Administración de buzones	El cliente tendrá acceso a una interfaz que le permita administrar sus buzones de correo.
RF-2.4	Alta de cuentas	Los clientes registrados podrán dar de alta cuentas dentro de sus dominios.
RF-2.5	Edición de cuentas	Los clientes registrados podrán editar las cuentas que pertenezcan a sus dominios.
RF-2.6	Baja de buzones de correo	Los clientes registrados podrán dar de baja los buzones de correo asociados a sus dominios.
RF-2.7	Gestión de reenvíos	Los clientes registrados podrán gestionar los reenvíos de los correos dentro de su dominio, pudiendo dar de alta nuevos dominios o modificar los reenvíos ya existentes.
RF-2.8	Modificación de contraseña	Los usuarios podrán cambiar la contraseña de sus cuentas de correo.
RF-2.9	Aumento capacidad	Los usuarios podrán aumentar la capacidad de su buzón de correo.
RF-2.10	Redirección	Los usuarios podrán gestionar las redirecciones dentro de su buzón.
RF-2.11	Autorespuesta	Los usuarios podrán habilitar la opción de autorespuesta dentro de su buzón.

5.1.2.1.1.2 *Requisitos de la gestión de dominios*

Código	***Nombre***	***Descripción***
RF-3.1	Alta de nuevos dominios	La aplicación permitirá la inclusión de nuevos dominios en la base de datos.
RF-3.2	Vinculación con el cliente	La aplicación permitirá la vinculación de los distintos dominios con el cliente al que pertenezcan. Pudiendo ser más de un

Código	*Nombre*	*Descripción*
		dominio por cliente.
RF-3.3	Control de vencimiento	El sistema deberá controlar las fechas de vencimiento de los dominios registrados en EasyNick y Arsys.
RF-3.4	Asociación de contratos	La aplicación permitirá asociar un cliente con un contrato determinado.
RF-3.5	Gestión de servicios	La aplicación dará soporte a varios tipos de servicios (registro y alojamiento de dominios, desarrollo web, etc.).
RF-3.6	Gestión de los precios	El precio hora de los servicios podrá variar entre la cantidad oficial y otras dependiendo del tipo de cliente.
RF-3.7	Generación de facturas	La aplicación permitirá emitir facturas a los clientes en concepto de los servicios que se les hayan prestado. Para ello se utilizará una plantilla estándar.
RF-3.8	Emisión de facturas	La aplicación emitirá las facturas cuando se alcance la fecha acordada con cada cliente, enviándola por correo en formato PDF al departamento de administración de la empresa.
RF-3.9	Asociación de contactos	Cada cliente podrá asociar uno o más contactos a un mismo dominio, cada uno con un usuario y contraseñas diferentes.
RF-3.10	Gestión de contactos	El cliente o el administrador serán los encargados de gestionar los contactos establecidos en los dominios.

5.1.2.2 *Requisitos no funcionales*

5.1.2.2.1 Requisitos de usuario

Código	*Nombre*	*Descripción*
RNF-1.1	Conocimientos previos	Dependiendo del perfil de usuario, deberá tener conocimientos previos sobre cómo funciona la empresa por dentro.
RNF-1.2	Acceso a Internet	En el caso de un cliente, para poder conectarse con la aplicación deberá disponer de una conexión a Internet.

5.1.2.2.2 Requisitos tecnológicos

Código	Nombre	Descripción
RNF-2.1	Desarrollo en entorno web	La aplicación deberá desarrollarse en un entorno web, con interfaces remotas.
RNF-2.2	Integración con la BD de la empresa	El sistema deberá integrarse con la base de datos de la empresa, de manera que no sea necesaria la creación de una nueva.
RNF-2.3	Gestión de menús	La aplicación se dividirá en una serie de módulos con distintos menús para facilitar la gestión de permisos.
RNF-2.4	Utilización de Java	El sistema deberá estar completamente desarrollado en Java, pudiendo utilizarse distintos frameworks gratuitos como Spring y Struts.
RNF-2.5	Compatibilidad de navegadores	La aplicación deberá ser completamente funcional en los distintos navegadores principales.
RNF-2.6	Cumplimiento de estándares	La aplicación se desarrollará teniendo en cuenta los estándares de las tecnologías utilizadas para su programación.
RNF-2.7	Servidor de	Se utilizará un servidor Tomcat para el despliegue de la aplicación,

Código	Nombre	Descripción
	aplicación	este servidor se encontrará alojado en las instalaciones de la empresa.
RNF-2.8	Máquina de Java	La máquina de Java utilizada deberá encontrarse en su versión 1.6+.

5.1.2.2.3 Requisitos de usabilidad y disponibilidad

Código	Nombre	Descripción
RNF-3.1	Disponibilidad	El sistema deberá estar preparado para funcionar 24/7/365, dependiendo en todo momento de la disponibilidad de los servidores de la empresa.
RNF-3.2	Funcionalidad básica	En caso de que no se tenga acceso a Internet, el sistema presentará una funcionalidad básica dentro de la empresa.
RNF-3.3	Usuarios simultáneos	La aplicación deberá proporcionar soporte para usuarios de manera simultánea y sin que esto interfiera en sus actividades.

5.1.2.2.4 Requisitos de seguridad y legalidad

Código	Nombre	Descripción
RNF-4.1	Encriptación	El sistema deberá cifrar la contraseña de los usuarios, de manera que se almacene de forma segura dentro de la base de datos.
RNF-4.2	Protección SQLInjection	Para evitar ataques comunes mediante SQL Injection, se protegerá la base de datos como resulte adecuado.
RNF-4.3	Licencias	Todas las tecnologías utilizadas en esta aplicación serán software libre, de manera que la empresa se reserve el derecho de su utilización, excepto QMailAdmin, cuyo módulo se desarrolla bajo la licencia GPL-v3.
RNF-4.4	Soporte LOPD	Debido a la utilización de datos personales, deberá cumplirse la LOPD dentro del territorio español.

5.1.2.2.5 Otros requisitos no funcionales

Código	Nombre	Descripción
RNF-5.1	Documentación	Se deberá adjuntar a los entregables, documentación de uso y mantenimiento de los distintos módulos.
RNF-5.2	Tiempos de respuesta	La aplicación deberá proporcionar resultados a las acciones en un tiempo de respuesta asequible para el usuario.

5.1.3 Identificación de actores del sistema

5.1.3.1 *Actores primarios*

El sistema constará de una serie de actores primarios, los cuales interactúan directamente con el sistema, estos actores quedan detallados a continuación.

- **Usuario registrado**: Englobará las acciones comunes para todos los actores del sistema.
    - **Trabajador de la empresa:** Será un tipo de usuario registrado, el cual representará todas las acciones que pueden hacer los sub actores que sean de este tipo.
        - **Administrador**: será el gestor de la página web. Tendrá permisos para dar de alta a nuevos usuarios en el sistema y modificar sus niveles de acceso.
        - **Técnico de Ecocomputer**: tendrá acceso a toda la información de los clientes con los que se trabaja, podrá además consultar, descargar y actualizar toda la documentación desde su cuenta de usuario.
    - **Cliente**: tendrá varios dominios registrados a su nombre, cada uno de ellos con un contacto o más asociado. También deberá tener acceso a una interfaz desde la que administrar sus buzones de correo.
- **Temporizador:** Será el encargado de lanzar las tareas automáticas como la generación de avisos del sistema o el envío de facturas.

5.1.3.2 *Actores secundarios*

Al igual que actores que interactúan directamente con el sistema, existen otras partes sobre las que el sistema tendrá repercusión. Por ello se lista a continuación una serie de actores secundarios.

- **Base de datos**: La aplicación almacenará toda la información en una base de datos, la cual será actualizada por el servidor.
- **Sistema de Log:** Deberán almacenarse unos archivos de log en los cuales aparezcan reflejados todos los cambios que se realizan sobre la aplicación.

5.2 Resto del proyecto

En esta sección se ofrece el análisis ya combinado fruto de todos los sprints del proyecto, de manera que resulte más sencillo de asimilar. De esta manera se expondrán los cambios en los requisitos así como las repercusiones que esto tuvo en el alcance y en qué momento del proyecto se dieron estos.

5.2.1 Definición del sistema

5.2.1.1 *Determinación del alcance del sistema*

Debido a cambios en la mentalidad del cliente, lo que en su comienzo era un proyecto propio y totalmente aislado se convertirá en algo común entre todos los alumnos que estemos desarrollando un proyecto para la empresa, esto implica cambios en el alcance inicial que serán expuestos a continuación.

La aplicación se transformará en tres módulos distinguibles entre sí, siendo estos:

- **Intranet:** Será el modulo principal, sirve como nexo de unión entre todos los módulos que se desplieguen sobre el servidor y tengan como objetivo ser parte de la intranet de la empresa, por lo que constará de un sistema de login de manera que sea único para todos ellos. Se implementará en él un sistema de log para ponerlo a disposición del resto de módulos.
- **Módulo de usuarios:** Con motivo de no repetir funcionalidades, recae sobre el proyecto todo lo concerniente a los usuarios, perfiles, grupos y la gestión de permisos del sistema, esté módulo deberá proporcionar todas esas herramientas de manera que sean fácilmente utilizables por el resto de módulos del sistema.

  En este caso los grupos no supondrán una diferencia a la hora de ejercer los permisos y serán meramente informativos, los usuarios y perfiles tendrán permisos propios sobre los módulos definidos en la base de datos, los cuales podrán ser de lectura o escritura.

- **Módulo de dominios:** Lo que en un principio era el objetivo final del proyecto se relega a algo secundario y por lo tanto el alcance se ve reducido, además debido a la decisión de utilizar software de terceros para la gestión de dominios y cuentas de correos, este pasa a ser una interfaz para el susodicho software, de manera que los usuarios puedan interactuar con el sin necesidad de salir de la intranet.

5.2.2 Requisitos del sistema

En esta sección aparecerán las mismas tablas de requisitos del apartado anterior "Sprint0" con los requisitos que se han visto eliminados o modificados tachados para facilitar al lector el entender cómo ha evolucionado el proyecto.

5.2.2.1 *Requisitos funcionales*

5.2.2.1.1 Intranet

Código	Nombre	Descripción
RF-1.1	Sistema de autenticación	La aplicación deberá proveer un sistema de login de manera que una vez el usuario se encuentre dentro de la aplicación no le sea necesario volver a introducir sus credenciales para navegar entre los distintos módulos
RF-1.2	Sistema de log	La aplicación deberá proveer un sistema de log que pueda ser utilizado por todos los módulos del sistema
RF-1.3	Conexiones con la base de datos	Para garantizar que las conexiones con la base de datos se hacen adecuadamente, todas estas se harán a través de este módulo
RF-1.4	Cambio de contraseña	El usuario desde dentro de la aplicación podrá cambiar su contraseña

5.2.2.1.2 Módulo de usuarios

Código	Nombre	Descripción
RF-2.1	Gestión de perfiles	La aplicación deberá constar de varios perfiles de usuario y gestionarlos. ~~Estos perfiles serán (Administrador, Técnico de Ecocomputer y cliente de mantenimiento).~~
RF-2.2	Gestión de permisos de acceso	~~La aplicación garantizará distintos permisos de acceso en función del perfil de usuario que acceda a la aplicación.~~ La aplicación garantizará acceso de lectura y/o escritura en función de los permisos asociados a cada usuario.
~~RF-2.3~~	~~Gestión de clientes~~	~~La aplicación tendrá una base de datos con datos fiscales de los clientes y las condiciones de facturación asociadas.~~ No se consideraba la gestión de clientes como parte del módulo de usuarios y se relego al módulo de gestión de contratos del mismo sistema.
~~RF-2.4~~	~~Identificación de usuarios~~	~~La aplicación deberá permitir a los usuarios identificarse en el sistema mediante un login y una contraseña.~~ Este requisito se ha relegado al nuevo módulo intranet.
RF-2.5	Asociación de usuarios	Los usuarios estarán contenidos en distintos grupos (Técnico, Desarrollo, Administración, Laboratorio informático), pudiendo formar parte de varios.
RF-2.6	Soporte CRUD usuarios	Se podrán dar de alta, baja, modificar y consultar los usuarios de la base de datos.
RF-2.7	Soporte CRUD grupos	Se podrán dar de alta, baja, modificar y consultar los grupos de usuarios de la base de datos.

RF-2.8	Asociación de perfiles	Cada perfil de usuario tendrá asociados unos permisos propios, que determinarán a que módulo y que tipo de acceso tendrán o no.
RF-2.9	Herencia de permisos	Los usuarios nuevos que se den de alta en el sistema heredarán los permisos del perfil al que pertenecen, pudiendo establecerse permisos nuevos.
RF-2.10	Modificación de permisos	La aplicación permitirá la modificación de los permisos de usuario en el sistema.
RF-2.11	Soporte CRUD perfiles	Se podrán dar de alta, baja, modificar y consultar los perfiles de usuario de la base de datos.
RF-2.12	Administrador	~~Existirá un usuario administrador encargado de la gestión de la página web. Será el único con permisos para realizar operaciones CRUD con usuarios y sus niveles de acceso.~~ Existirá un perfil administrador que podrá asociarse a los usuarios que se requiera, de manera que estos tengan acceso a todos los módulos de la aplicación.
~~RF-2.13~~	~~Acceso a la información~~	~~El personal de la empresa deberá ser tener acceso a toda la información de los clientes con los que trabajen.~~
~~RF-2.14~~	~~Modificación de la documentación~~	~~El personal de la empresa deberá ser capaz de consultar, descargar y modificar la documentación desde su cuenta de usuario.~~
~~RF-2.15~~	~~Búsqueda de clientes~~	~~La aplicación permitirá realizar una búsqueda de clientes en el "back-end".~~
~~RF-2.16~~	~~Recordar contraseña~~	~~El usuario podrá pedir a la aplicación que le recuerde su contraseña mediante un correo electrónico.~~ Debido a que los usuarios de la aplicación serán trabajadores de la empresa en su mayoría, que la contraseña va cifrada y que el perfil administrador puede cambiar la contraseña a los usuarios está opción se ha desechado.
RF-2.17	Búsquedas	Para facilitar el manejo de la información, se ofrecerá a los usuarios filtros de búsqueda.

5.2.2.1.3 Módulo de dominios

Código	***Nombre***	***Descripción***
~~RF-3.1~~	~~Alta de nuevos dominios~~	~~La aplicación permitirá la inclusión de nuevos dominios en la base de datos~~. Debido a la inclusión del software de terceros "ISPConfig" la aplicación solo supone un interfaz para el mismo y no se guardan los datos en la base de datos de la aplicación.
RF-3.2	Vinculación con el cliente	La aplicación permitirá la vinculación de los distintos dominios con el cliente al que pertenezcan. Pudiendo ser más de un dominio por cliente.
~~RF-3.3~~	~~Control de vencimiento~~	~~El sistema deberá controlar las fechas de vencimiento de los dominios registrados en EasyNick y Arsys.~~
~~RF-3.4~~	~~Asociación de contratos~~	~~La aplicación permitirá asociar un cliente con un contrato determinado.~~
~~RF-3.5~~	~~Gestión de servicios~~	~~La aplicación dará soporte a varios tipos de servicios (registro y alojamiento de dominios, desarrollo web, etc.).~~
~~RF-3.6~~	~~Gestión de los precios~~	~~El precio hora de los servicios podrá variar entre la cantidad oficial y otras dependiendo del tipo de cliente.~~
~~RF-3.7~~	~~Generación de~~	~~La aplicación permitirá emitir facturas a los clientes en~~

Código	*Nombre*	*Descripción*
	~~facturas~~	~~concepto de los servicios que se les hayan prestado. Para ello se utilizará una plantilla estándar.~~ Esta funcionalidad se relega por completo al módulo de gestión de contratos.
~~RF-3.8~~	~~Emisión de facturas~~	~~La aplicación emitirá las facturas cuando se alcance la fecha acordada con cada cliente, enviándola por correo en formato PDF al departamento de administración de la empresa.~~
RF-3.9	Asociación de contactos	Se podrán asociar uno o más contactos (usuarios del sistema) a un mismo dominio, cada uno con un usuario y contraseñas diferentes.
~~RF-3.10~~	~~Gestión de contactos~~	~~El cliente o el administrador serán los encargados de gestionar los contactos establecidos en los dominios.~~
RF-3.11	Interfaz ISPConfig	La aplicación será un interfaz para el producto ISPConfig, en el cual se puede realizar la gestión de dominios y correos.
RF-3.12	Gestión de cuentas de correo	Mediante la aplicación del requisito RF-3.11 se desarrollará un sistema que permita gestionar todos los aspectos relativos a las cuentas de correo.
RF-3.13	Administración de buzones	El cliente tendrá acceso a una interfaz que le permita administrar sus buzones de correo.
RF-3.14	Alta de cuentas	Los clientes registrados podrán dar de alta cuentas de correo dentro de sus dominios.
RF-3.15	Edición de cuentas	Los clientes registrados podrán editar las cuentas que pertenezcan a sus dominios.
~~RF-3.16~~	~~Baja de buzones de correo~~	~~Los clientes registrados podrán dar de baja los buzones de correo asociados a sus dominios.~~
~~RF-3.17~~	~~Gestión de reenvíos~~	~~Los clientes registrados podrán gestionar los reenvíos de los correos dentro de su dominio, pudiendo dar de alta nuevos dominios o modificar los reenvíos ya existentes.~~
~~RF-3.18~~	~~Modificación de contraseña~~	~~Los usuarios podrán cambiar la contraseña de sus cuentas de correo.~~
~~RF-3.19~~	~~Aumento capacidad~~	~~Los usuarios podrán aumentar la capacidad de su buzón de correo.~~ Solo el administrador del sistema podrá aumentar la capacidad de los buzones de correo.
~~RF-3.20~~	~~Redirección~~	~~Los usuarios podrán gestionar las redirecciones dentro de su buzón.~~
RF-3.21	Autorespuesta	Los usuarios podrán habilitar la opción de autorespuesta dentro de su buzón.

5.2.2.2 *Requisitos no funcionales*

5.2.2.2.1 Requisitos de usuario

Código	*Nombre*	*Descripción*
RNF-1.1	Conocimientos previos	Dependiendo del perfil de usuario, deberá tener conocimientos previos sobre cómo funciona la empresa por dentro.
RNF-1.2	Acceso a Internet	En el caso de un cliente, para poder conectarse con la aplicación deberá disponer de una conexión a Internet.

5.2.2.2.2 Requisitos tecnológicos

Código	Nombre	Descripción
RNF-2.1	Desarrollo en entorno web	La aplicación deberá desarrollarse en un entorno web, con interfaces remotas.
RNF-2.2	Integración con la BD de la empresa	El sistema deberá integrarse con la base de datos de la empresa, de manera que no sea necesaria la creación de una nueva.
RNF-2.3	Gestión de menús	La aplicación se dividirá en una serie de módulos con distintos menús para facilitar la gestión de permisos.
RNF-2.4	Utilización de Java	El sistema deberá estar completamente desarrollado en Java, pudiendo utilizarse distintos frameworks gratuitos como Spring y Struts.
RNF-2.5	Compatibilidad de navegadores	La aplicación deberá ser completamente funcional en los distintos navegadores principales.
RNF-2.6	Cumplimiento de estándares	La aplicación se desarrollará teniendo en cuenta los estándares de las tecnologías utilizadas para su programación.
RNF-2.7	Servidor de aplicación	Se utilizará un servidor Tomcat para el despliegue de la aplicación, este servidor se encontrará alojado en las instalaciones de la empresa.
RNF-2.8	Máquina de Java	La máquina de Java utilizada deberá encontrarse en su versión 1.6+.

5.2.2.2.3 Requisitos de usabilidad y disponibilidad

Código	Nombre	Descripción
RNF-3.1	Disponibilidad	El sistema deberá estar preparado para funcionar 24/7/365, dependiendo en todo momento de la disponibilidad de los servidores de la empresa.
~~RNF-3.2~~	~~Funcionalidad básica~~	~~En caso de que no se tenga acceso a Internet, el sistema presentará una funcionalidad básica dentro de la empresa.~~ Debido a que la base de datos se encuentra en un servidor aparte es imposible ofrecer una funcionalidad básica sin Internet.
RNF-3.3	Usuarios simultáneos	La aplicación deberá proporcionar soporte para usuarios de manera simultánea y sin que esto interfiera en sus actividades.

5.2.2.2.4 Requisitos de seguridad y legalidad

Código	Nombre	Descripción
RNF-4.1	Encriptación	El sistema deberá cifrar la contraseña de los usuarios, de manera que se almacene de forma segura dentro de la base de datos.
RNF-4.2	Protección SQLInjection	Para evitar ataques comunes mediante SQL Injection, se protegerá la base de datos como resulte adecuado.

Código	Nombre	Descripción
RNF-4.3	Licencias	Todas las tecnologías utilizadas en esta aplicación serán software libre, de manera que la empresa se reserve el derecho de su utilización, excepto QMailAdmin, cuyo módulo se desarrolla bajo la licencia GPL-v3.
RNF-4.4	Soporte LOPD	Debido a la utilización de datos personales, deberá cumplirse la LOPD dentro del territorio español.

5.2.2.2.5 Otros requisitos no funcionales

Código	Nombre	Descripción
RNF-5.1	Documentación	Se deberá adjuntar a los entregables, documentación de uso y mantenimiento de los distintos módulos.
~~RNF-5.2~~	~~Tiempos de respuesta~~	~~La aplicación deberá proporcionar resultados a las acciones en un tiempo de respuesta asequible para el usuario.~~ Debido a la utilización de software de terceros, nos vemos limitados por el tiempo de comunicación entre ambas aplicaciones.

5.2.3 Identificación de actores del sistema

5.2.3.1 *Actores primarios*

El sistema contará de una serie de actores primarios, los cuales interactúan directamente con el sistema, estos actores son:

- **Usuario registrado:** Engloba las acciones comunes para todos los usuarios del sistema.
  - **Usuario con acceso de lectura:** Será un usuario registrado con el permiso de lectura para módulos concretos, podrá realizar cualquier operación que no suponga modificación de datos.
  - **Usuario con acceso de escritura:** Podrá realizar además de las opciones de lectura, cualquier modificación sobre los datos para los módulos que tenga permitido.
  - **Cliente:** Los usuarios podrán tener asociados clientes, esto les proporcionará algunas acciones extra sobre los dominios o correos del mismo.

5.2.3.2 *Actores secundarios*

Al igual que actores que interactúan directamente con el sistema, existen otras partes sobre las que el sistema tendrá repercusión. Por ello se lista a continuación una serie de actores secundarios.

- **Base de datos**: La aplicación almacenará toda la información en una base de datos, la cual será actualizada por el servidor.

- **Sistema de Log:** Deberán almacenarse unos archivos de log en los cuales aparezcan reflejados todos los cambios que se realizan sobre la aplicación, estos se guardarán en la base de datos
- **ISPConfig:** La aplicación se comunicará con el software de terceros ISPConfig y producirá modificaciones en su sistema de ficheros o base de datos propia.

5.2.4 Especificación de Casos de Uso

Nótese en este apartado que en la mayor parte de los casos de uso el diagrama de robustez es similar, por lo cual solo se crearán para los primeros casos de uso que resulten distintos de alguna manera significativa a los anteriores.

5.2.4.1 *Usuario registrado*

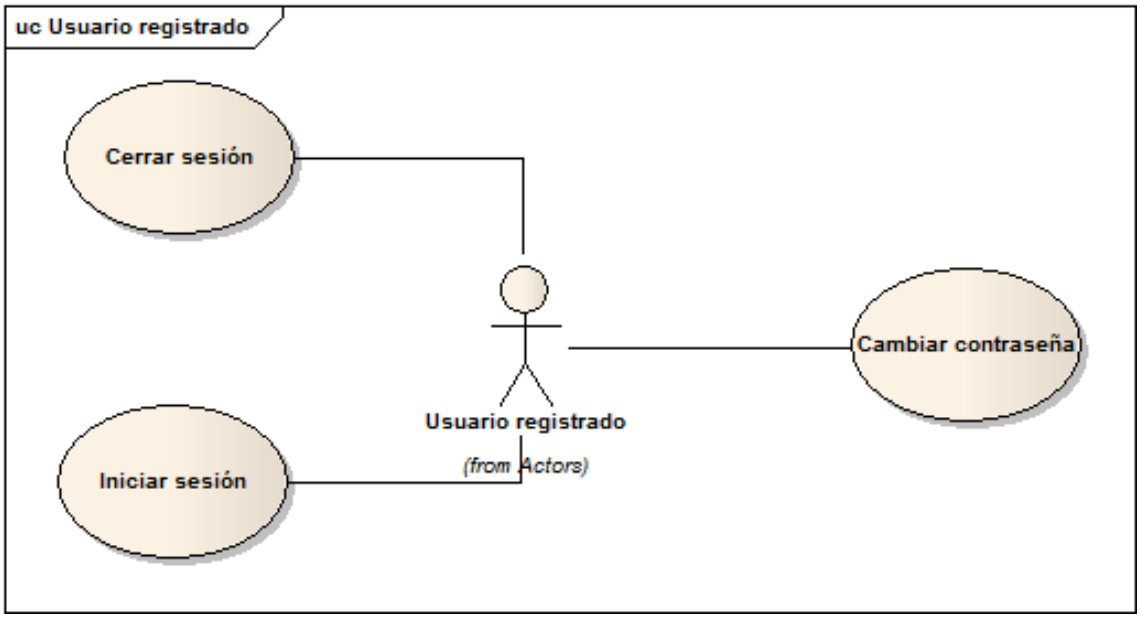


Diagrama 5.1 : Casos de uso-Usuario resgistrado

5.2.4.1.1 Iniciar sesión

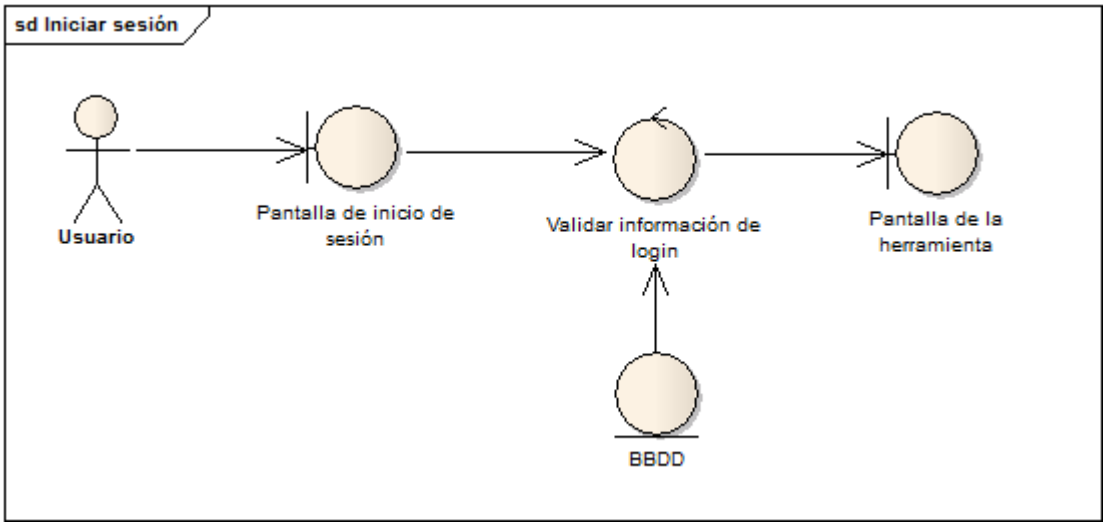


Diagrama 5.2: Robustez-Iniciar sesión

Iniciar sesión	
Precondiciones	El usuario debe encontrarse fuera de la aplicación.
Poscondiciones	El usuario se encontrará debidamente identificado en la aplicación.

Actores	Iniciado y terminado por el usuario.
Descripción	1. El sistema muestra la pantalla de inicio de sesión 2. El usuario rellena el formulario con sus datos de acceso y los confirma 3. El sistema valida los datos de acceso contra la base de datos y carga el usuario en sesión 4. El usuario se encuentra dentro de la aplicación

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Alguno de los campos del formulario quedan sin rellenar.
	  ○ Se le muestra un mensaje informativo al usuario en la pantalla de login.
	• **Escenario Alternativo 2**: Las credenciales de acceso son erróneas.
	  ○ Se le muestra al usuario un mensaje informándole de ello sin especificar en que se ha equivocado por motivos de seguridad.

Excepciones	• **La base de datos no está disponible**: No se puede realizar la comprobación de los datos de usuario. ○ Notificar un error asociado al problema encontrado
Notas	En caso de que la base de datos no esté disponible se le notificará al usuario con un error indicándole que intente loguearse al cabo de unos minutos.

5.2.4.1.2 Cerrar sesión

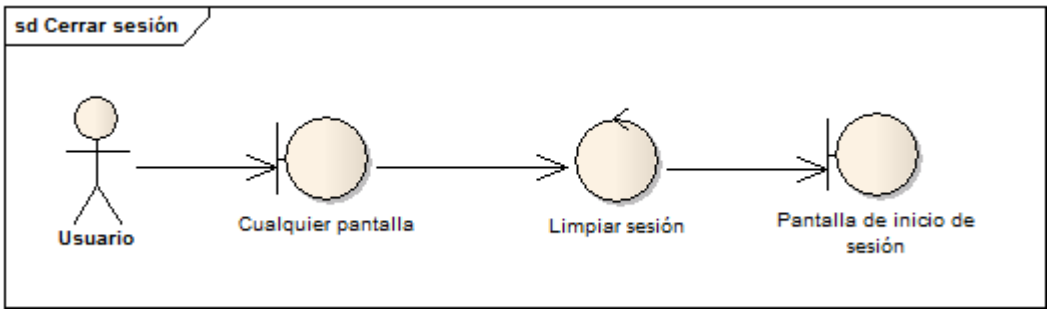


Diagrama 5.3: Robustez-Cerrar sesión

Cerrar sesión	
Precondiciones	El usuario debe encontrarse identificado en la aplicación.
Poscondiciones	El usuario se encontrará fuera de la aplicación.
Actores	Iniciado y terminado por el usuario.
Descripción	1. El usuario pincha sobre el enlace de cerrar sesión 2. El sistema limpia los datos de usuario de la sesión 3. El sistema muestra la pantalla de Inicio de sesión 4. El usuario se encuentra fuera de la aplicación
Notas	Junto con el usuario, se eliminará cualquier información relativa al mismo en sesión.

5.2.4.1.3 Cambiar contraseña

Cambiar contraseña	
Precondiciones	El usuario deberá encontrarse identificado en la aplicación y conocer su contraseña actual
Poscondiciones	El usuario se encontrará fuera de la aplicación y su contraseña se habrá modificado.
Actores	Iniciado y terminado por el usuario.
Descripción	1. El sistema muestra la pantalla de perfil del usuario 2. El usuario rellena el formulario con su contraseña actual y repitiendo la nueva dos veces. 3. El sistema valida los datos. 4. El sistema actualiza la información en la base de datos. 5. El usuario se encuentra fuera de la aplicación.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: Alguno de los campos del formulario quedan sin rellenar.
	  ○ Se le muestra un mensaje informativo al usuario.
	• **Escenario Alternativo 2**: Las contraseñas no coinciden.
	  ○ Se le muestra al usuario un mensaje informándole de ello.

Excepciones	• **La base de datos no está disponible**: No se puede realizar la comprobación de los datos de usuario. ○ Notificar un error asociado al problema encontrado
Notas	La contraseña deberá cumplir ciertos criterios (mínimo de 8 caracteres, una mayúscula, minúscula, digito y símbolo). La contraseña nueva se cifra y se almacena cifrada.

5.2.4.2 *Usuario con permisos para el módulo de usuarios*

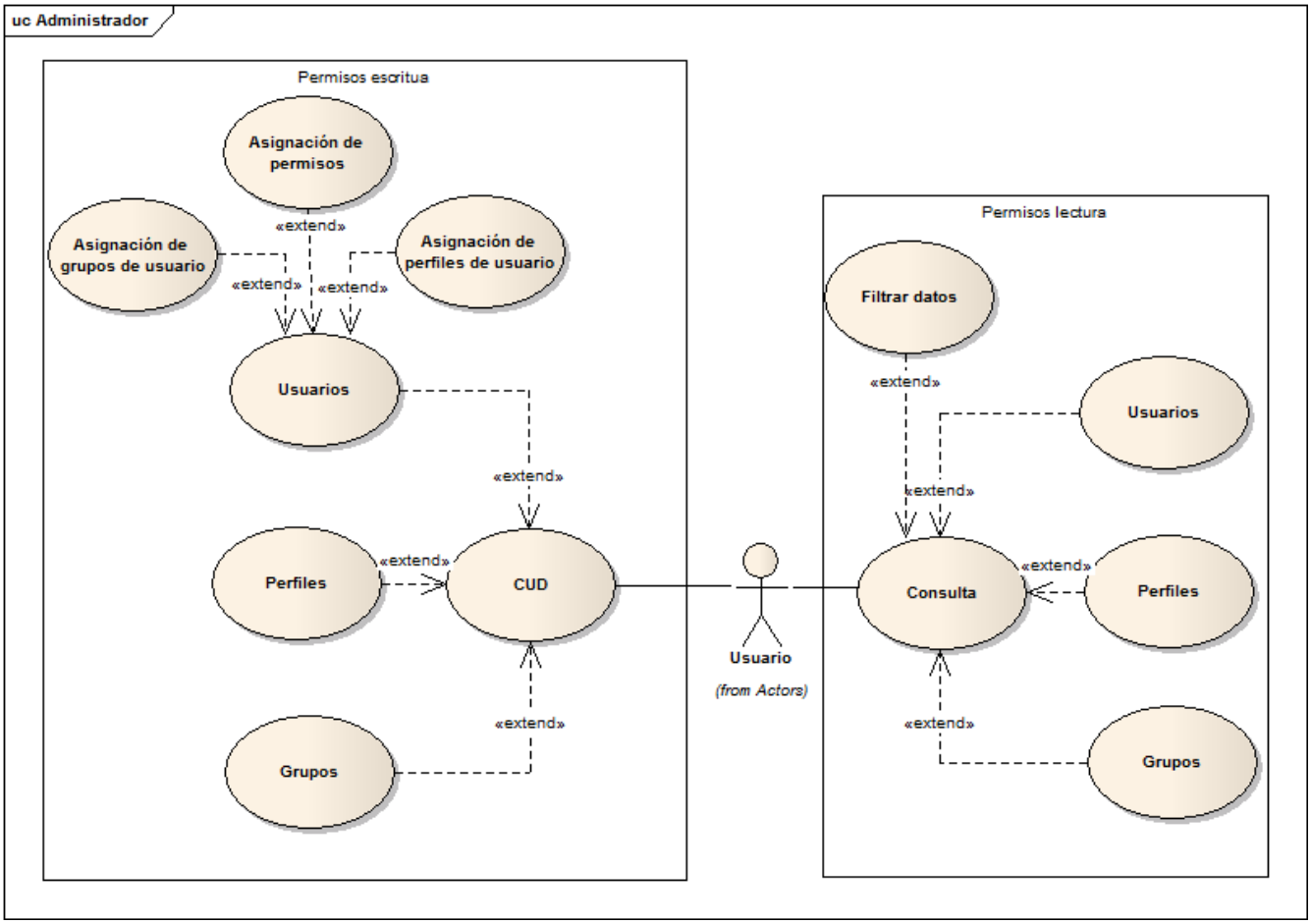


Diagrama 5.4 : Casos de uso-Usuario con permisos en el módulo de usuarios

5.2.4.2.1 Crear usuario

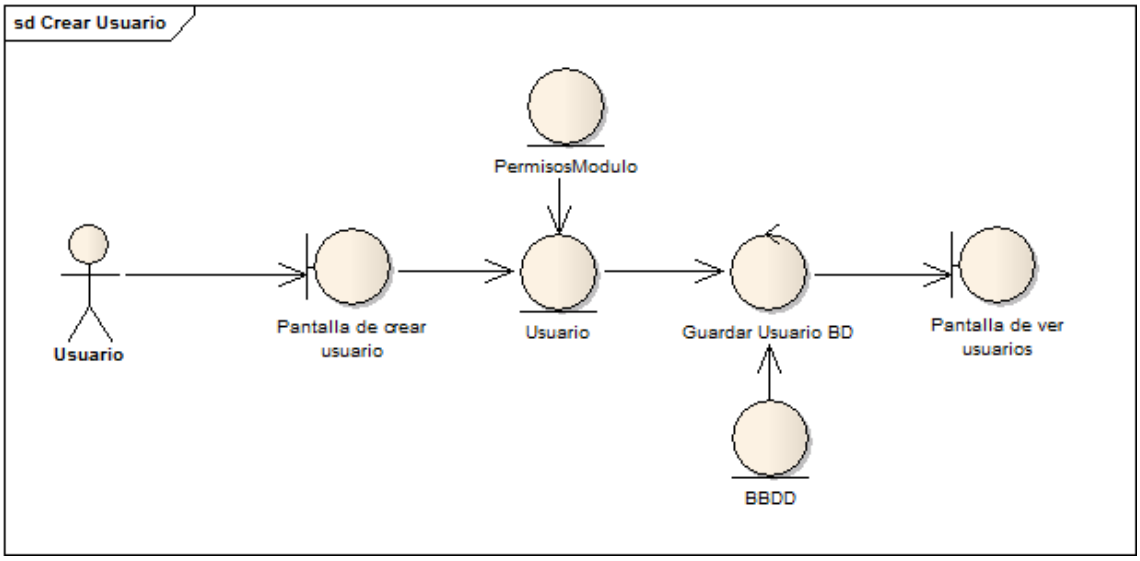


Diagrama 5.5: Robustez-Crear usuario

Crear Usuario	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de perfiles, grupos y módulos necesaria.
Poscondiciones	Debe existir un nuevo usuario con identificador único en el sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de nuevo usuario 2. El sistema cargará los datos relativos a los perfiles, módulos y grupos 3. El sistema mostrará la pantalla con el formulario de crear usuarios 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Usuario y generará los permisos correspondientes para el mismo 6. El sistema actualizará la base de datos con la inserción del nuevo usuario y la modificación de las tablas correspondientes 7. Se enviará al usuario a la pantalla de ver usuarios

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación
	  ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo posteriormente.
	• **Escenario Alternativo 2**: El usuario que se intenta crear ya existe en la base de datos
	  ○ Se informa al usuario de que el usuario que intenta crear ya se encuentra en la base de datos
	  ○ Se le redirige a la pantalla de creación de usuarios

Excepciones	• **La base de datos no está disponible**: No se pueden guardar usuarios.

	○ Se informa al usuario de que no puede crear nuevos usuarios temporalmente. • **La base de datos falla al insertar el usuario**: Durante el proceso de inserción este no se termina ○ Se realiza un rollback de la base de datos y se notifica al usuario que no se guardó el nuevo usuario.
Notas	-

5.2.4.2.2 Editar usuario

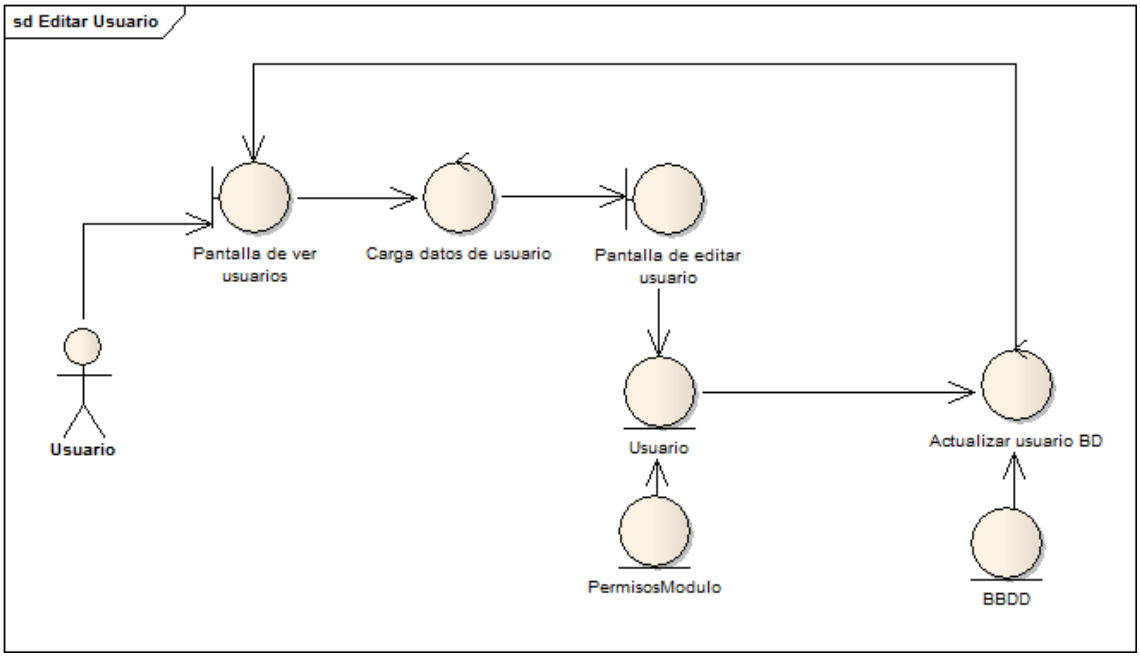


Diagrama 5.6: Robustez-Editar usuario

Editar Usuario	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de perfiles, grupos y módulos necesaria.
Poscondiciones	Los datos del usuario seleccionado deberán aparecer modificados en el sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de editar usuario 2. El sistema carga los datos del usuario seleccionado y los datos relativos a perfiles, módulos y grupos 3. El sistema mostrará la pantalla con el formulario de edición de usuarios 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Usuario y generará los permisos correspondientes para el mismo 6. El sistema actualizará la base de datos con la modificación de las tablas correspondientes 7. Se enviará al usuario a la pantalla de ver usuarios
Variaciones (escenarios	• **Escenario Alternativo 1**: El formulario está incompleto o no

secundarios)	cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo posteriormente. • **Escenario Alternativo 2**: El usuario que se intenta modificar se superpone a uno ya existente en la base de datos ○ Se informa al usuario de que el usuario que intenta modificar colisiona con uno que ya se encuentra en la base de datos ○ Se le redirige a la pantalla de edición de usuarios
Excepciones	• **La base de datos no está disponible**: No se pueden guardar usuarios. ○ Se informa al usuario de que no puede realizar la actualización temporalmente. • **La base de datos falla al insertar el usuario**: Durante el proceso de inserción este no se termina ○ Se realiza un rollback de la base de datos y se notifica al usuario que no se guardaron las modificaciones.
Notas	Los usuarios se superpondrán cuando tengan el mismo login.

5.2.4.2.3 Ver usuarios

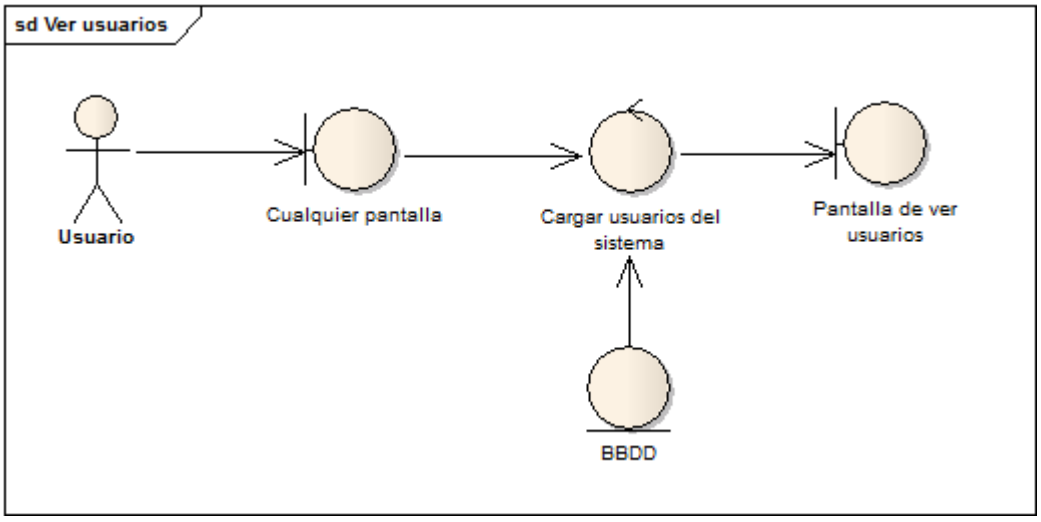


Diagrama 5.7: Robustez-Ver usuarios

Ver usuarios	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de usuarios necesaria.
Poscondiciones	Se le muestran al usuario todos los demás usuarios del sistema
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de ver usuarios 2. El sistema carga la información de los usuarios en la base de datos 3. El sistema muestra toda la información en una interfaz web
Excepciones	• **La base de datos no está disponible**: No se pueden consultar

	los usuarios ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	Para cada usuario deberá recogerse además la información de sus perfiles y grupos de usuario.

5.2.4.2.4 Eliminar usuario

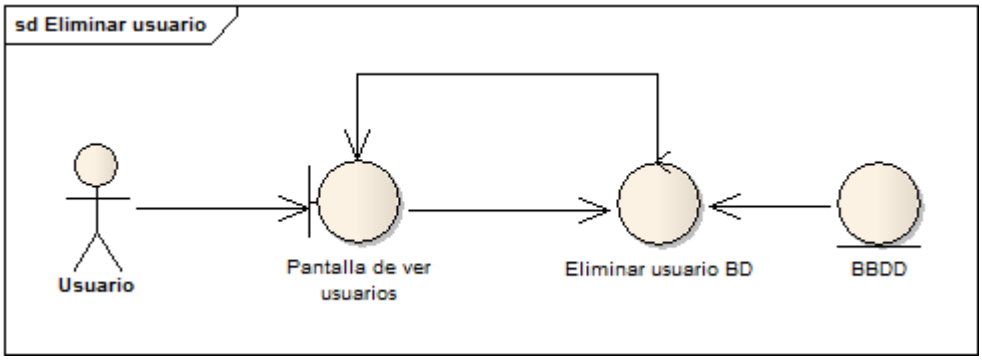


Diagrama 5.8: Robustez-Eliminar usuario

Eliminar usuario	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de usuarios necesaria.
Poscondiciones	El usuario se marca como eliminado en la base de datos y se pierde el acceso al mismo desde la aplicación.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de eliminar relativo al usuario que desea borrar 2. El sistema muestra un dialogo de aceptación 3. El usuario confirma la acción 4. El sistema marca como eliminado el usuario en la base de datos 5. El sistema muestra la página de ver usuarios actualizada
Excepciones	• **La base de datos no está disponible**: No se pueden consultar los usuarios ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript. El usuario no se elimina, solo se marca como tal.

5.2.4.2.5 Crear perfil

Crear Perfil	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de usuarios y módulos.
Poscondiciones	Debe existir un nuevo perfil con identificador único en el sistema
Actores	Iniciado y terminado por el usuario

Descripción	1. Accederá a la pantalla de nuevo perfil 2. El sistema cargará los datos relativos a los usuarios y módulos 3. El sistema mostrará la pantalla con el formulario de crear perfiles 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Perfil y generará los permisos correspondientes para el mismo 6. El sistema actualizará la base de datos con la inserción del nuevo perfil y la modificación de las tablas correspondientes 7. Se enviará al usuario a la pantalla de ver perfiles

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación
	  ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo posteriormente.
	• **Escenario Alternativo 2**: El perfil que se intenta crear ya existe en la base de datos
	  ○ Se informa al usuario de que el perfil que intenta crear ya se encuentra en la base de datos
	  ○ Se le redirige a la pantalla de creación de perfiles

Excepciones	• **La base de datos no está disponible**: No se pueden guardar perfiles.
	  ○ Se informa al usuario de que no puede crear nuevos perfiles temporalmente.
	• **La base de datos falla al insertar el perfil**: Durante el proceso de inserción este no se termina
	  ○ Se realiza un rollback de la base de datos y se notifica al usuario que no se guardó el nuevo perfil.

Notas	Dos perfiles colisionan cuanto tienen la misma descripción.

5.2.4.2.6 Editar perfil

Editar perfil	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de usuarios y módulos necesaria.
Poscondiciones	Los datos del perfil seleccionado deberán aparecer modificados en el sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de editar perfil 2. El sistema carga los datos del perfil seleccionado y los datos relativos a usuarios y módulos 3. El sistema mostrará la pantalla con el formulario de edición de perfiles 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Perfil y generará los permisos correspondientes para el mismo 6. El sistema actualizará la base de datos con la modificación de las

	tablas correspondientes 7. Se enviará al usuario a la pantalla de ver perfiles

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación
	  ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo posteriormente.
	• **Escenario Alternativo 2**: El perfil que se intenta modificar se superpone a uno ya existente en la base de datos
	  ○ Se informa al usuario de que el perfil que intenta modificar colisiona con uno que ya se encuentra en la base de datos
	  ○ Se le redirige a la pantalla de edición de perfiles

Excepciones	• **La base de datos no está disponible**: No se pueden guardar perfiles.
	  ○ Se informa al usuario de que no puede realizar la actualización temporalmente.
	• **La base de datos falla al insertar el perfil**: Durante el proceso de inserción este no se termina
	  ○ Se realiza un rollback de la base de datos y se notifica al usuario que no se guardaron las modificaciones.

Notas	Los perfiles colisionarán cuando tengan la misma descripción.

5.2.4.2.7 Ver perfiles

Ver perfiles	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de perfiles necesaria.
Poscondiciones	Se le muestran al usuario todos los perfiles del sistema
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de perfiles 2. El sistema carga la información de los perfiles en la base de datos 3. El sistema muestra toda la información en una interfaz web
Excepciones	• **La base de datos no está disponible**: No se pueden consultar los perfiles ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	Para cada perfil deberá recoger además la información de los usuarios y módulos.

5.2.4.2.8 Eliminar perfiles

Eliminar perfiles	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de perfiles necesaria.
Poscondiciones	El perfil se marca como eliminado en la base de datos y se pierde el

	acceso al mismo desde la aplicación.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de eliminar relativo al perfil que desea borrar 2. El sistema muestra un dialogo de aceptación 3. El usuario confirma la acción 4. El sistema marca como eliminado el perfil en la base de datos 5. El sistema muestra la página de ver perfiles actualizada
Excepciones	• **La base de datos no está disponible**: No se pueden consultar los perfiles ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript. El perfil no se elimina, solo se marca como tal.

5.2.4.2.9 Crear grupo

Crear grupo	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de usuarios necesaria.
Poscondiciones	Debe existir un nuevo grupo con identificador único en el sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de nuevo grupo 2. El sistema cargará los datos relativos a los usuarios 3. El sistema mostrará la pantalla con el formulario de crear grupos 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Grupo 6. El sistema actualizará la base de datos con la inserción del nuevo grupo y la modificación de las tablas correspondientes 7. Se enviará al usuario a la pantalla de ver grupos

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación
	  ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo posteriormente.
	• **Escenario Alternativo 2**: El grupo que se intenta crear ya existe en la base de datos
	  ○ Se informa al usuario de que el grupo que intenta crear ya se encuentra en la base de datos
	  ○ Se le redirige a la pantalla de creación de grupos

Excepciones	• **La base de datos no está disponible**: No se pueden guardar grupos. ○ Se informa al usuario de que no puede crear nuevos grupos temporalmente. • **La base de datos falla al insertar el grupo**: Durante el proceso de inserción este no se termina

	○ Se realiza un rollback de la base de datos y se notifica al usuario que no se guardó el nuevo grupo.
Notas	Dos grupos colisionarán si tienen la misma descripción.

5.2.4.2.10 Editar grupo

Editar grupo	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de usuarios necesaria.
Poscondiciones	Los datos del grupo seleccionado deberán aparecer modificados en el sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de editar grupo 2. El sistema carga los datos del grupo seleccionado y los datos relativos a usuarios 3. El sistema mostrará la pantalla con el formulario de edición de grupos 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Grupo 6. El sistema actualizará la base de datos con la modificación de las tablas correspondientes 7. Se enviará al usuario a la pantalla de ver grupos
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo posteriormente. • **Escenario Alternativo 2**: El grupo que se intenta modificar se superpone a uno ya existente en la base de datos ○ Se informa al usuario de que el grupo que intenta modificar colisiona con uno que ya se encuentra en la base de datos ○ Se le redirige a la pantalla de edición de grupos
Excepciones	• **La base de datos no está disponible**: No se pueden guardar grupos. ○ Se informa al usuario de que no puede realizar la actualización temporalmente. • **La base de datos falla al insertar el grupo**: Durante el proceso de inserción este no se termina ○ Se realiza un rollback de la base de datos y se notifica al usuario que no se guardaron las modificaciones.
Notas	Los grupos colisionarán cuanto tengan la misma descripción.

5.2.4.2.11 Ver grupos

Ver grupos	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de grupos. Además, deberá haberse cargado

	antes la información de grupos necesaria.
Poscondiciones	Se le muestran al usuario todos los grupos del sistema
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de ver grupos 2. El sistema carga la información de los grupos en la base de datos 3. El sistema muestra toda la información en una interfaz web
Excepciones	• **La base de datos no está disponible**: No se pueden consultar los grupos ○ Se informa al grupo de que no puede realizar la operación temporalmente.
Notas	Para cada grupo deberá recogerse además la información de sus miembros.

5.2.4.2.12 Eliminar grupo

Eliminar grupo	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de usuarios. Además, deberá haberse cargado antes la información de grupos necesaria.
Poscondiciones	El grupo se marca como eliminado en la base de datos y se pierde el acceso al mismo desde la aplicación.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de eliminar relativo al grupo que desea borrar 2. El sistema muestra un dialogo de aceptación 3. El usuario confirma la acción 4. El sistema marca como eliminado el grupo en la base de datos 5. El sistema muestra la página de ver grupos actualizada
Excepciones	• **La base de datos no está disponible**: No se pueden consultar los grupos ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript. El grupo no se elimina, solo se marca como tal.

5.2.4.3 *Usuario con permisos para el módulo de dominios*

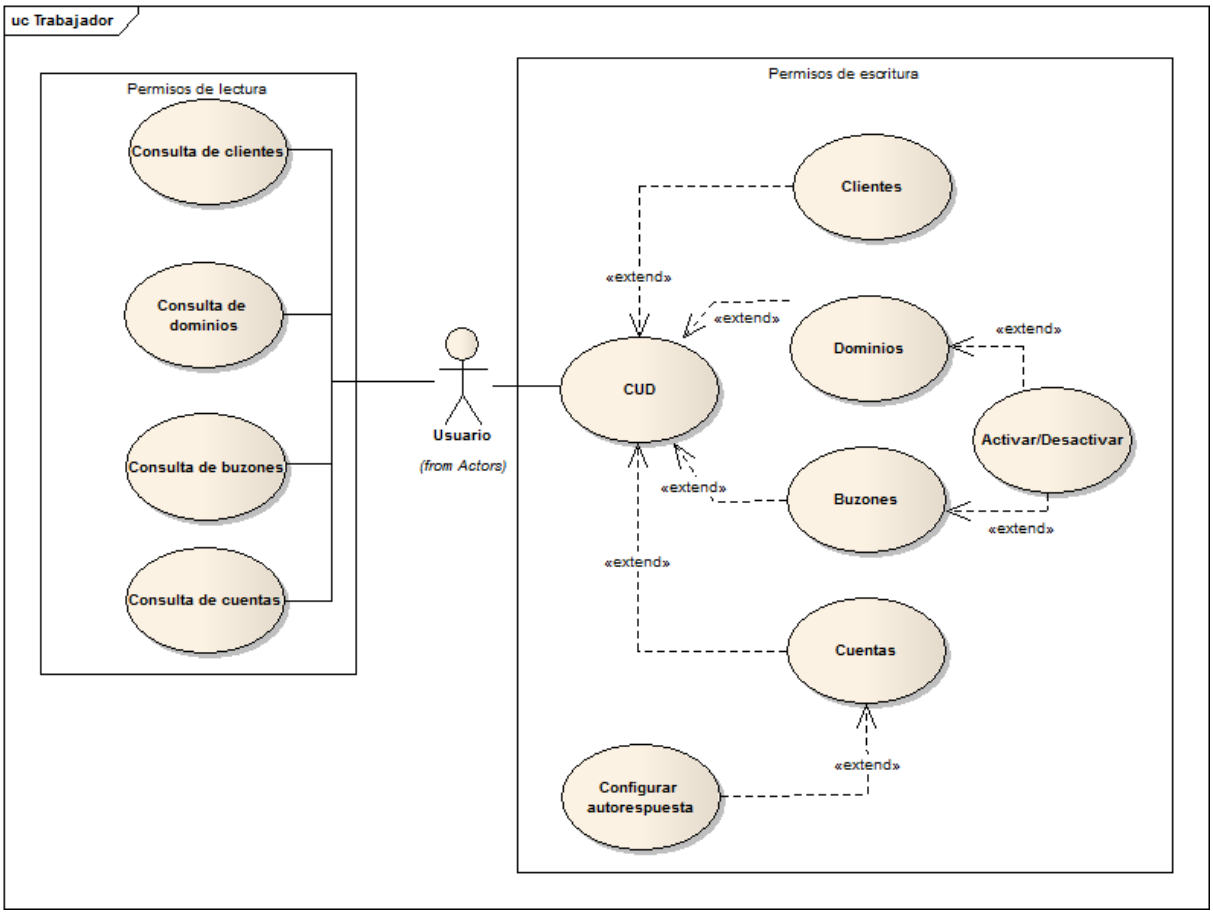


Diagrama 5.9: Casos de uso-Usuario con permisos en el módulo de dominios

5.2.4.3.1 **Crear Cliente**

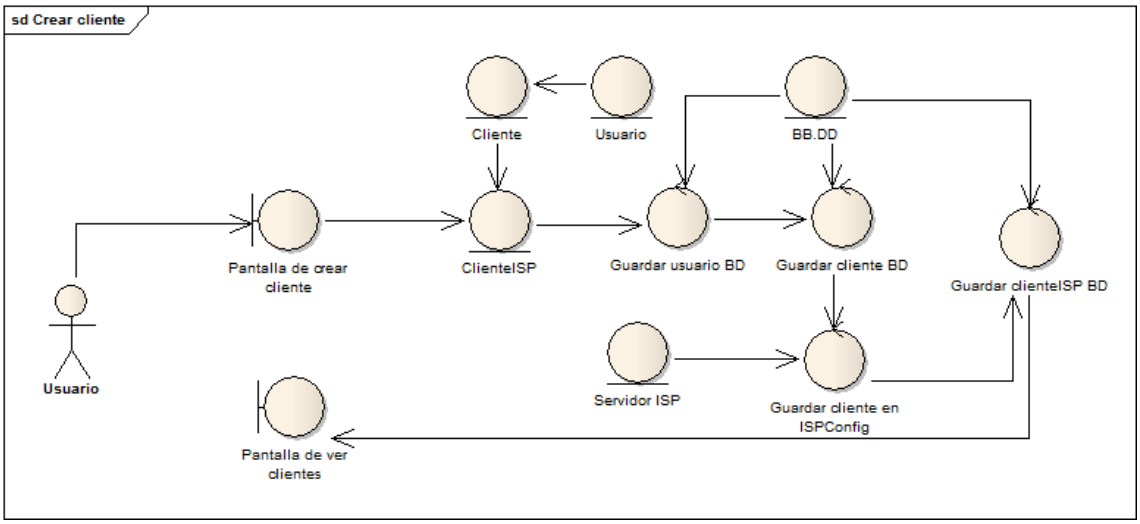


Diagrama 5.10: Robustez-Crear cliente

Crear Cliente	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios. Además, deberá haberse cargado

	antes la información de usuarios y clientes necesaria.
Poscondiciones	Pueden darse tres Poscondiciones posibles: • Existen un nuevo usuario, cliente y relación con el servidor ISPConfig en la base de datos y en el servidor. • Existe un nuevo usuario o un nuevo cliente en la base de datos y en el servidor. • Existe un nuevo usuario en el servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de nuevo cliente 2. El sistema cargará los datos relativos a los usuarios y clientes. 3. El sistema mostrará la pantalla con el formulario de crear clientes. 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en varios objetos, Usuario, Cliente y ClienteISP. 6. El sistema creará el nuevo usuario en la base de datos. 7. El sistema creará el nuevo cliente en la base de datos. 8. El sistema creará la relación del cliente con el servidor ISPConfig. 9. El sistema generará un fichero JSON con el objeto cliente ISP y se lo pasará a un script PHP. 10. El sistema invocará el método remoto para crear el usuario en ISPConfig 11. Se enviará al usuario a la pantalla de ver clientes.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **La base de datos no está disponible:** No se pueden guardar clientes. • **El servidor ISPConfig no está disponible**: No se pueden guardar clientes. ○ Se informa al usuario de que no puede crear nuevos clientes temporalmente.
Notas	En el caso de que falle la conexión con el servidor quedarán creados en base de datos los objetos usuario y cliente en caso de que fueran necesarios.

5.2.4.3.2 Editar cliente

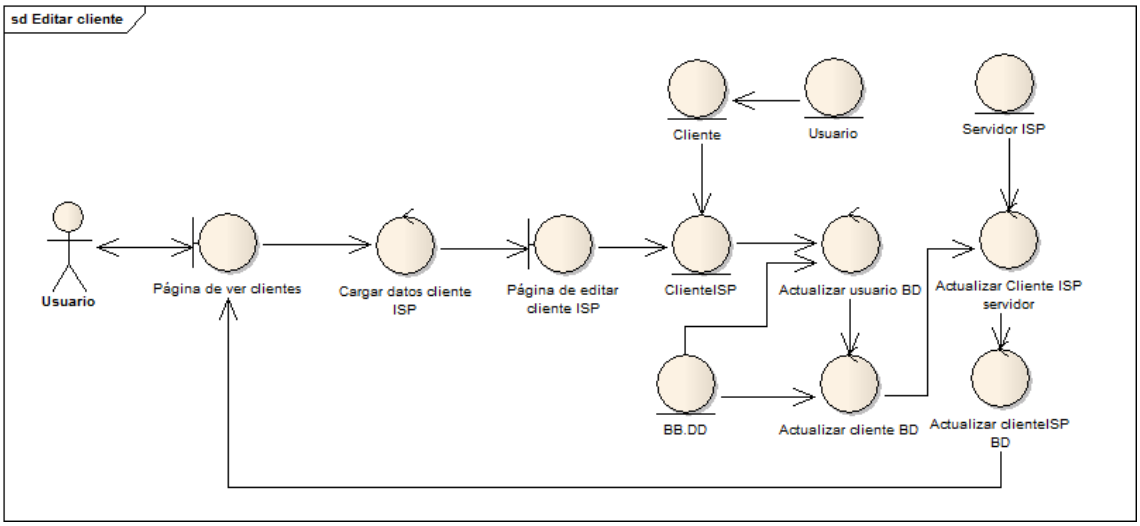


Diagrama 5.11: Robustez-Editar cliente

Editar Cliente	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios. Además, deberá haberse cargado antes la información de usuarios y clientes necesaria.
Poscondiciones	La información del cliente deberá encontrarse actualizada en el servidor ISPConfig y en la base de datos.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de editar cliente. 2. El sistema carga los datos del cliente seleccionado y los datos relativos a usuarios y clientes. 3. El sistema mostrará la pantalla con el formulario de edición de clientes. 4. El usuario rellenará los campos del formulario necesarios y los enviará. 5. El sistema transformará los datos en objetos Usuario, cliente y clienteISP. 6. El sistema actualizara la información de la base de datos con los objetos Usuario y Cliente. 7. El sistema transformará el objeto ClienteISP en un fichero JSON. 8. El sistema invocará remotamente el método de actualización del servidor mediante un script en PHP. 9. Se enviará al usuario a la pantalla de ver cliente.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **La base de datos no está disponible:** No se pueden actualizar los datos de usuario ni de cliente. • **El servidor ISPConfig no está disponible**: No se pueden guardar clientes.

	○ Se informa al usuario de que no puede crear nuevos dominios temporalmente.
Notas	-

5.2.4.3.3 Ver clientes

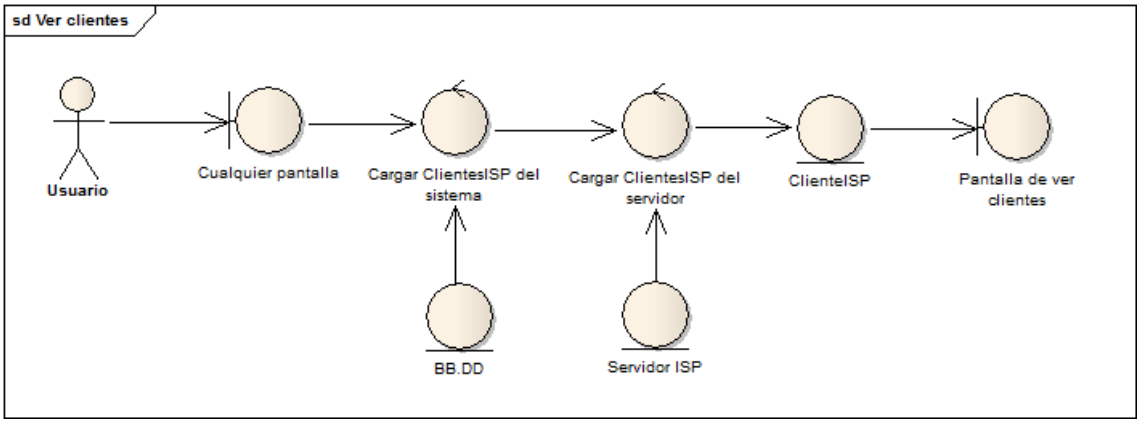


Diagrama 5.12: robustez-Ver clientes

Ver clientes	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de dominios. Además, deberá haberse cargado antes la información de dominios necesaria.
Poscondiciones	Se le muestran al usuario todos los clientes del sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de ver clientes 2. El sistema carga la información de los clientes relacionados con el servidor ISPConfig. 3. El sistema realiza una conexión con el servidor por cada ID de cliente asociado para obtener los datos. 4. El sistema muestra toda la información en una interfaz web
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden consultar los clientes ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	Para cada usuario deberá recogerse la información de los dominios que tenga asociados.

5.2.4.3.4 Eliminar cliente

Eliminar cliente	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	El cliente desaparecerá del servidor ISPConfig y se eliminará la relación de la base de datos con el mismo.
Actores	Iniciado y terminado por el usuario

Descripción	1. El usuario pulsa sobre el botón de eliminar relativo al cliente que desea borrar. 2. El sistema muestra un dialogo de aceptación. 3. El usuario confirma la acción. 4. El sistema elimina la información relativa a la relación del cliente con el servidor de la base de datos. 5. El sistema invoca remotamente el método de eliminar clientes del servidor. 6. El sistema muestra la página de ver clientes actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede eliminar el dominio ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript.

5.2.4.3.5 Crear dominio

Crear Dominio	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios. Además, deberá haberse cargado antes la información de servidores y clientes necesaria.
Poscondiciones	Debe existir un nuevo dominio en el software ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de nuevo dominio 2. El sistema cargará los datos relativos a los servidores, dominios y clientes. 3. El sistema mostrará la pantalla con el formulario de crear dominios. 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Dominio 6. El sistema transformara el objeto Dominio en un archivo JSON y se lo pasará al script PHP. 7. El sistema invocará el método remoto para crear el dominio en ISPConfig 8. Se enviará al usuario a la pantalla de ver dominios
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden guardar dominios. ○ Se informa al usuario de que no puede crear nuevos dominios temporalmente.
Notas	-

5.2.4.3.6 Editar dominio

Editar Dominio	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios. Además, deberá haberse cargado antes la información de servidores y clientes necesaria.
Poscondiciones	La información del dominio deberá encontrarse actualizada en el servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de editar dominio 2. El sistema carga los datos del dominio seleccionado y los datos relativos a dominios, servidores y clientes. 3. El sistema mostrará la pantalla con el formulario de edición de dominios. 4. El usuario rellenará los campos del formulario necesarios y los enviará. 5. El sistema transformará los datos en un objeto Dominio. 6. El sistema transformará el objeto Dominio en un fichero JSON. 7. El sistema invocará remotamente el método de actualización del servidor mediante un script en PHP. 8. Se enviará al usuario a la pantalla de ver dominios.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden guardar dominios. ○ Se informa al usuario de que no puede crear nuevos dominios temporalmente.
Notas	-

5.2.4.3.7 Ver dominios

Ver dominios	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de dominios. Además, deberá haberse cargado antes la información de dominios, clientes y servidores necesaria.
Poscondiciones	Se le muestran al usuario todos los dominios del sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de ver dominios 2. El sistema carga la información de los dominios en el servidor ISPConfig. 3. El sistema muestra toda la información en una interfaz web
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden consultar los dominios ○ Se informa al usuario de que no puede realizar la

	operación temporalmente.
Notas	Para cada dominio deberá recogerse la información de su cliente y servidor.

5.2.4.3.8 Eliminar dominio

Eliminar dominio	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	El dominio desaparece del servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de eliminar relativo al dominio que desea borrar 2. El sistema muestra un dialogo de aceptación 3. El usuario confirma la acción 4. El sistema invoca remotamente el método de eliminar dominios del servidor. 5. El sistema muestra la página de ver dominios actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede eliminar el dominio ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript.

5.2.4.3.9 Activar/Desactivar dominio

Activar/Desactivar dominio	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	El dominio cambia de estado en el servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de activar/desactivar relativo al dominio. 2. El sistema invoca remotamente el método de activar/desactivar dominios del servidor. 3. El sistema muestra la página de ver dominios actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede activar el dominio ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	-

5.2.4.3.10 Crear buzón

Crear Buzón	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios. Además, deberá haberse cargado

	antes la información de servidores y clientes necesaria.
Poscondiciones	Debe existir un nuevo buzón en el software ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de ver buzones 2. El sistema cargará los datos relativos a los servidores y clientes. 3. El usuario rellenará los campos del formulario necesarios y los enviará 4. El sistema transformará los datos en un objeto Buzón 5. El sistema transformara el objeto Buzón en un archivo JSON y se lo pasará al script PHP. 6. El sistema invocará el método remoto para crear el buzón en ISPConfig 7. Se enviará al usuario a la pantalla de ver buzones.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden guardar buzones. ○ Se informa al usuario de que no puede crear nuevos dominios temporalmente.
Notas	Al ser tan pocos los datos necesarios no se ve necesidad de crear una pantalla nueva.

5.2.4.3.11 Ver buzones

Ver buzones	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de dominios. Además, deberá haberse cargado antes la información de dominios, clientes y servidores necesaria.
Poscondiciones	Se le muestran al usuario todos los buzones del sistema.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de ver buzones 2. El sistema carga la información de los buzones en el servidor ISPConfig. 3. El sistema muestra toda la información en una interfaz web
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden consultar los buzones ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	Para cada buzón deberá recogerse la información de su cliente y servidor.

5.2.4.3.12 Eliminar buzón

Eliminar dominio	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	El buzón desaparece del servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de eliminar relativo al buzón que desea borrar 2. El sistema muestra un dialogo de aceptación 3. El usuario confirma la acción 4. El sistema invoca remotamente el método de eliminar buzones del servidor. 5. El sistema muestra la página de ver buzones actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede eliminar el dominio ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript.

5.2.4.3.13 Activar/Desactivar buzón

Activar/Desactivar buzón	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	El buzón cambia de estado en el servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de activar/desactivar relativo al buzón. 2. El sistema invoca remotamente el método de activar/desactivar buzones del servidor. 3. El sistema muestra la página de ver buzones actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede activar el buzón ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	-

5.2.4.3.14 Crear cuenta

Crear Cuenta	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	Debe existir una nueva cuenta para el dominio en el servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de ver buzón 2. Accederá a la pantalla de nueva cuenta

	3. El sistema mostrará la pantalla con el formulario de crear cuentas. 4. El usuario rellenará los campos del formulario necesarios y los enviará 5. El sistema transformará los datos en un objeto Cuenta 6. El sistema transformara el objeto Cuenta en un archivo JSON y se lo pasará al script PHP. 7. El sistema invocará el método remoto para crear la cuenta en ISPConfig 8. Se enviará al usuario a la pantalla de ver buzones
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden guardar cuentas. ○ Se informa al usuario de que no puede crear nuevos dominios temporalmente.
Notas	-

5.2.4.3.15 Editar cuenta

Editar Cuenta	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	La información de la cuenta deberá encontrarse actualizada en el servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. Accederá a la pantalla de editar cuenta 2. El sistema carga los datos de la cuenta seleccionada. 3. El sistema mostrará la pantalla con el formulario de edición de cuentas. 4. El usuario rellenará los campos del formulario necesarios y los enviará. 5. El sistema transformará los datos en un objeto Cuenta. 6. El sistema transformará el objeto Cuenta en un fichero JSON. 7. El sistema invocará remotamente el método de actualización del servidor mediante un script en PHP. 8. Se enviará al usuario a la pantalla de ver buzones.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1**: El formulario está incompleto o no cumple los requisitos de validación ○ Se informa al usuario y se le da la posibilidad de rellenarlo nuevamente, guardando lo que introdujo anteriormente.
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden guardar cuentas. ○ Se informa al usuario de que no puede crear nuevos dominios temporalmente.

Notas	-

5.2.4.3.16 Ver cuentas

Ver cuentas	
Precondiciones	El usuario debe estar validado y debe contar con permisos de lectura en el módulo de Gestión de dominios.
Poscondiciones	Se le muestran al usuario todas las cuentas del buzón en el que se encuentre.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario accede a la página de ver buzón. 2. El sistema carga la información de las cuentas en el servidor ISPConfig. 3. El sistema muestra toda la información en una interfaz web
Excepciones	• **El servidor ISPConfig no está disponible**: No se pueden consultar los dominios ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	-

5.2.4.3.17 Eliminar cuenta

Eliminar cuenta	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	La cuenta desaparece del servidor ISPConfig.
Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de eliminar relativo a la cuenta que desea borrar 2. El sistema muestra un dialogo de aceptación 3. El usuario confirma la acción 4. El sistema invoca remotamente el método de eliminar cuenta del servidor. 5. El sistema muestra la página de ver buzones actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede eliminar el dominio ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	El cuadro de dialogo se generará con javascript.

5.2.4.3.18 Configurar autorespuesta

Configurar autorespuesta	
Precondiciones	El usuario debe estar validado y debe contar con permisos de escritura en el módulo de Gestión de dominios.
Poscondiciones	Se cambiar la configuración de autorespuesta de la cuenta indicada.

Actores	Iniciado y terminado por el usuario
Descripción	1. El usuario pulsa sobre el botón de activar/desactivar relativo a la cuenta. 2. El sistema le muestra un formulario con los datos de autorespuesta. 3. El sistema invoca remotamente el método de actualizar autorespuesta del servidor. 4. El sistema muestra la página de ver buzones actualizada
Excepciones	• **El servidor ISPConfig no está disponible**: No se puede activar el dominio ○ Se informa al usuario de que no puede realizar la operación temporalmente.
Notas	Debido a un error con la versión de ISPConfig la fecha de la autorespuesta se pone siempre a cero. http://bugtracker.ispconfig.org/index.php?do=details&task_id=2398

5.2.4.4 *Cliente*

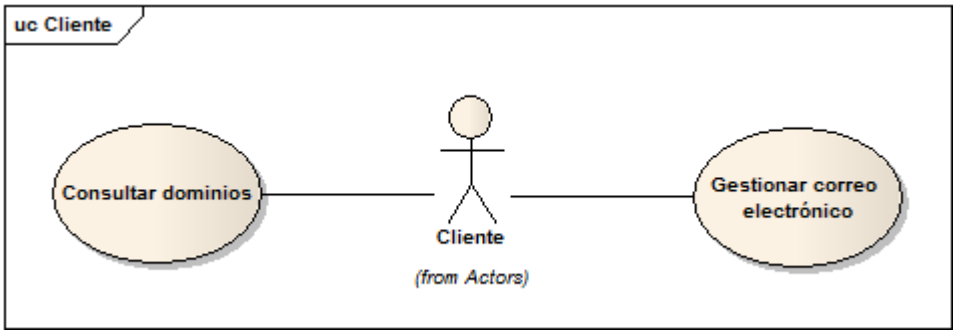


Diagrama 5.13: Casos de uso-Cliente

No se ve necesario la creación de tablas en este punto, pues las acciones de cliente ya se han explicado de forma general en los casos de uso del módulo de gestión de dominios.

Nombre del Caso de Uso
Consultar dominios
Descripción
Se le permitirá al cliente consultar los dominios que tenga asociados. Sin posibilidad de modificarlos o darlos de baja.

Nombre del Caso de Uso
Gestionar correo electrónico
Descripción
Se le permitirá al cliente gestionar su cuenta de correo, pudiendo habilitar la autorespuesta o consultar sus datos.

5.2.5 Identificación de los Subsistemas en la Fase de Análisis

5.2.5.1 *Descripción de los Subsistemas*

5.2.5.1.1 Intranet

Este será el subsistema principal de la intranet, será utilizado por todos los subsistemas desarrollados en este proyecto así como por los subsistemas desarrollados por otros, desde ella el usuario tendrá acceso a los módulos cuyos permisos le permitan y a ciertas acciones propias del usuario en sí.

5.2.5.1.2 Módulo de gestión de la base de datos

Este subsistema, será el encargado de realizar todas las operaciones necesarias con la base de datos, se conectará con el resto de los subsistemas de la aplicación, para que, mediante

instrucciones ofrecidas por ellos, pueda completar y dejar reflejados los cambios necesarios en la base de datos.

En un primer acercamiento, se utilizará un patrón DAO para el acceso a dicha base de datos, de manera que este subsistema sea lo más independiente posible del resto del proyecto.

5.2.5.1.3 Módulo de gestión de usuarios

Implicará todas las operaciones relacionadas con la gestión de usuarios, alta, baja, modificación… Además ofrecerá todos los servicios de búsqueda que tengan como objetivo obtener información sobre los clientes y lo relativo a permisos relacionados con perfiles de usuario y grupos.

5.2.5.1.4 Módulo de gestión de dominios

Esté módulo, englobará todas las operaciones relativas a la gestión de un dominio propiamente dicha, permitirá dar de alta nuevos dominios, vincularlos con clientes, gestión de los buzones de correo, etc.

5.2.5.1.4.1 *Módulo de gestión de correo electrónico*

Todas las operaciones de gestión del correo electrónico se incluirán en este subsistema, gestión de buzones, alta de nuevas cuentas, modificación de las mismas…

5.2.5.1.5 Módulo de interfaces

La aplicación constará con una serie de interfaces web, las cuales permitirán al usuario comunicarse con la lógica del sistema, cada módulo antes mencionado tendrá una seria de interfaces propias, que proporcionarán soporte para ejecutar toda la lógica de la aplicación.

Nuevamente, para intentar conseguir el máximo nivel posible de independencia entre la parte visual (interfaces) y la lógica del negocio se aplicará un modelo vista controlador, junto con una fachada de servicios.

5.2.5.2 *Descripción de los Interfaces entre Subsistemas*

La aplicación estará dividida en tres capas principales:

- **Capa de presentación:** Incluirá todas las interfaces de la aplicación (Módulo de interfaces) y se conectará con la capa de lógica mediante la utilización de Servlets, el usuario solo interactuará directamente con esta capa. Al ser interfaces web, no sé encontrará en la misma máquina que el resto de capas.
- **Capa de lógica:** Incluirá todos los módulos relacionados con la lógica del negocio, por encima tendrá siempre la capa de presentación y por debajo se conectará a la capa de persistencia (o módulo de gestión de la base de datos), está conexión se realizará por medio de JDBC, se encontrará situada en la misma máquina que la capa de presentación.
- **Capa de persistencia:** Será la capa más baja de la aplicación, y solo se podrá acceder a ella mediante la capa de lógica, como ya se ha dicho antes, está conexión se realizará por medio de JDBC.

5.2.5.3 *Diagrama de Clases*

5.2.5.3.1 **Intranet**

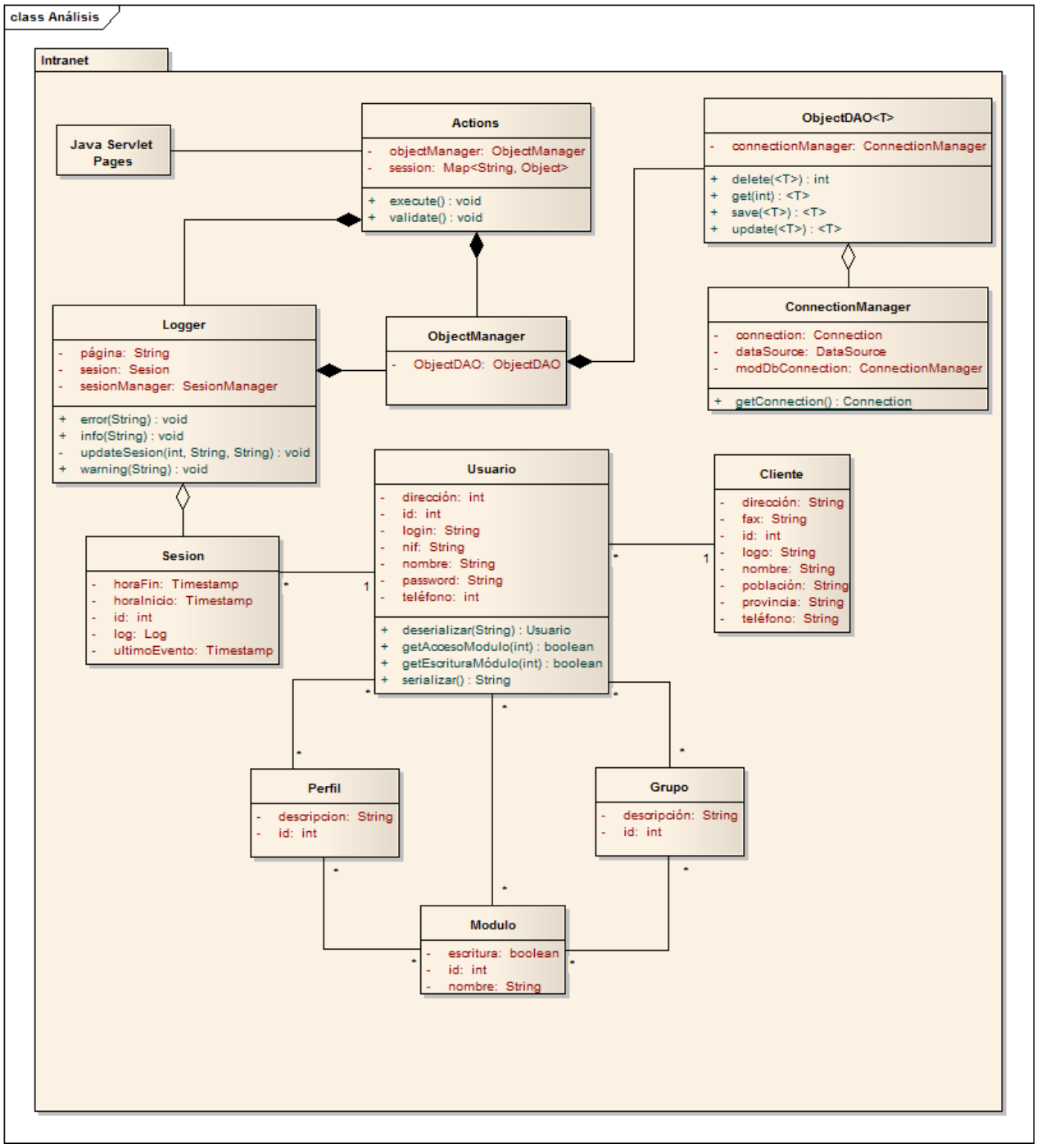


Diagrama 5.14: Clases-Intranet

Gracias al diagrama podemos ver a primera vista la estructura del módulo, distinguiéndose las JSPs (páginas encargadas de mostrar y recoger la información del usuario), los actions (clases encargadas de la adquisición y validación de los datos insertados en las JSPs y de transmitirlos a los managers), los managers (clases encargadas del tratamiento de la información de la comunicación con los DAO), los DAO (clases encargadas de la persistencia de los objetos) y finalmente las entidades como tal.

5.2.5.3.1.1 *Descripción de las clases*

Es importante tener en cuenta que se trata de un modelo simplificado para poder entender rápidamente la arquitectura del proyecto. Sin ir más lejos, se han combinado todos los actions en una misma clase por tener un funcionamiento similar, de igual manera se ha hecho lo propio con "ObjectManager" y "ObjectDAO", siendo "Object" una entidad concreta de la aplicación.

Nombre de la clase
Actions
Descripción
Clase que representa de manera genérica los diferentes Actions que se encargan de recoger las solicitudes del usuario y tramitarlas para pasarle las órdenes a la siguiente capa de la arquitectura, así como preparar la información para mostrarla en el navegador del usuario.
Responsabilidades
Recibir las peticiones del usuario y procesarlas para solicitar la información a la lógica de la aplicación. También valida que los datos introducida por el usuario sean correctos.
Atributos propuestos
• **objectManager:** Objeto genérico que representa a todos los Manager de los diferentes objetos que necesite cada Action, y se encarga de realizar las diferentes llamadas a la capa de persistencia para llevar a cabo las operaciones solicitadas por el usuario. • **session:** Contiene los datos de la sesión del usuario (incluida la identificación del mismo), y es utilizado también para transportar cuando sea necesario la información a las JavaServer Pages.
Métodos propuestos
• **execute():** Este método es el ejecutado por defecto cuando Struts llama al Action encargado de gestionar una acción del usuario concreta. Recoge la información y la prepara para transmitírsela a la siguiente capa de la arquitectura a través del objectManager. • **validate():** Este método, si declarado, es llamado antes del execute(), y su función es la de verificar si los datos introducidos por el usuario son adecuados y pueden ser utilizados.

Nombre de la clase
ObjectManager
Descripción
Clase que representa a todos los manejadores de los diferentes objetos del modelo de la aplicación que necesiten acceso a datos.
Responsabilidades
Representar un puente lógico entre la capa de presentación en la que el usuario solicita información y la capa de persistencia donde se obtiene. Contiene la lógica de negocio y prepara los datos para hacer las llamadas oportunas a los objetos DAO.

Atributos propuestos
• **objectDAO:** Objeto genérico que representa todos los DAO de los diferentes objetos que necesite cada ObjectManager.
Métodos propuestos
• **eliminar (int):** Recibe el identificador del objeto y hace la llamada correspondiente al método de eliminación del DAO a través del objectDAO. • **Obtener por id (int):** Recibe el identificador del objeto y hace la llamada correspondiente al método de lectura del DAO a través del objectDAO. • **insertar (Object):** Recibe el identificador del objeto y hace la llamada correspondiente al método de inserción del DAO a través del objectDAO. • **modificar (Object):** Recibe el objeto, con toda su información, y hace la llamada correspondiente al método de modificación del DAO a través del objectDAO.

Nombre de la clase
ObjectDAO
Descripción
Clase que representa de manera genérica a los diferentes objetos de acceso a datos que se comunican directamente con la base de datos.
Responsabilidades
Se encarga de implementar los métodos CRUD (crear, obtener, actualizar y borrar, en sus siglas en inglés) o aquellos que se consideren necesarios de acceso a la base de datos. Recibe la información con la que debe trabajar y la prepara para lectura o escritura en base de datos en función a los mismos.
Atributos propuestos
• **ConnectionManager:** Hace referencia al objeto encargado de gestionar las conexiones con la base de datos, o en este caso con el pool de conexiones.
Métodos propuestos
• **eliminar (int):** Recibe el identificador del objeto y realiza la consulta correspondiente a la base de datos. • **Obtener por id (int):** Recibe el identificador del objeto y realiza la consulta correspondiente a la base de datos. • **insertar (Object):** Recibe el identificador del objeto y realiza la consulta correspondiente a la base de datos. • **modificar (Object):** Recibe el objeto, con toda su información, y realiza la consulta correspondiente a la base de datos.

Nombre de la clase
ConnectionManager
Descripción
Clase que se encarga del manejo de las conexiones con la base de datos o el pool de datos en este caso.
Responsabilidades
Facilitar conexiones a los objetos DAO de manera que estos puedan interactuar con la base de datos.
Atributos propuestos
• **connection:** Es el objeto conexión que se devolverá a los DAOs cuando soliciten una.

- **dataSource:** Indica el origen de los datos, o en este caso el recurso de Tomcat en el que se especifican los detalles de conexión.
- **modDbConnection:** Es una referencia al propio objeto, para poder invocarlo estáticamente.

Métodos propuestos

- **getConnection():** Obtiene, si la hubiera una conexión del pool y en el caso de que no crea una nueva para devolverla.

Nombre de la clase
Logger

Descripción

Clase que se encarga del manejo de los logs de la aplicación.

Responsabilidades

Generar líneas de log para ser almacenadas en la base de datos vinculadas a una sesión concreta.

Atributos propuestos

- **página:** Indica desde que action ha sido llamado el logger, de manera que aparezca esa información en la base de datos.
- **session:** Indica la sesión de Tomcat y se utiliza principalmente para acceder a los datos del usuario.
- **sessionManager:** Es una referencia al manager de sesión, de manera que se pueda persistir la información del log.

Métodos propuestos

- **info(String):** Genera un log en modo información.
- **warning(String):** Genera un log en modo warning.
- **error(String):** Genera un log en modo error.
- **updateSession(int, String, String):** Genera y almacena un log basándose en el nivel (info, error, warning), el evento y el action en el que ha ocurrido.

Nombre de la clase
Usuario

Descripción

Clase que representa el usuario dentro de la aplicación

Responsabilidades

Contener la información relativa a usuarios y ofrecer métodos de lectura y escritura de sus atributos, además de permitir conocer sus permisos y serializarlo.

Atributos propuestos

- **id:** Identificador numérico del objeto en base de datos.
- **login:** Nombre de usuario en la aplicación, es único.
- **password:** Contraseña de usuario.
- **nombre:** Nombre completo de la persona vinculada al usuario.
- **nif:** Documento de identificación vinculado al usuario.
- **dirección:** Dirección vinculada al usuario.
- **teléfono:** Número de teléfono vinculado al usuario.

Métodos propuestos

- **serializar():** Convierte el objeto a una cadena de bytes de manera que pueda ser

almacenado como tal, utiliza codificación en base64 para llevar a cabo la serialización.
- **deserializar(String):** A partir de una cadena intenta obtener el objeto Usuario correspondiente.
- **getAccesoModulo(int):** Indica si un usuario tiene o no permisos de lectura para un módulo dado.
- **getEscrituraModulo(int):** Indica si un usuario tiene o no permisos de escritura para un módulo dado.

Nombre de la clase
Sesión
Descripción
Clase que representa una sesión simulada para el usuario dentro de la aplicación
Responsabilidades
Contener la información relativa a sesiones y ofrecer métodos de lectura y escritura de sus atributos. Además consta de una clase privada Log.
Atributos propuestos
• **id:** identificador numérico de la sesión en base de datos. • **horaFin:** Indica la fecha y hora en la que se terminó la sesión. • **horaInicio:** Indica la hora en la que se inició la sesión. • **log:** Contiene la información del evento, página y nivel que se quiere guardar en base de datos. • **ultimoEvento:** indica la hora a la que se realizó el último evento dentro de la sesión.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Cliente
Descripción
Clase que representa un cliente dentro de la aplicación
Responsabilidades
Contener la información relativa a clientes y ofrecer métodos de lectura y escritura de sus atributos.
Atributos propuestos
• **id:** Identificador numérico del cliente dentro de la base de datos. • **nombre:** Nombre completo del cliente. • **dirección:** Dirección del cliente. • **población:** Población del cliente. • **provincia:** Provincia del cliente. • **teléfono:** Teléfono del cliente. • **fax:** Fax del cliente. • **logo:** Referencia a una imagen que sirve como logo para el cliente.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Perfil
Descripción

Clase que representa un perfil dentro de la aplicación
Responsabilidades
Contener información de acceso a módulos y ser implementados por los usuarios.
Atributos propuestos
• **id:** Identificador numérico del perfil dentro de la base de datos. • **descripción:** Pequeña descripción o nombre que recibe el perfil.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Grupos
Descripción
Clase que representa un grupo dentro de la aplicación
Responsabilidades
Asociar distintos usuarios en grupos para su gestión
Atributos propuestos
• **id:** Identificador numérico del perfil dentro de la base de datos. • **descripción:** Pequeña descripción o nombre que recibe el perfil.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Modulo
Descripción
Clase que representa un módulo dentro de la aplicación
Responsabilidades
Contener la información de permisos relativa a un usuario o perfil para los distintos módulos
Atributos propuestos
• **id:** Identificador numérico del módulo. • **Escritura:** Indica si se tiene o no permisos de escritura. • **Nombre:** Nombre del módulo al que hace referencia.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

5.2.5.3.2 Módulo de gestión de usuarios

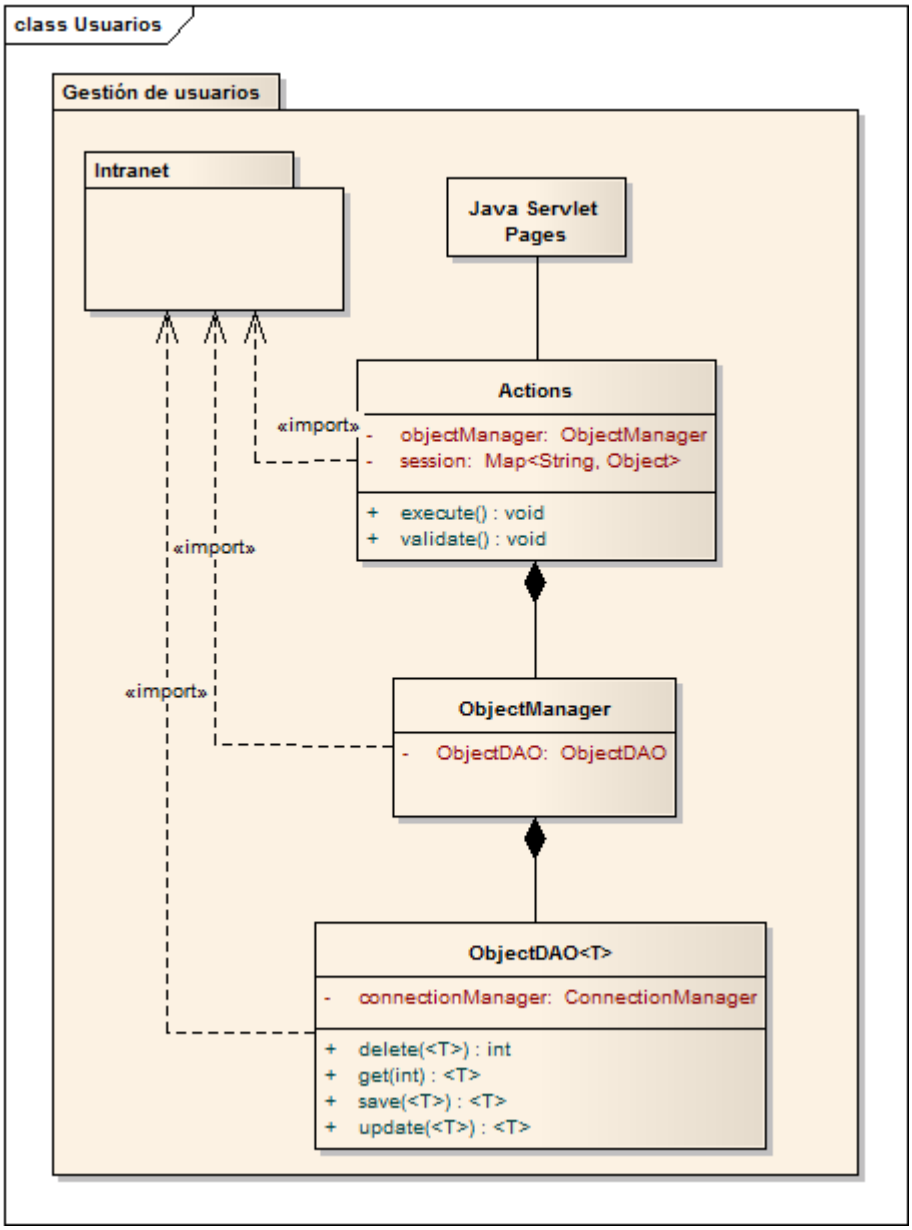


Diagrama 5.15: Clases-Módulo de usuarios

Similar al diagrama anterior debido a la utilización casi por completo de las clases implementadas en el módulo Intranet.

5.2.5.3.2.1 *Descripción de las clases*

Al ser las mismas clases que en el caso anterior no se describirán nuevamente.

5.2.5.3.3 Módulo de gestión de dominios

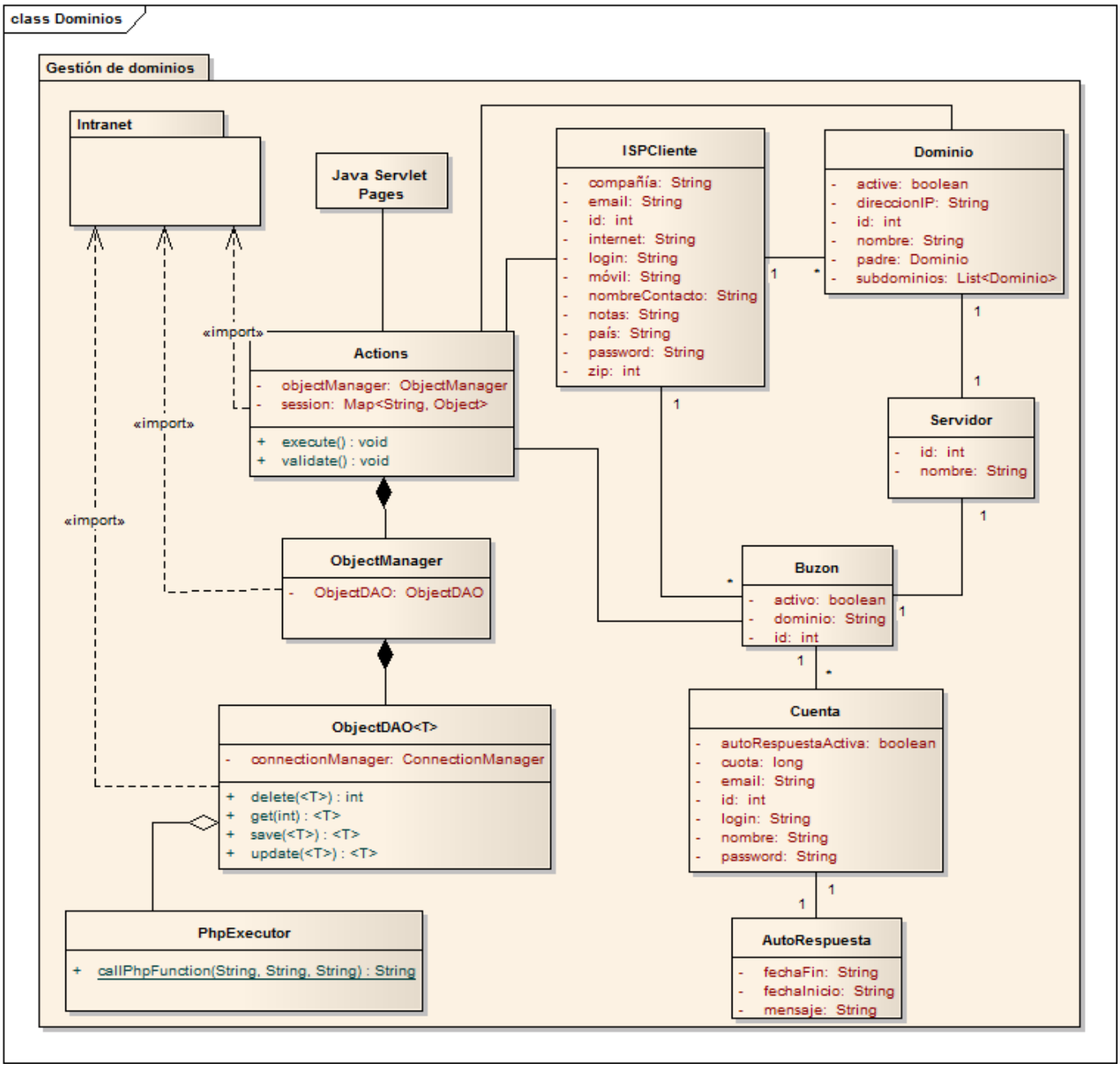


Diagrama 5.16: Clases-Módulo gestión de dominios

Nuevamente la arquitectura que se aprecia es la misma que en el primer diagrama de este apartado y por lo tanto no se hará especial mención sobre ella nuevamente.

5.2.5.3.3.1 *Descripción de las clases*

Muchas de las clases del diagrama coinciden con las del primer diagrama y por lo tanto no se explicarán otra vez.

Nombre de la clase
ISPCliente
Descripción
Clase que representa un cliente del servidor ISP dentro de la aplicación

Responsabilidades
Contener información del cliente.
Atributos propuestos
• **id:** Identificador numérico del cliente dentro del servidor ISP. • **Login:** Nombre de usuario del cliente para el servidor ISP. • **Password:** Contraseña de usuario del cliente para el servidor ISP. • **nombreContacto:** Nombre real de la persona asociada al cliente. • **compañía:** Compañía a la que pertenece el cliente. • **zip:** Código postal relacionado con el cliente. • **país:** País de residencia del cliente. • **email:** Email de contacto del cliente. • **móvil:** Teléfono de contacto del cliente. • **internet:** Página web asociada al cliente o compañía. • **notas:** Notas relativas al cliente.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Servidor
Descripción
Clase que representa un servidor dentro de la aplicación
Responsabilidades
Contener información de servidor y ser utilizado por las clases dominio y buzón para indicar donde se encuentran.
Atributos propuestos
• **id:** Identificador numérico del servidor dentro de ISPConfig. • **nombre:** Nombre del servidor.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Dominio
Descripción
Clase que representa un dominio dentro de la aplicación
Responsabilidades
Contener información de acceso a módulos y ser implementados por los usuarios.
Atributos propuestos
• **id:** Identificador numérico del dominio dentro de ISPConfig. • **Nombre:** DNS del dominio. • **direcciónIP:** Dirección IP contra la que se puede resolver el dominio. • **Active:** Indica el estado del dominio dentro del servidor ISP. • **Padre:** En el caso de existir, indica el dominio padre del dominio. • **Subdominios:** En el caso de haberlos, indica los dominios que descienden de este.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Buzón
Descripción
Clase que representa un buzón dentro de la aplicación.
Responsabilidades
Contener información relativa a los buzones.
Atributos propuestos
• **id:** Identificador numérico del buzón dentro del servidor ISP. • **dominio:** Dominio para el cual se ha creado el buzón. • **Activo:** Indica el estado del buzón dentro del servidor ISP.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Cuenta
Descripción
Clase que representa una cuenta de correo dentro de la aplicación.
Responsabilidades
Contener información de cuentas de correo electrónico.
Atributos propuestos
• **id:** Identificador numérico de la cuenta dentro del servidor ISP. • **Nombre:** Nombre del usuario de la cuenta. • **Login:** Nombre de usuario para acceder a la cuenta de correo. • **Password:** Contraseña de acceso para la cuenta de correo. • **Email:** Es lo mismo que el login. • **Cuota:** Espacio en megabytes asignado a la cuenta de correo.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
Autorespuesta
Descripción
Clase que representa una respuesta automática dentro de la aplicación.
Responsabilidades
Contener información de auto respuesta y ser implementada por las cuentas.
Atributos propuestos
• **fechaFin:** Indica la fecha de finalización de la autorespuesta configurada. • **fechaInicio:** Indica la fecha de inicio de la autorespuesta configurada. • **Mensaje:** Indica el mensaje que será enviado como autorespuesta ante la llegada de correos electrónicos.
Métodos propuestos
Ninguno a destacar más allá de los “getters” y “setters” propios de un bean.

Nombre de la clase
PhpExecutor
Descripción

Clase utilizada para realizar la invocación de scripts php.
Responsabilidades
Invocar los scripts php y asegurarse de que el retorno sea válido.
Atributos propuestos
-
Métodos propuestos
• **callPhpFunction(String, String, String):** Método que recibe el path de los ficheros Php, el nombre del fichero y los parámetros de invocación del script necesarios.

5.2.6 Análisis de Interfaces de Usuario

La interfaz de usuario para la aplicación será desarrollada por parte de un diseñador de la empresa, la manera de conseguir dichas interfaces es proporcionándole un pequeño esqueleto de lo que contendrá la interfaz (formularios, tablas, etc.). Para lo cual el desarrollará una hoja de estilos que deberá ser adaptada posteriormente en la aplicación.

5.2.6.1.1 Descripción de la interfaz

Interfaz 5.1: Descripción de la interfaz

Como vemos, está será la interfaz final de las pantallas de la aplicación, en ella podemos distinguir las siguientes zonas:

- **Cabecera:** Parte superior de la interfaz, muestra el logo de la empresa
  - **Barra de sesión:** Muestra el usuario conectado y permite cerrar la sesión.
  - **Menú de navegación:** Muestra los distintos módulos a los que podrá navegarse.
- **Contenido:** Parte principal de la interfaz, muestra el contenido de las jsps que queramos cargar
  - **Barra de herramientas:** Muestra el menú en el que nos encontramos y botones que permitan ejecutar acciones sobre el mismo.
  - **Barra de búsqueda:** Ofrecerá opciones de búsqueda dentro del menú.
- **Pie:** Muestra información de derechos sobre la página y la empresa.

5.2.6.1.2 Descripción del comportamiento de la interfaz

La interfaz tendrá un comportamiento reseñable, debido a la utilización de tiles para la generación de las distintas páginas a través de unas plantillas.

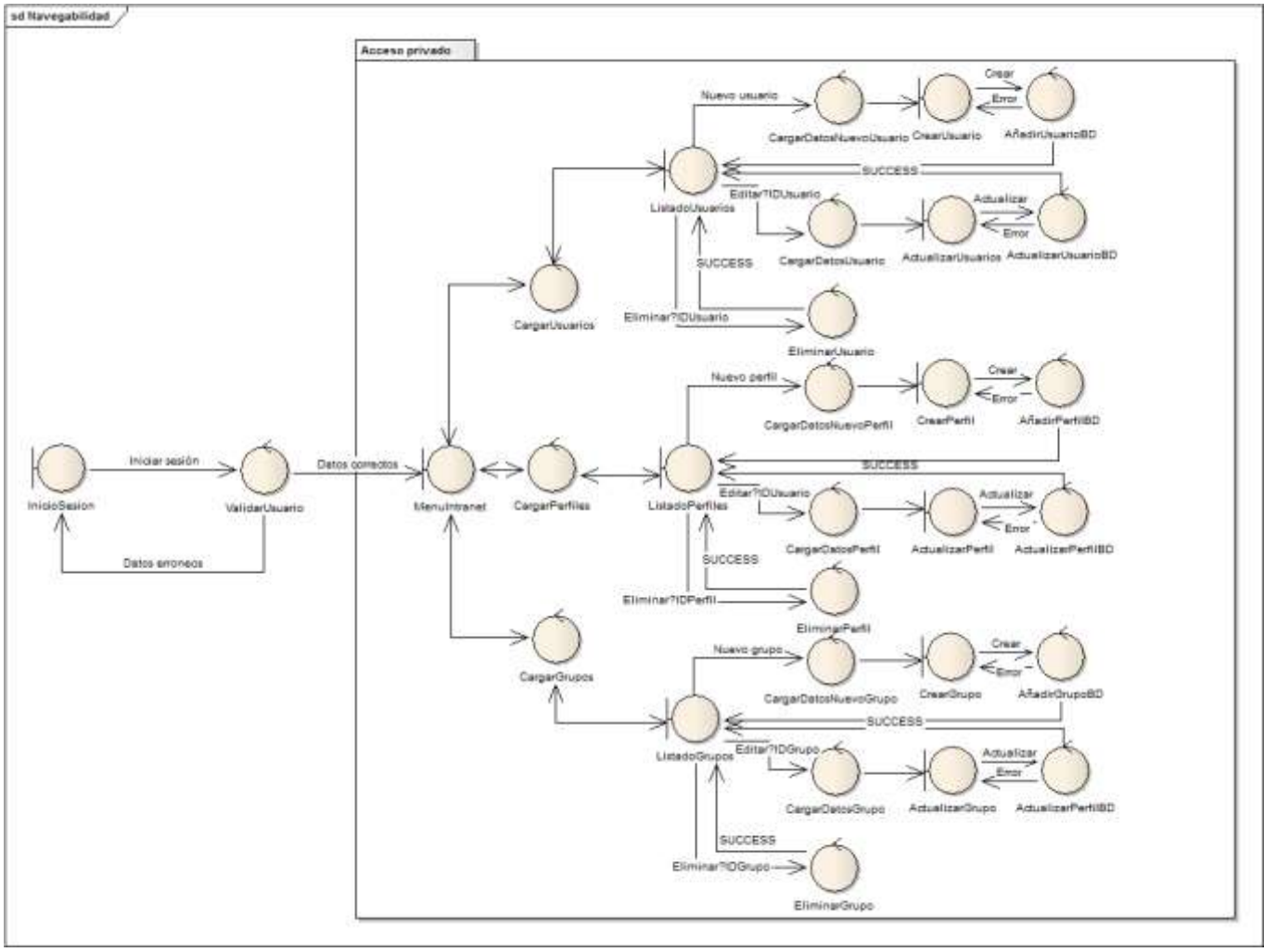
5.2.6.1.2.1 *Generación mediante tiles*

Tiles será utilizado para generar la parte de la interfaz que sea común a varias páginas, por ello lo primero que se define con Tiles es la estructura de "encabezado", "cuerpo" y "pie" de página que se puede observar en las imágenes del punto anterior. Dado que esto será común a todas las páginas de la aplicación será considerada la plantilla básica.

Una vez que nos encontremos identificados en la aplicación, aparecen dos nuevos elementos que serán el menú de navegación entre módulos (como parte del "encabezado"), la barra de herramientas y las pestañas de navegación entre menús (dentro del cuerpo).

Cada vez que se navegue entre módulos, se utilizará tiles para la generación de la página que servirá como interfaz para dicho módulo, facilitándole en un archivo de configuración los parámetros correspondientes y recargando por completo la página.

5.2.6.1.3 Diagrama de navegabilidad



5.2.7 Especificación del plan de pruebas

En esta sección aparece desglosado el plan de pruebas a tener en cuenta durante el desarrollo del proyecto. Para cada tarea se especifican los diferentes resultados que pueden darse y el resultado esperado para cada uno.

5.2.7.1 *Pruebas unitarias*

Debido a la naturaleza de la arquitectura modelo, vista, controlador. Solo tiene sentido desarrollar el plan de pruebas para las clases del modelo, se realizarán por lo tanto pruebas sobre las clases que producen de una u otra manera comunicación con la base de datos, puesto que el resto de la arquitectura supone solo llamadas a niveles superiores que acaban llegando a estas clases.

Todas las acciones especificadas en las pruebas deben producir salidas de log en la base de datos.

5.2.7.1.1 Intranet

5.2.7.1.1.1 *Sistema de inicio de sesión*

Prueba	Resultado esperado
Iniciar sesión con un usuario válido.	El usuario aparece logeado en la aplicación.
Iniciar sesión un usuario no válido.	Se devuelve al usuario a la pantalla de login y se le informa de que las credenciales no son válidas.
Iniciar sesión con una contraseña equivocada.	Se devuelve al usuario a la pantalla de login y se le informa de que las credenciales no son válidas.
Acceder a páginas sin haber iniciado sesión.	Se intercepta al usuario y se le dirige a la pantalla de inicio de sesión.

5.2.7.1.1.2 *Sistema de cierre de sesión*

Prueba	Resultado esperado
Cerrar sesión con un usuario identificado.	Se cierra totalmente la sesión del usuario y se le expulsa de la aplicación.

5.2.7.1.2 Módulo de usuarios

5.2.7.1.2.1 *Sistema de usuarios*

Prueba	Resultado esperado
Crear usuario nuevo correctamente.	Aparece un nuevo usuario en la base de datos del sistema con los datos especificados por el usuario.
Crear usuario nuevo rellenando los datos de manera incorrecta.	Se indica al usuario que campo ha rellenado mal y se le permite volver a rellenarlo.
Intentar crear un usuario ya existente.	Se le indica al usuario que el usuario que intenta crear ya existe en la base de datos.
Cancelar la creación de nuevo usuario.	Se devuelve al usuario a la página anterior y no se producen cambios en la base de datos.

Obtener usuario por identificador.	Se recupera el objeto usuario, así como sus detalles asociados.
Añadir nuevo perfil al usuario.	Se añade la referencia en la tabla de relación usuarios perfiles para hacer esto efectivo.
Añadir nuevo grupo al usuario.	Se añade la referencia en la tabla de relación usuarios grupos para hacer esto efectivo.
Eliminar usuario existente.	Se marca el usuario como eliminado en la base de datos.
Eliminar usuario no existente.	El sistema detecta que el usuario no existe no produciéndose cambios en la base de datos.

5.2.7.1.2.2 *Sistema de perfiles*

Prueba	Resultado esperado
Crear perfil nuevo correctamente.	Aparece un nuevo perfil en la base de datos del sistema con los datos especificados por el usuario.
Crear perfil nuevo rellenando los datos de manera incorrecta.	Se indica al usuario que campo ha rellenado mal y se le permite volver a rellenarlo.
Intentar crear un perfil ya existente.	Se le indica al usuario que el perfil que intenta crear ya existe en la base de datos.
Cancelar la creación de nuevo perfil.	Se devuelve al usuario a la página anterior y no se producen cambios en la base de datos.
Obtener perfil por identificador.	Se recupera el objeto perfil, así como sus detalles asociados.
Añadir nuevo usuario al perfil.	Se añade la referencia en la tabla de relación usuarios perfiles para hacer esto efectivo.
Eliminar perfil existente.	Se marca el usuario como eliminado en la base de datos.
Eliminar perfil no existente.	El sistema detecta que el perfil no existe no produciéndose cambios en la base de datos.

5.2.7.1.2.3 *Sistema de grupos*

Prueba	Resultado esperado
Crear grupo nuevo correctamente.	Aparece un nuevo grupo en la base de datos del sistema con los datos especificados por el usuario.
Crear grupo nuevo rellenando los datos de manera incorrecta.	Se indica al usuario que campo ha rellenado mal y se le permite volver a rellenarlo.
Intentar crear un grupo ya existente.	Se le indica al usuario que el grupo que intenta crear ya existe en la base de datos.
Cancelar la creación de nuevo grupo.	Se devuelve al usuario a la página anterior y no se producen cambios en la base de datos.
Obtener grupo por identificador.	Se recupera el objeto grupo, así como sus detalles asociados.
Añadir nuevo usuario al grupo.	Se añade la referencia en la tabla de relación usuarios grupos para hacer esto efectivo.
Eliminar grupo existente.	Se marca el grupo como eliminado en la base de datos.
Eliminar grupo no existente.	El sistema detecta que el grupo no existe no produciéndose cambios en la base de datos.

5.2.7.1.3 Módulo de dominios

5.2.7.1.3.1 *Sistema de clientes*

Prueba	Resultado esperado
Crear cliente ISP nuevo con cliente y usuario.	Aparece un nuevo usuario, un nuevo cliente y una relación cliente-cliente ISP en la base de datos, así como un nuevo usuario en el servidor ISP.
Crear cliente ISP nuevo con usuario ya existente.	Si el usuario tiene un cliente vinculado se crea solo la relación cliente-cliente ISP y un usuario nuevo en el servidor ISP. Si no tiene el cliente vinculado se crea un nuevo cliente en la BD además de lo anterior.
Crear cliente ISP nuevo con cliente ya existente.	Se crea un nuevo usuario y la relación cliente-cliente ISP en la base de datos, finalmente se crea el usuario en el servidor ISP.
Intentar crear un usuario ya existente.	El sistema lo detecta e informa al usuario del problema.
Cancelar la creación de un nuevo cliente.	Se devuelve al usuario a la página anterior y no se producen cambios en ninguno de los sistemas.
Obtener cliente por identificador.	Se recupera el objeto cliente, así como sus detalles asociados.
Eliminar cliente existente.	Se elimina el usuario del servidor ISP y la relación cliente-clienteISP de la base de datos.
Eliminar cliente no existente.	El sistema detecta que el cliente no existe no produciéndose cambios en la base de datos.

5.2.7.1.3.2 *Sistema de dominios*

Prueba	Resultado esperado
Crear dominio nuevo.	Aparece un nuevo dominio en el servidor ISP relacionado con el cliente y servidor indicados.
Crear subdominio nuevo.	Aparece un nuevo dominio en el servidor ISP relacionado con el cliente, servidor y dominio padre indicados.
Intentar crear un dominio existente.	El sistema lo detecta e informa al usuario del problema.
Cancelar la creación de un nuevo dominio.	Se devuelve al usuario a la página anterior y no se producen cambios en ninguno de los sistemas.
Obtener dominio por identificador.	Se recupera el objeto dominio, así como sus detalles asociados.
Eliminar dominio existente.	Se elimina el dominio del servidor ISP.
Eliminar dominio no existente.	El sistema detecta que el dominio no existe no produciéndose cambios en el servidor.

5.2.7.1.3.3 *Sistema de buzones*

Prueba	Resultado esperado
Crear buzón nuevo.	Aparece un nuevo buzón relacionado con el servidor y el cliente indicados.
Crear cuenta nueva.	Aparece una cuenta nueva en el servidor

	relacionada con el buzón sobre el que se creó y con los datos indicados por el usuario.
Intentar crear un buzón existente.	El sistema lo detecta e informa al usuario del problema.
Cancelar la creación de una cuenta nueva.	Se devuelve al usuario a la página anterior y no se producen cambios en ninguno de los sistemas.
Obtener buzón por identificador.	Se recupera el objeto buzón, así como sus detalles asociados.
Obtener cuenta por identificador.	Se recupera el objeto cuenta, así como sus detalles asociados.
Obtener cuenta por email.	Se recupera el objeto cuenta, así como sus detalles asociados.
Eliminar buzón existente.	Se elimina el buzón y las cuentas asociadas del servidor ISP.
Eliminar cuenta existente.	Se elimina la cuenta del buzón asociado.
Eliminar buzón no existente.	El sistema detecta que el buzón no existe no produciéndose cambios en el servidor.

5.2.7.2 *Pruebas de integración*

5.2.7.2.1 Integración con Intranet

La intranet debe proporcionar mecanismos de integración sencillos para que el resto de módulos puedan funcionar sin suponer demasiado problema al desarrollador, para ello se seguirán varias pruebas de integración que deberán ser superadas con motivo de solucionar este problema.

Prueba	Resultado esperado
Guardar la información de usuario entre aplicaciones.	El usuario puede cambiar de módulo libremente una vez iniciada sesión sin necesidad de volver a indicar credenciales.
Eliminar la información de usuario al cerrar sesión.	Cuando el usuario cierra sesión su información se elimina del navegador y no puede acceder a ninguna de las aplicaciones.
Eliminar la información del usuario al cerrar el navegador.	Cuando el usuario vuelve a abrir el navegador se le pide volver a identificarse en el sistema para poder navegar por él.
Usuario con permisos de lectura intenta acceder.	Se lanza un interceptor que comprueba que los permisos son correctos para el módulo al que intenta acceder y se le permite el acceso.
Usuario sin permisos de lectura intenta acceder.	Se lanza un interceptor que comprueba los permisos, como no son correctos se le redirige a una página informativa explicándole la situación.
Usuario con permisos de escritura intenta modificar datos.	Se lanza un interceptor que comprueba los permisos, en el caso de ser correctos se transcurre con la modificación.
Usuario sin permisos de escritura intenta modificar datos.	Se lanza un interceptor que comprueba los permisos, en el caso de no ser correctos se redirige al usuario a una página informativa explicándole la situación.

Otro módulo utiliza el sistema de log.	Aparecerá en base de datos información de log introducida por otros módulos.
Otro módulo utiliza el manejador de conexiones.	Los otros módulos son capaces de conectarse con la base de datos de manera adecuada a través del pool.

5.2.7.3 *Pruebas del sistema*

Este tipo de pruebas comprueban la buena construcción de los mecanismos que comunican el conjunto de todo el sistema con los otros sistemas externos.

Prueba	Resultado esperado
Comunicar con el servidor de base de datos.	La base de datos responde correctamente a la orden SQL ejecutada.
Comunicar con el servidor ISP Config.	Existen comunicación entre ambas partes y el resultado devuelto por el servidor es el esperado.

5.2.7.4 *Pruebas de usabilidad*

Como ya ha sido comentado en apartados anteriores, el cliente ha sido el responsable de diseñar las interfaces de usuario del proyecto. Debido a lo cual, y considerando que el cliente ha estudiado los aspectos relativos a esto, no se han realizado estudios sobre la satisfacción del cliente con la usabilidad del proyecto.

Capítulo 6. Diseño

6.1 Arquitectura del sistema

6.1.1 Diagrama de paquetes

Debido a la gran cantidad de clases que conlleva una aplicación desarrollada con Struts y siguiendo el modelo vista controlador, es necesario un diagrama de paquetes que agrupe de manera lógica bloques de funcionalidad similar.

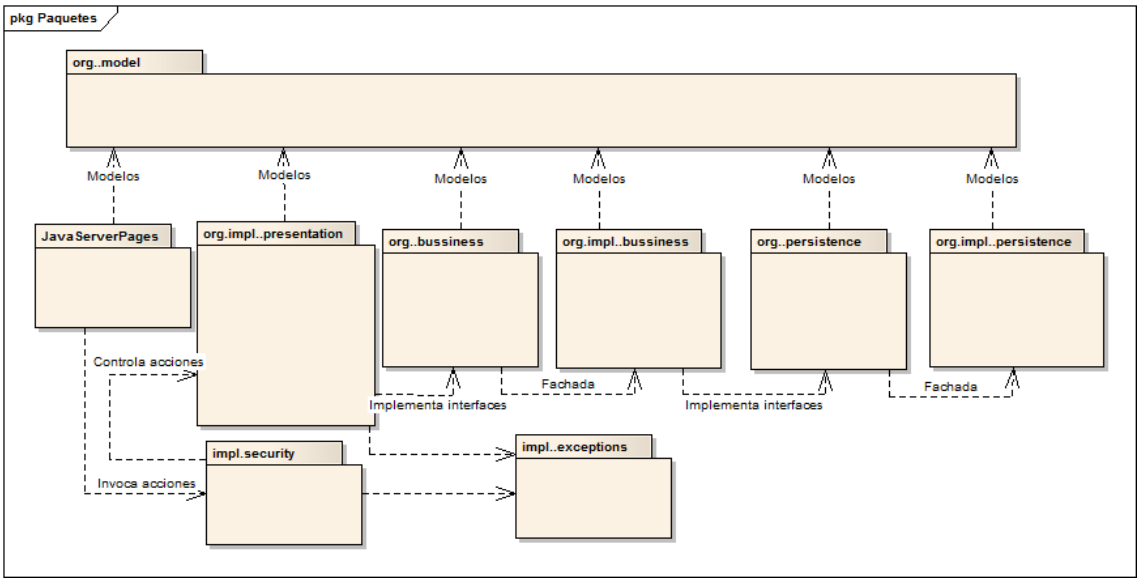


Diagrama 6.1: Paquetes

6.1.1.1 *Descripción de los paquetes*

6.1.1.1.1 Org.eco.mvc.model

Este paquete contiene todas las clases Java que sirven como modelo de un objeto POJO, se utilizan para realizar el modelaje de objetos con diferentes atributos, de manera que sea más sencillo trabajar con ellos y enviarlos entre distintos paquetes.

6.1.1.1.2 JavaServerPages

Aunque no se trata de un paquete de clases como tal, es interesante incluirlo en el diagrama para facilitar la visión de la infraestructura, en este diagrama el paquete sería la carpeta desplegada en el servidor "WEB-INF". Desde las webs de este paquete, se hacen llamadas a clases "actions" situadas dentro del paquete de presentación, todas estas llamadas pasan primero por los interceptores situados en el paquete de seguridad.

6.1.1.1.3 Org.eco.mvc.impl.security

Dentro de este paquete se encuentran situados los distintos interceptores del módulo, estos interceptores son invocados según sean requeridos al hacer llamadas sobre los "actions" del módulo.

6.1.1.1.4 Org.eco.mvc.impl.presentation

El contenido de este paquete son todas las clases "actions" que se ejecutan desde las distintas páginas web. Como el usuario tiene interacción casi directa con ellos, se sitúan como una capa de presentación. Las clases situadas dentro de este paquete se comunicarán con las interfaces de negocio requeridas.

6.1.1.1.5 Org.eco.mvc.bussiness

Este paquete está completamente formado por interfaces que serán implementadas por otras clases dentro del paquete de implementación, se utilizan para separar la lógica, de manera que si la clase encargada de implementarla se modifica no suponga un cambio a la capa de presentación y además nos permite tener varias implementaciones de la interfaz distintas y poder combinarlas según sea necesario.

6.1.1.1.6 Org.eco.mvc.impl.bussiness

Aquí se sitúan las distintas implementaciones de las interfaces antes mencionadas, cada implementación puede tener una lógica distinta y esto no afectará a la capa de presentación correspondiente que se comunicará con ellas mediante su interfaz (o fachada).

6.1.1.1.7 Org.eco.mvc.persistence

Al igual que ocurría con la capa de negocio, existen unas interfaces para la capa de persistencia, de manera que por ejemplo si algún día se quiere implementar una nueva capa usando otra tecnología de acceso a datos como hibernate no se requiera modificar las capas superiores.

6.1.1.1.8 Org.eco.mvc.impl.persistence

Finalmente, este paquete es el que incluye todas las clases DAO (Data Access Object) de manera que es el único que se comunica totalmente con la base de datos.

6.1.2 Diagrama de componentes

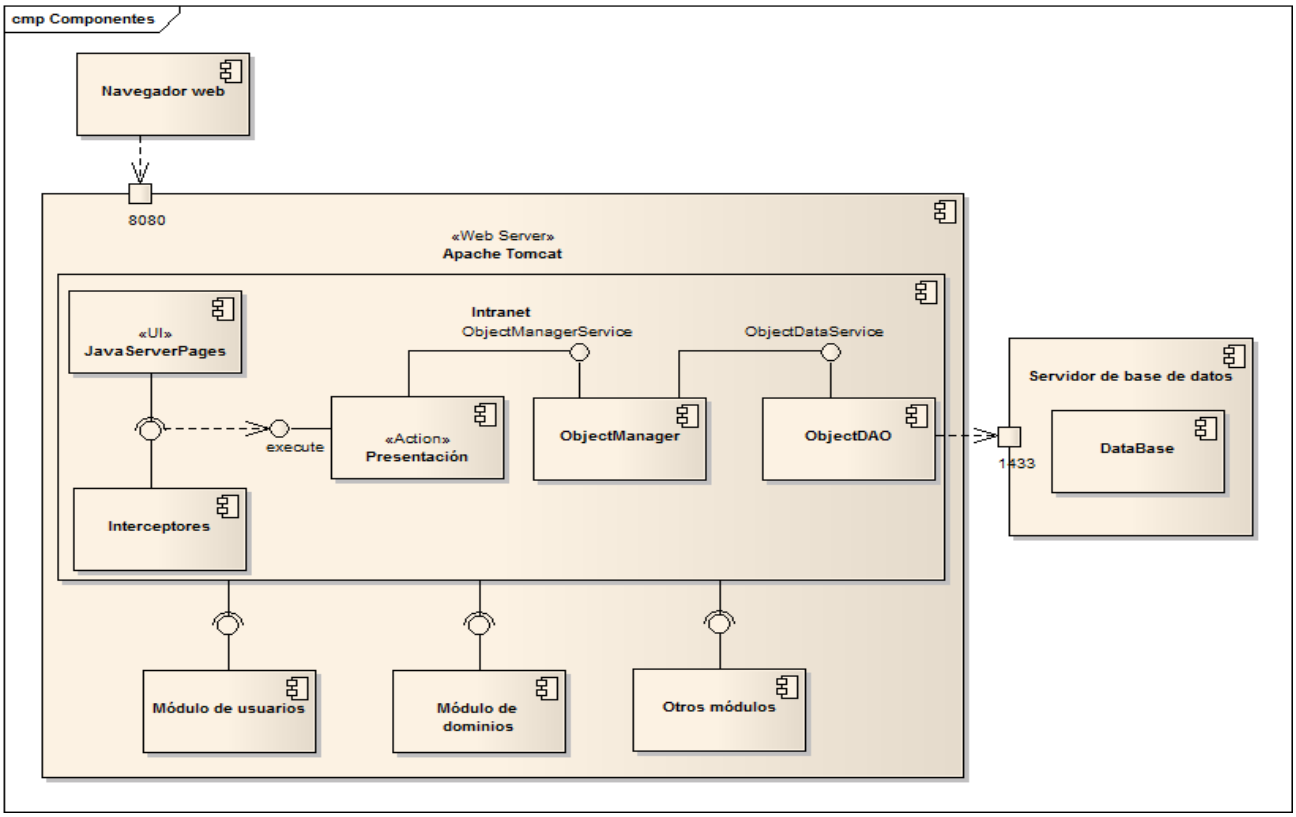


Diagrama 6.2: Componentes

6.1.2.1 *Descripción de los componentes*

6.1.2.1.1 Navegador Web

Hace referencia al navegador que el usuario utilizará para conectarse con el sistema, esté componente mostrará los archivos JSP de manera legible para el usuario.

6.1.2.1.2 Servidor Web

Es el servidor de aplicaciones encargado de servir y ejecutar el contenido de las JSPs así como las llamadas que estos hacen a los "actions".

6.1.2.1.2.1 *Intranet*

Es el módulo principal de todo el servidor, funciona de manera autónoma y en forma de librería para todos los demás módulos que aprovechan funciones del mismo, como el acceso a base de datos, el sistema de log, los interceptores…

6.1.2.1.2.1.1 JavaServerPages

Simulan las distintas JSPs a las que accederá el usuario y con las que interactuará, es decir las interfaces de usuario. Desde ellas se realizan llamadas al método "execute" de los distintos "actions", pero pasando antes por los interceptores.

6.1.2.1.2.1.2 Interceptores

Son clases que extienden de la clase “AbstractInterceptor”, según un archivo de configuración se activan o no al realizar llamadas a los distintos “actions”.

6.1.2.1.2.1.3 Presentation

Engloba todos los “actions” del módulo, se activan mediante una llamada a su método “execute”, siendo este el que presentan como interfaz con el exterior por defecto. Las distintas clases de este componente implementarán las interfaces que requieran para su funcionamiento.

6.1.2.1.2.1.4 ObjectManager

Es la clase de lógica asignada al manejo de los objetos, presenta una interfaz “ObjectManagerService” para ser implementada, a su vez implementa las interfaces correspondientes para el acceso a los objetos DAO que necesite para funcionar.

6.1.2.1.2.1.5 ObjectDAO

Clase de acceso a datos de los objetos, se comunica con base de datos mediante JDBC y ofrece una interfaz ObjectDataService.

6.1.2.1.3 Servidor de base de datos

Contiene la base de datos con la que se comunica el sistema mediante el pool de conexiones del servidor.

6.1.3 Diagrama de despliegue

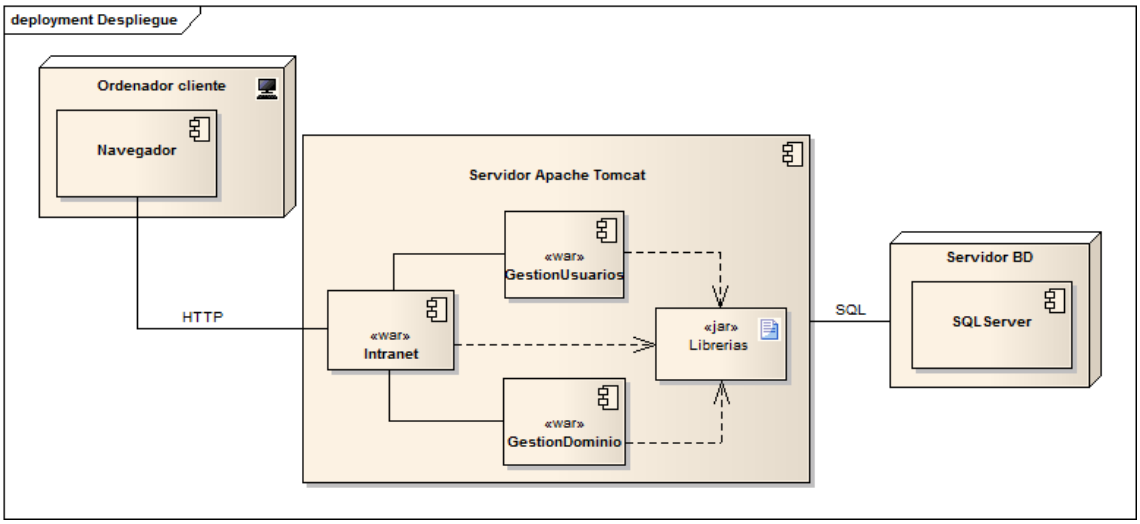


Diagrama 6.3: Despliegue

6.1.3.1 *Ordenador cliente*

Representa cualquier máquina física o virtual que se utilice para acceder a la interfaz web del módulo (contenido en la Intranet). Dentro se encuentra el navegador que deberá utilizarse para realizar la conexión con el servidor de aplicaciones web.

6.1.3.2 *Servidor Apache Tomcat*

Será un servidor dentro de la empresa en el que se dispondrá de un servidor de aplicaciones web Tomcat. Esté servidor alojará varias aplicaciones distintas:

- **Intranet.war:** Aplicación independiente que proporciona el soporte básico para el inicio de sesión.
- **GestionUsuario.war:** Módulo que se encarga del control total de los usuarios, perfiles, grupos y servicios.
- **GestionDominios.war:** Módulo que se encarga del control total de los dominios, clientes con dominios asociados, buzones de correo y cuentas.
- **Librerias.jar:** Este componente representa todas las librerías situadas dentro del servidor, entre ellas se encuentra el módulo Intranet para poder utilizar sus componentes importándolos, así como todas las librerías de Struts comunes a todos los proyectos para evitar que los proyectos desplegados tengan un peso significativo.

Queda recordar que dentro del servidor se encuentran otros módulos desarrollados por compañeros los aprovechan la configuración y la librería Intranet desarrollada en este proyecto.

6.1.3.3 *Servidor BD*

De manera análoga, tendremos un servidor dentro de la empresa con la base de datos montada, esté servidor podrá ofrecer otro tipo de servicios ya que no tiene por qué estar dedicado totalmente a la base de datos ni a la Intranet.

6.2 Diseño de clases

6.2.1 Diagramas de Clases

Como ya hemos visto en secciones anteriores, el proyecto se divide en tres módulos principales con distinta funcionalidad. Al ser independientes entre sí, se detallarán sus diagramas de manera separada.

Es necesario recordar el esquema general de clases presentado en el diagrama preliminar de análisis, como en el diagrama de paquetes del diseño. Esto es debido a que se segmentarán los diagramas de clases en paquetes para facilitar su comprensión.

6.2.1.1 *Intranet*

6.2.1.1.1 Paquete org.model

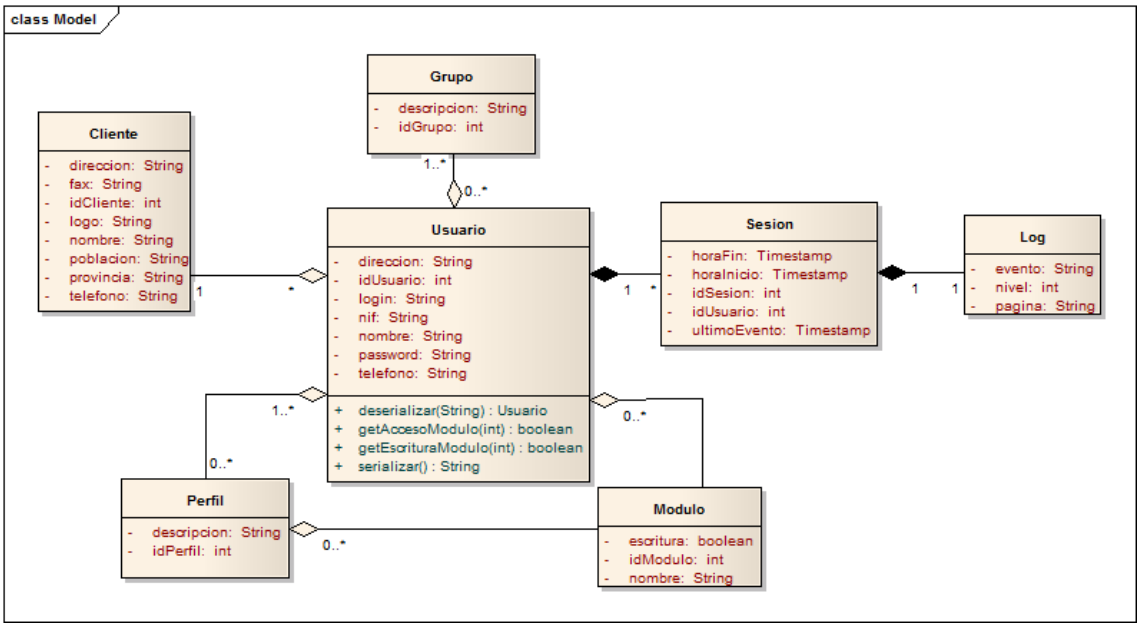


Diagrama 6.4: Clases-Modelo intranet

Clases que representan el modelo del dominio del módulo Intranet. Son en su mayoría JavaBeans, pues aparte de sus atributos y correspondientes “getters” y “setters” no contienen más funcionalidad, en el caso de la clase Usuario podemos encontrar algún método especial que forma parte de su funcionamiento intrínseco.

6.2.1.1.2 Paquete org.persistence

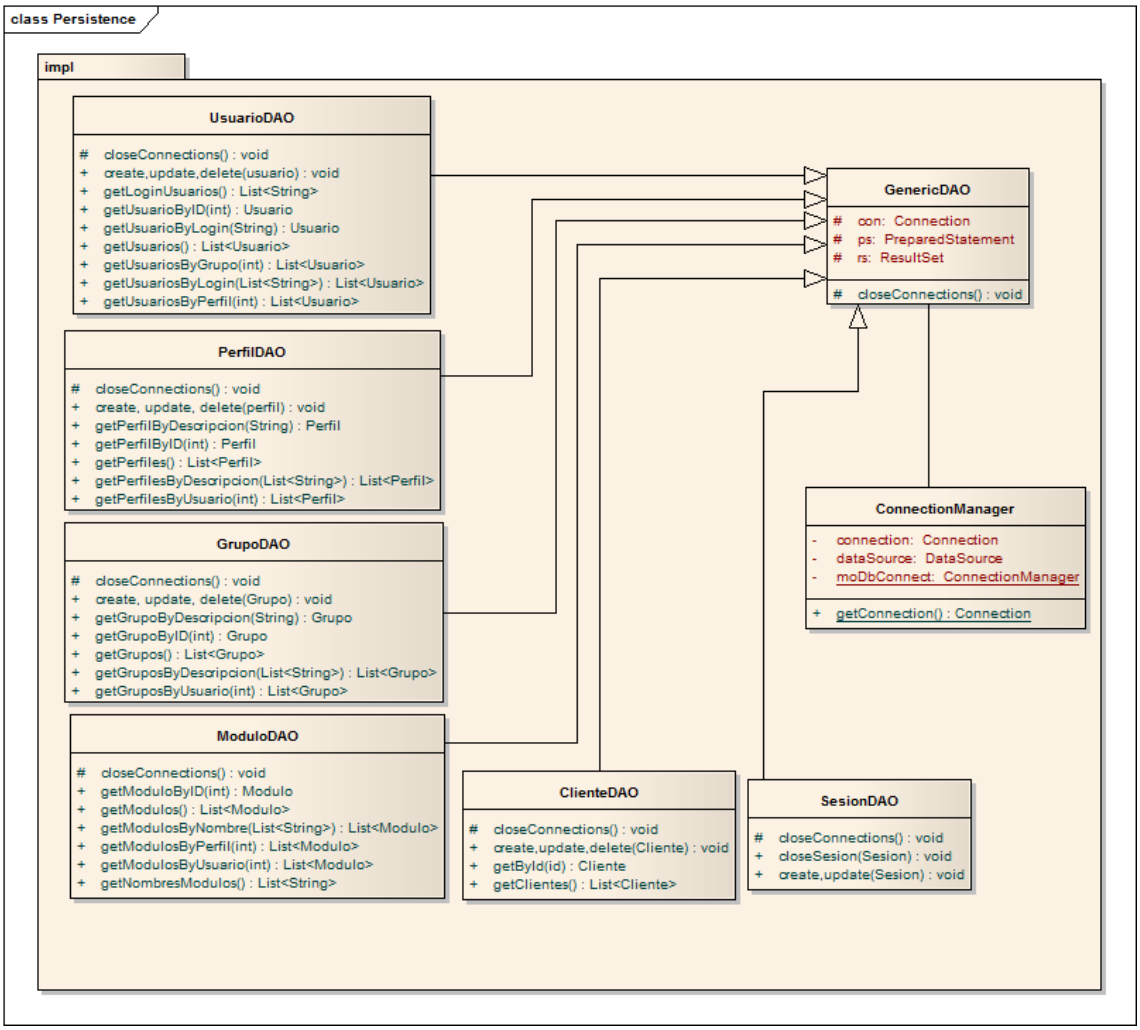


Diagrama 6.5: Clases-Persistencia intranet

En este diagrama se puede observar la capa de persistencia del módulo, como podemos ver se encuentra en un paquete “impl” que quiere decir que es la implementación de una fachada de interfaces que tiene por encima dicho paquete.

6.2.1.1.3 Paquete org.business

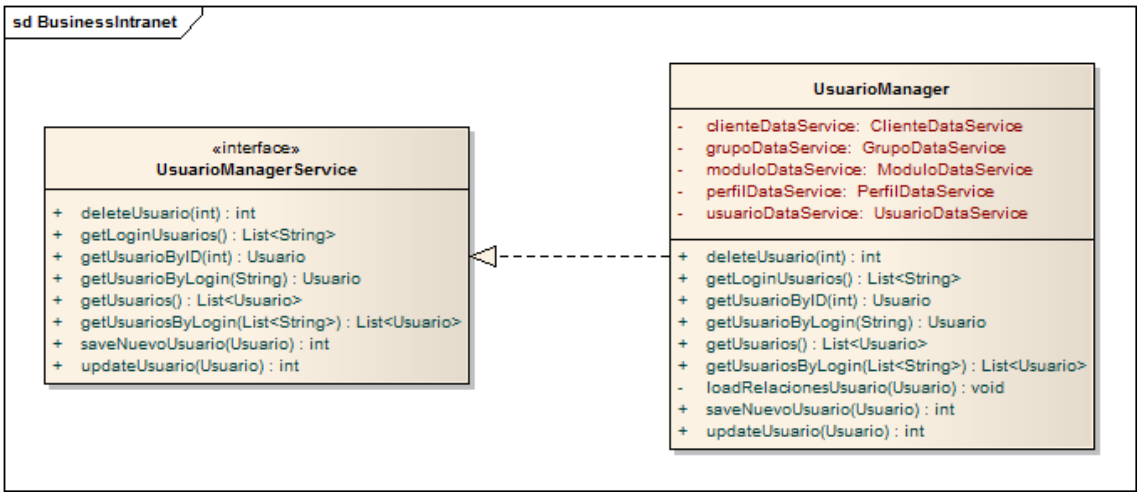


Diagrama 6.6: Clases-Negocio intranet

Como vemos en este diagrama, el módulo intranet no tiene una capa de lógica muy extensa, puesto que sus interacciones con la base de datos datan poco más que de cargar un usuario de la misma o en un caso distinto, actualizar su contraseña, sin embargo se dota de un manejador lo más completo posible para evitar a los módulos que necesiten utilizar el objeto usuario tener que crear uno propio.

6.2.1.1.4 Paquete org.presentation

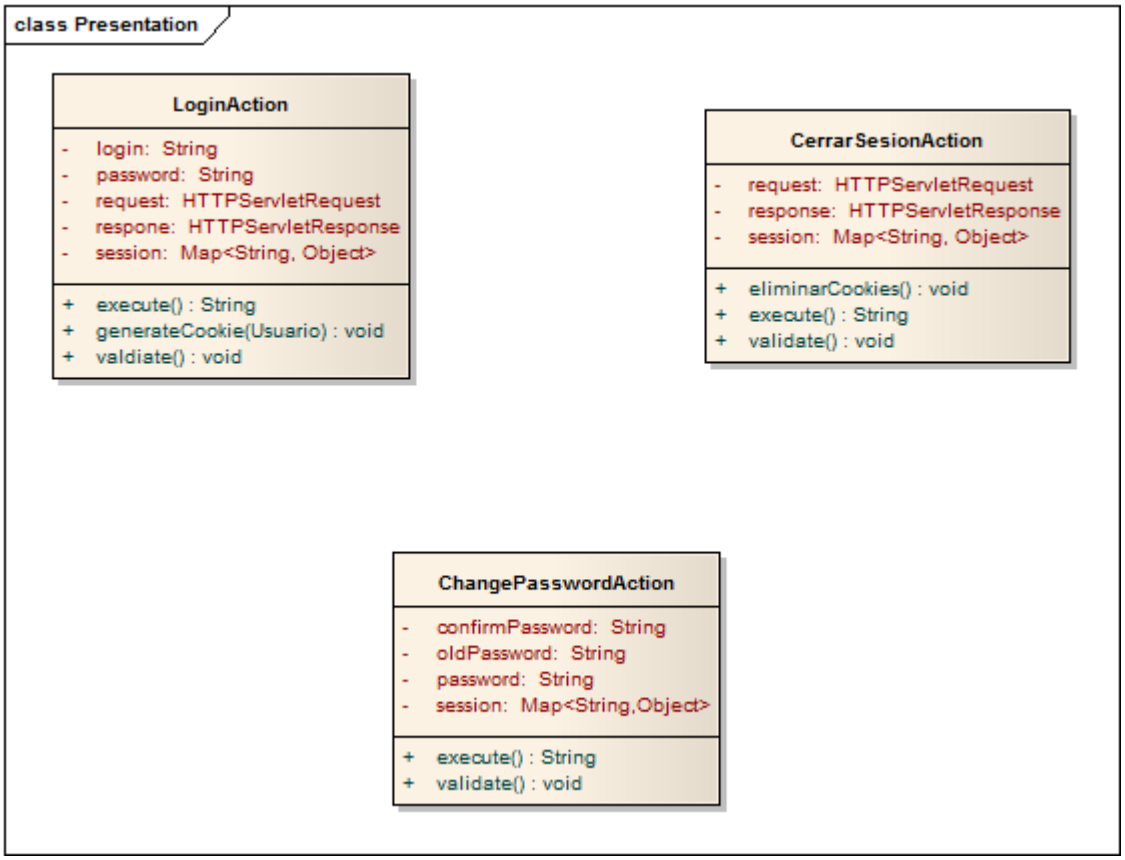


Diagrama 6.7: Clases-Presentación intranet

Este paquete contiene los "actions" del módulo Intranet, todos extienden de la clase "ActionSupport" e implementan como mínimo la interfaz "SessionAware".

6.2.1.1.5 Paquete org.security

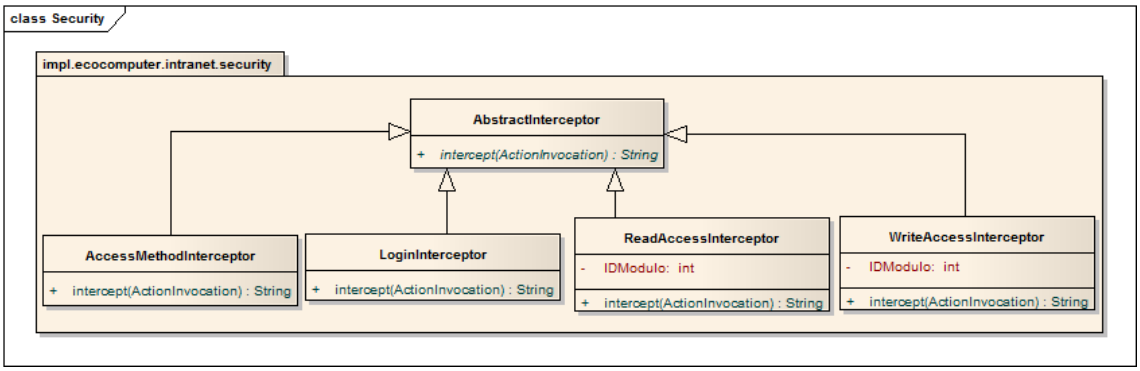


Diagrama 6.8: Clases-Seguridad intranet

Este paquete recoge la implementación de todos los interceptores que utilizará la aplicación y todas las aplicaciones que se desarrollen para la Intranet.

6.2.1.2 *Módulo de usuarios*

6.2.1.2.1 Paquete org.model

Esté módulo no tiene paquete de modelos, pues utiliza por completo los implementados en el módulo Intranet por medio de la librería generada con dicho módulo.

6.2.1.2.2 Paquete org.persistence

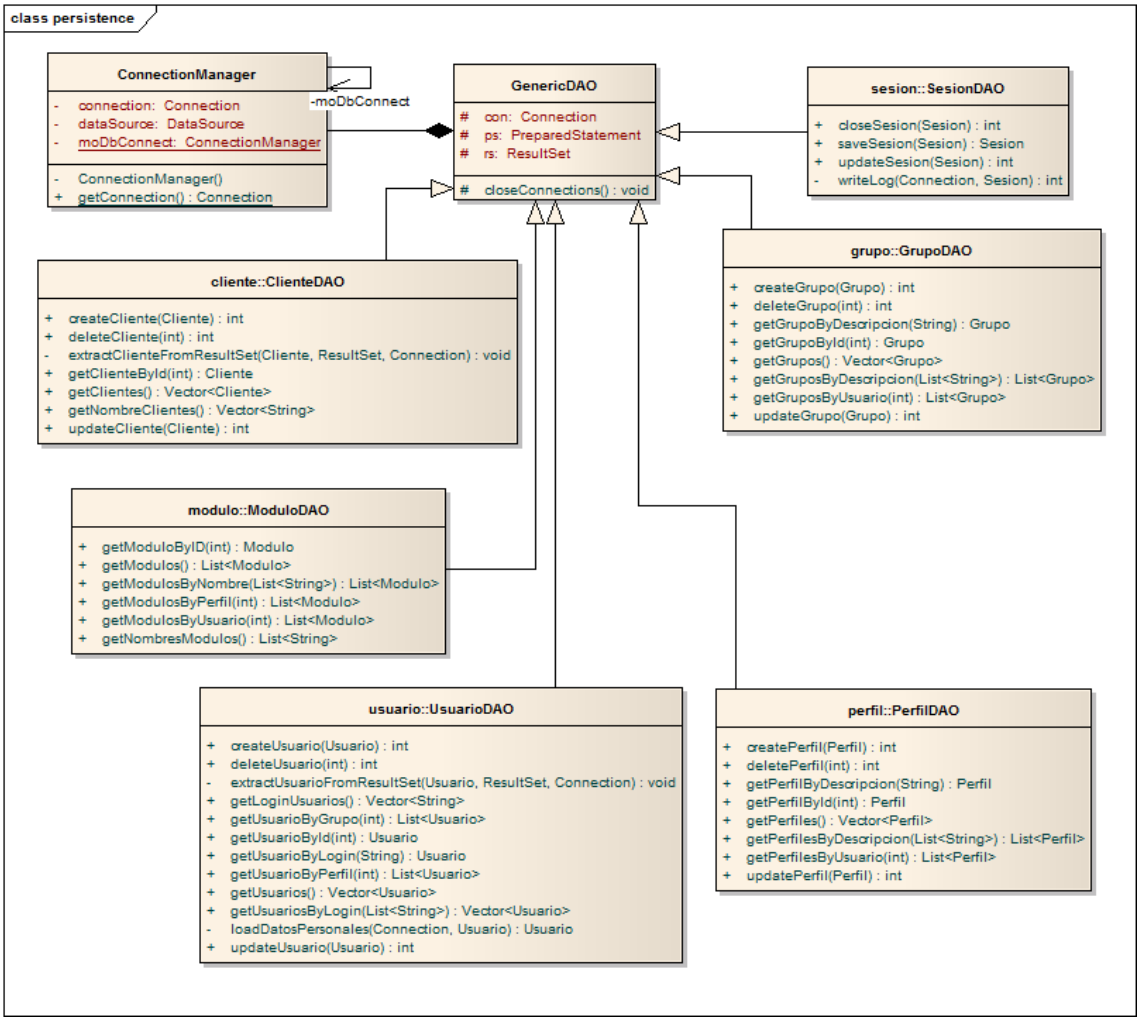


Diagrama 6.9: Clases-Persistencia usuarios

De este diagrama cabe mencionar que todos los DAO implementan unas interfaces DataService que funcionan como fachada para la capa de negocio, de esta manera si se requiriese un cambio en la funcionalidad de los mismos (por ejemplo utilizar JPA en lugar de JDBC) podría hacerse sin que esto supusiese cambios en las capas superiores.

6.2.1.2.3 Paquete org.business

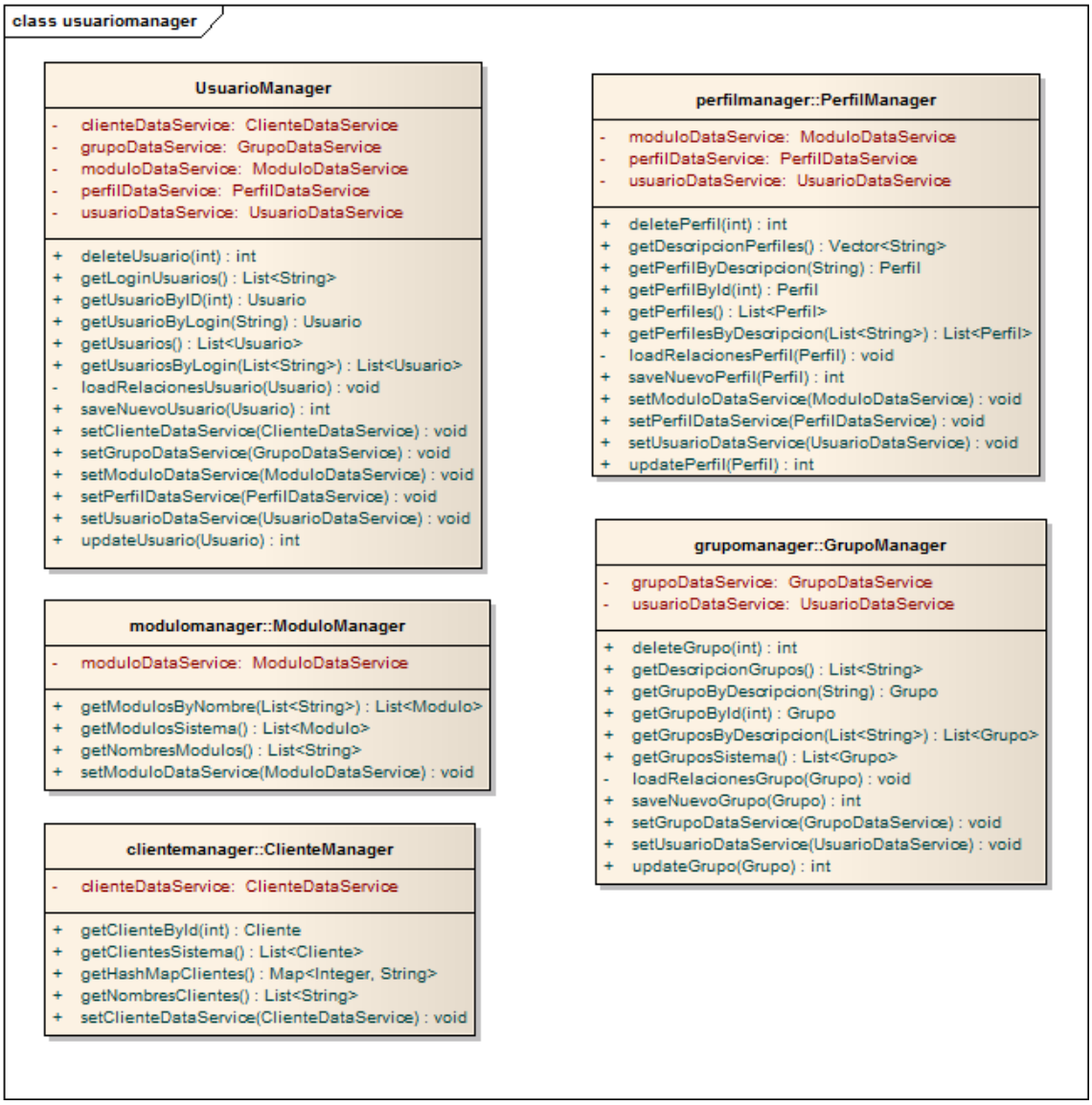


Diagrama 6.10: Clases-Negocio usuarios

Nuevamente todas estas clases se comunican con las capas superiores e inferiores a través de fachadas. Cada manager es el encargado de preparar los datos y convertirlos a los objetos del modelo que necesite para persistirlos en la base de datos. Por ello, los manejadores de clases del modelo con relaciones tienen referencias a más de una interfaz de acceso a datos.

6.2.1.2.4 Paquete org.presentation

Este paquete se divide a su vez en subpaquetes para cada uno de los objetos principales con los que interactúa el usuario en la capa de presentación, por ello, se mostrarán por separada facilitando así la comprensión por parte de los lectores.

Lo que ha de tenerse en cuenta sobre todos los diagramas de este apartado se resume en lo siguiente:

- Todos los Actions heredan de la clase definida en la librería de Struts "ActionSupport", de la que implementan los métodos "execute" y "validate".
- Los Actions que implementan la interfaz ModelDriven lo hacen sobre un objeto, esto se utiliza para que con los datos de la request que llegan desde la jsp se genere automáticamente el objeto mediante inyección de dependencias por nombre.
- Los Actions que implementar las interfaces "xAware" se utilizan para acceder de manera sencilla a objetos propios de la aplicación, entre otros: sesión, parámetros, request, response…

6.2.1.2.4.1 *Grupos*

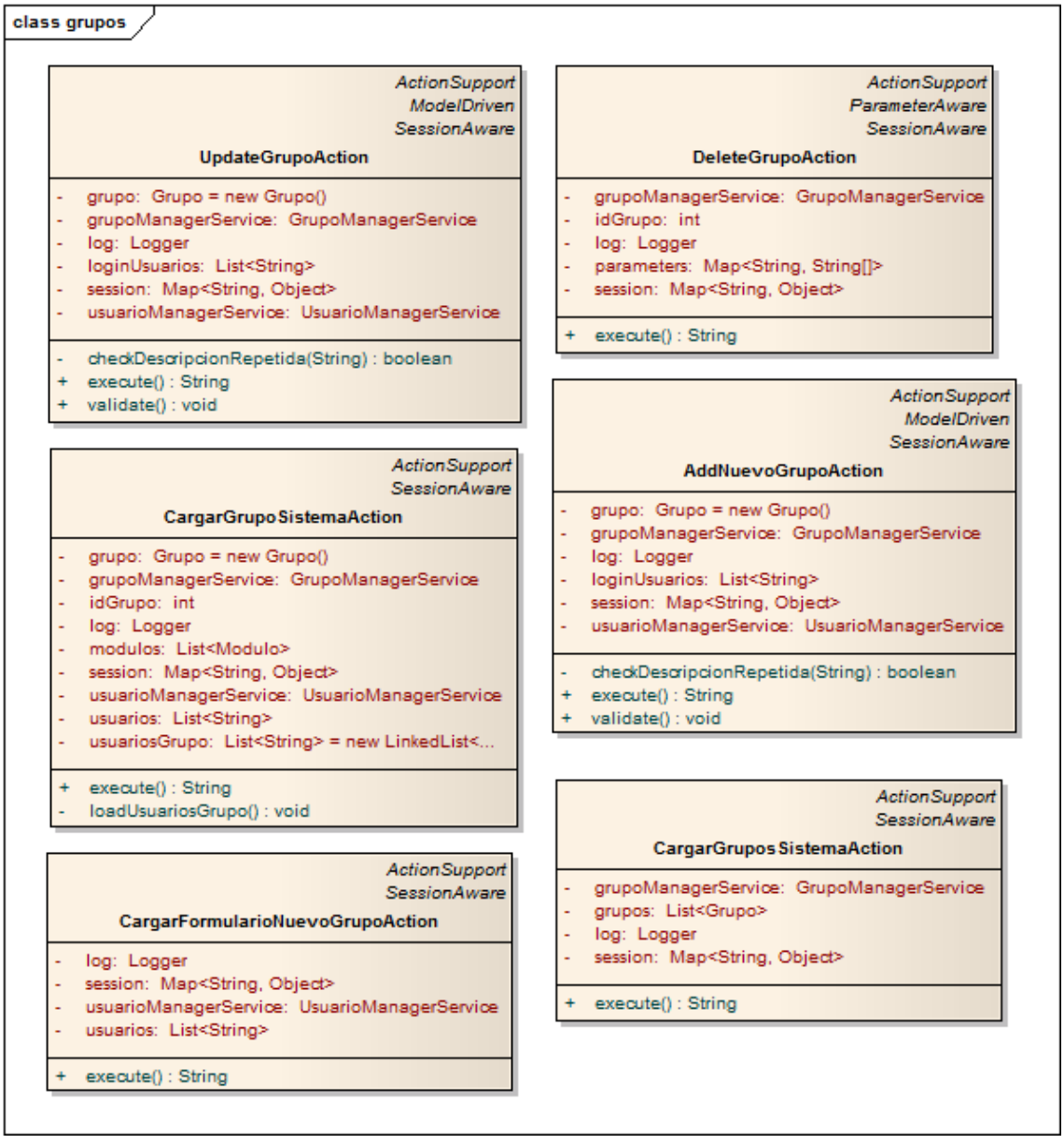


Diagrama 6.11: Clases-Presentación usuarios (grupos)

6.2.1.2.4.2 *Perfiles*

Diagrama 6.12: Clases-Presentación usuarios (perfiles)

6.2.1.2.4.3 *Usuarios*

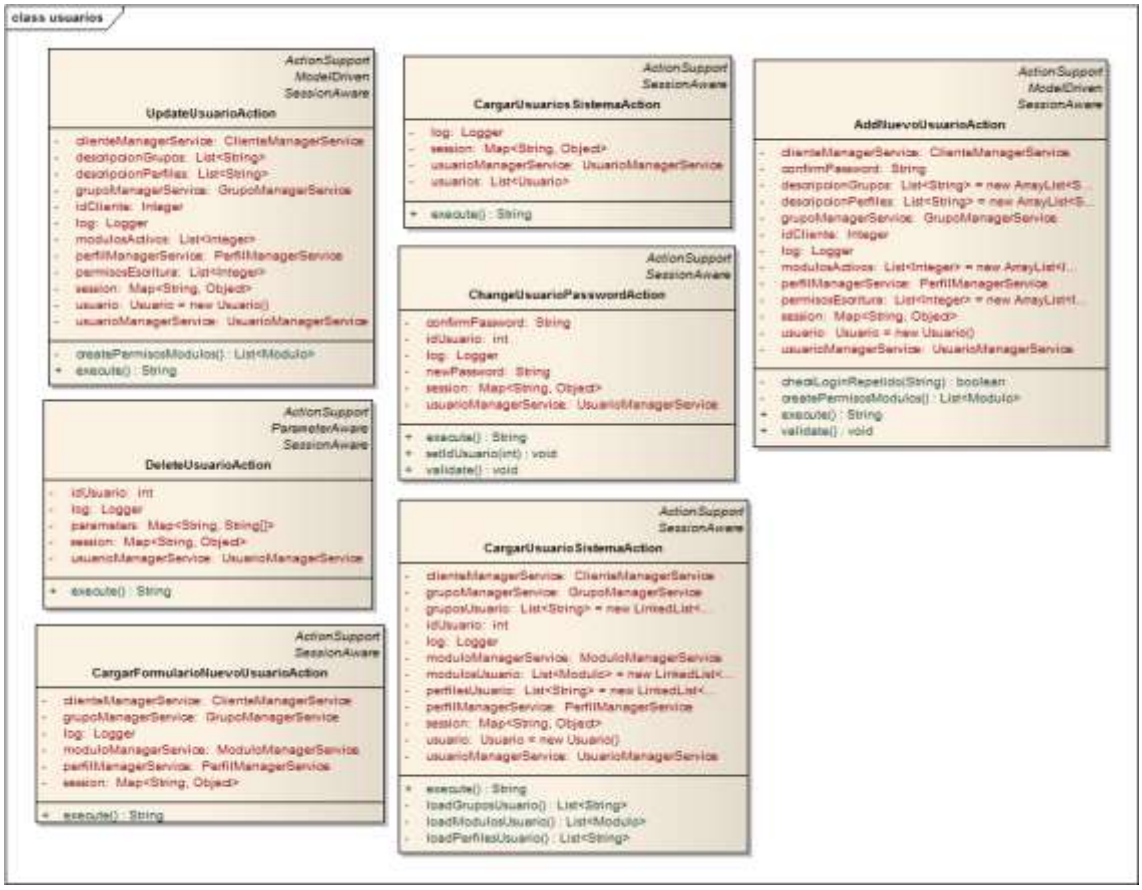


Diagrama 6.13: Clases-Presentación usuarios (usuarios)

6.2.1.3 *Módulo de dominios*

6.2.1.3.1 Paquete org.model

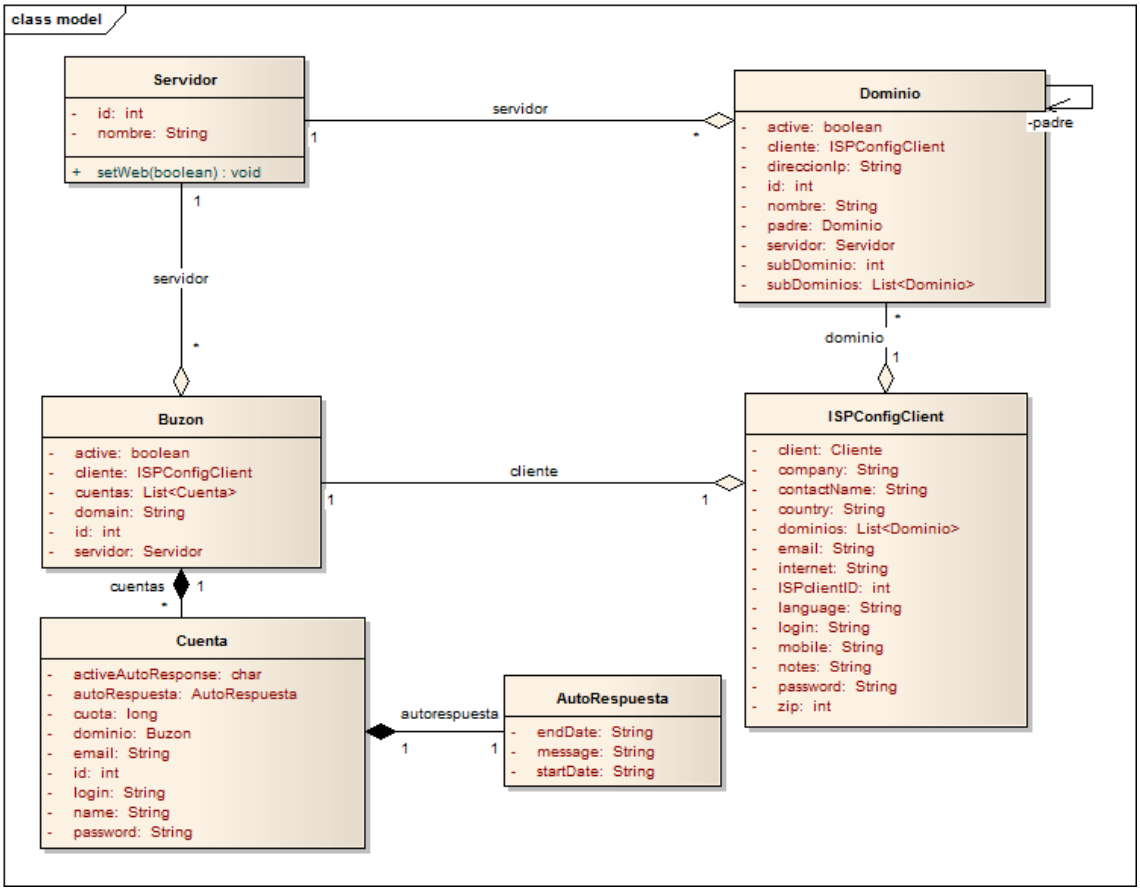


Diagrama 6.14: Clases-Modelo dominios

Como vemos en este módulo aparece un conjunto totalmente nuevo de modelos, para facilitar el entenderlo, Los dominios y los buzones son elementos similares, los dos se alojan en un servidor y tienen un cliente asociado, en el caso de buzón tiene asociadas unas cuentas de correo, las cuales a su vez pueden tener configurada una autorespuesta.

Finalmente el elemento principal es el cliente ISP, desde el podemos acceder a los datos personales del cliente, tanto el que se encuentra en el servidor ISP como el cliente de la Intranet.

6.2.1.3.2 Paquete org.persistence

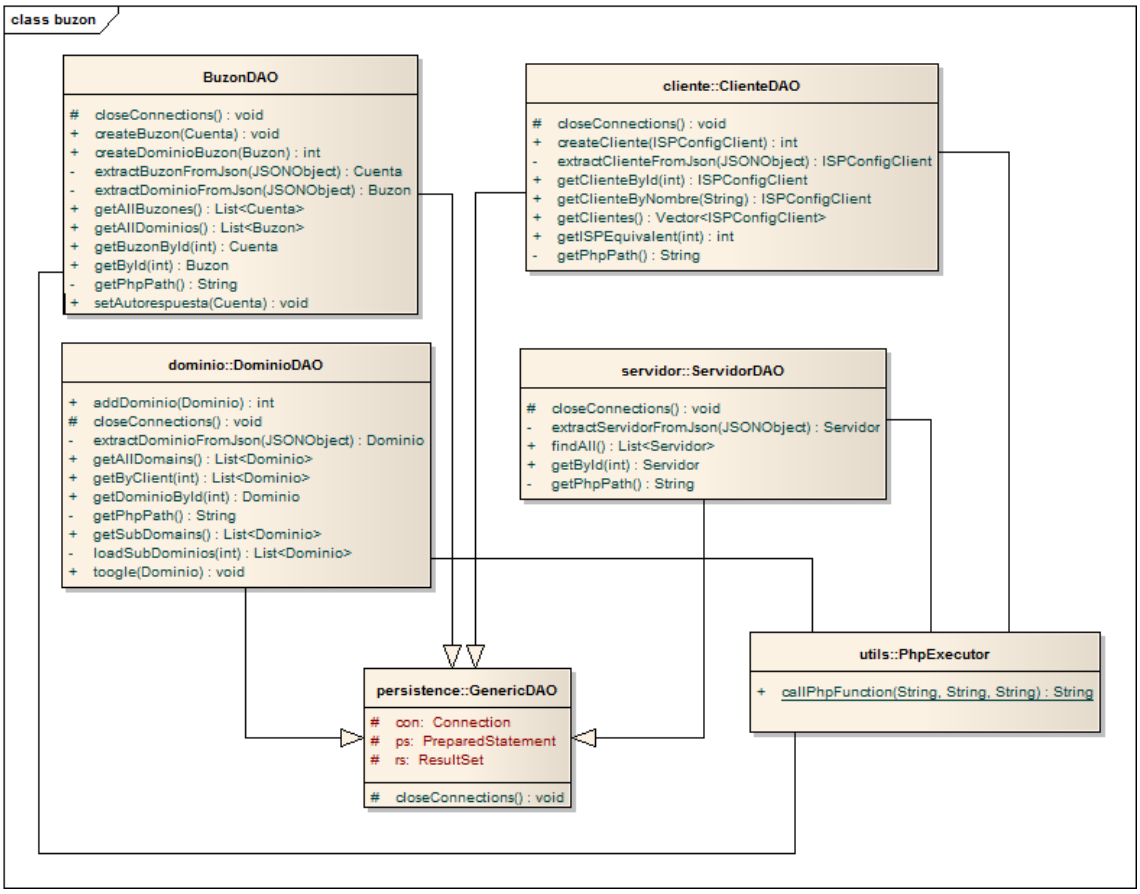


Diagrama 6.15: Clases-Persistencia dominios

Al igual que con los otros módulos los objetos DAO están detrás de una interfaz que hace a su vez de fachada, cabe destacar aquí que mucha de la persistencia no se hace directamente a base de datos, si no a un servidor externo ISP config, por medio de servicios SOAP, estos servicios vienen con unos ejemplos de invocación remota hecho en PHP. Se intentó llevar a cabo en Java, pero debido a la ausencia de descriptor de servicios wsdl, el consumo de dichos servicios resultaba complicar en gran medida el código y se decidió implementar una clase que ejecutase los ejemplos PHP antes mencionados con la finalidad de conseguir el objetivo de serialización.

La manera de interactuar con estos conlleva generar un archivo con la información de los modelos en formato JSON, el cual se lee desde el script PHP y es transferido al servidor ISP.

6.2.1.3.3 Paquete org.business

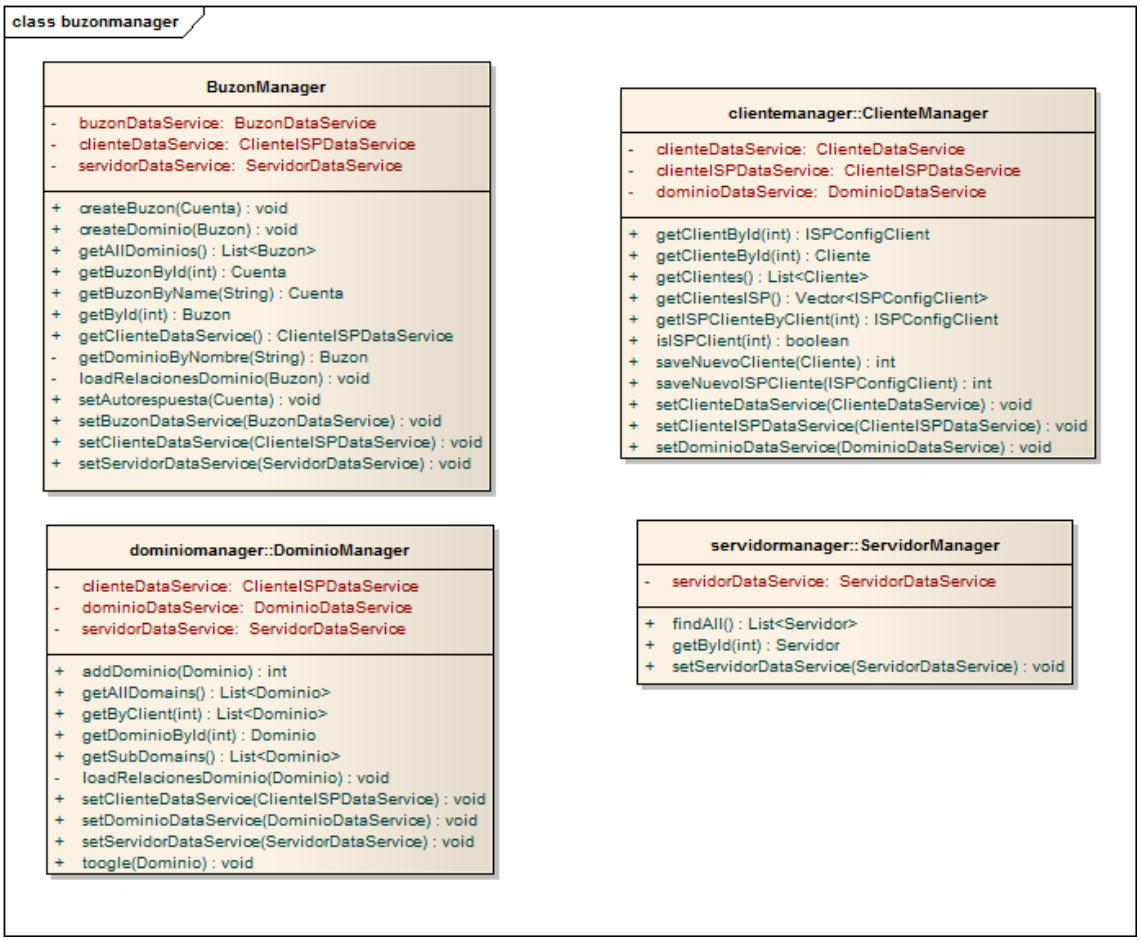


Diagrama 6.16: Clases-Negocio dominios

Nada que destacar en este paquete que no se haya explicado ya en otro modulo, son clases Manager que implementan interfaces Service de cara a la capa de presentación y que se comunican con la fachada de la capa de presentación.

6.2.1.3.4 Paquete org.presentation

Al igual que en el módulo de gestión de usuarios, se dividirá esto en tres ramas principales concernientes a los objetos más importantes del módulo.

6.2.1.3.4.1 *Buzones*

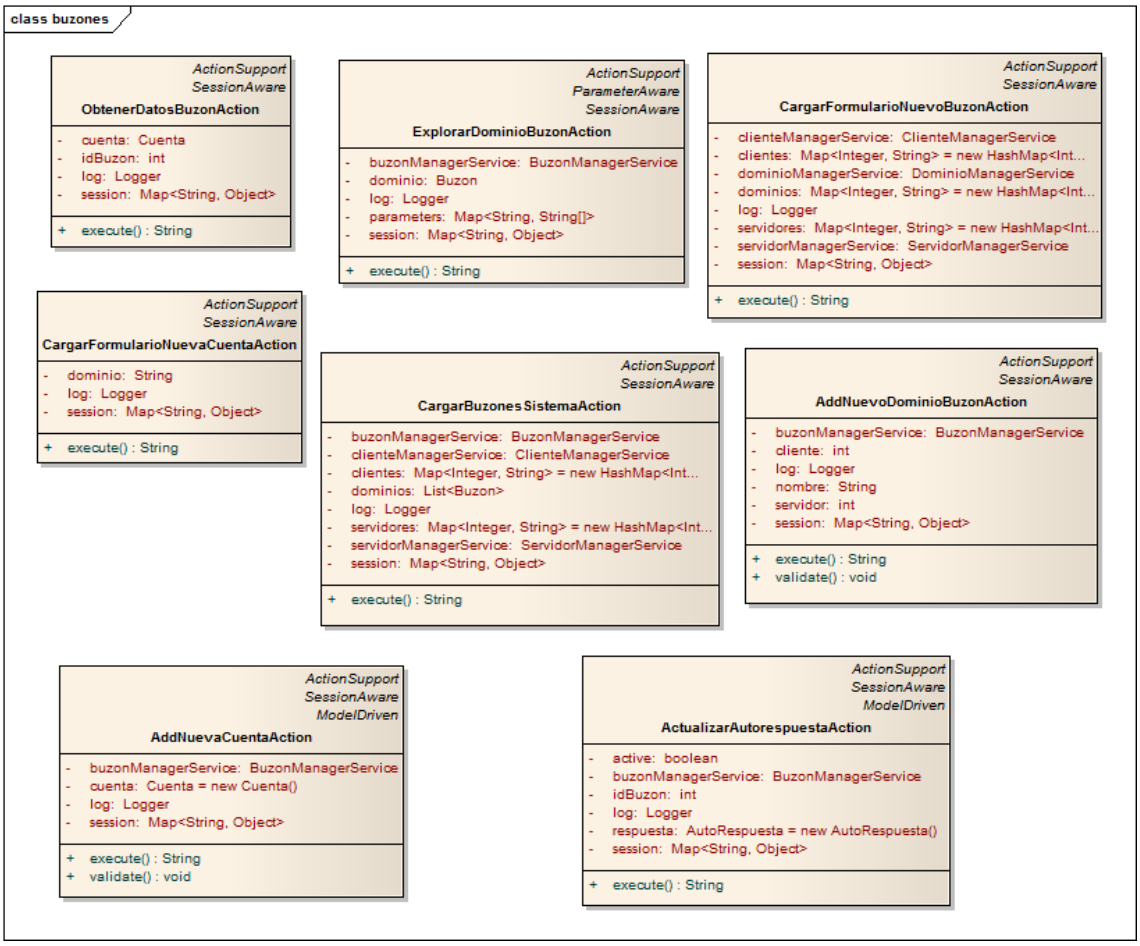


Diagrama 6.17: Clases-Presentación dominios (buzones)

6.2.1.3.4.2 *Clientes*

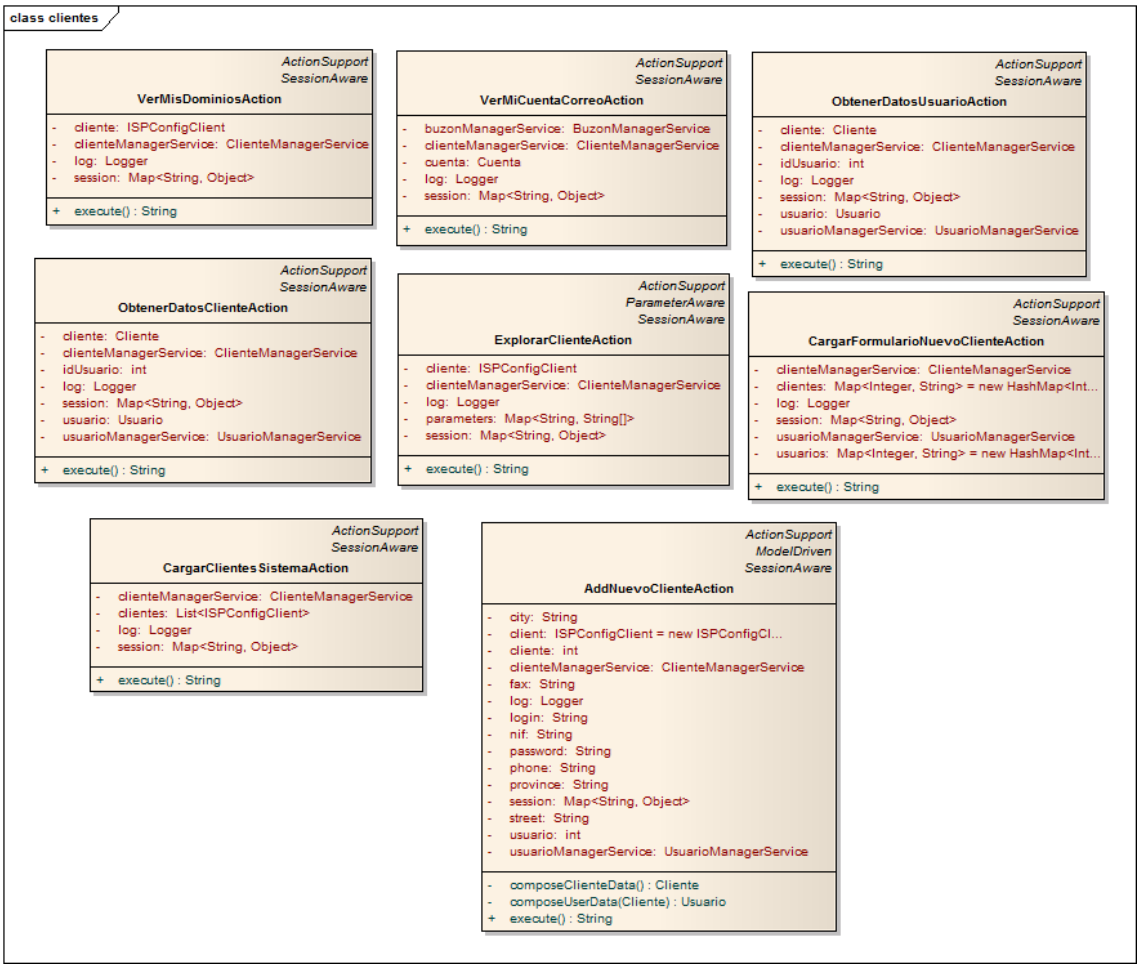


Diagrama 6.18: Clases-Presentación dominios (clientes)

El único Action reseñable en este caso es AddNuevoCliente, debido a su característica especial que es generar tres objetos distintos a través de un mismo formulario, si bien es cierto que la aproximación no ha sido la más acertada, pues lo que se suele hacer es crear un objeto contenedor para todos los objetos que se quieran generar y aplicar modelDriven para dicho objeto la falta de tiempo hace que quede como posible ampliación.

6.2.1.3.4.3 *Dominios*

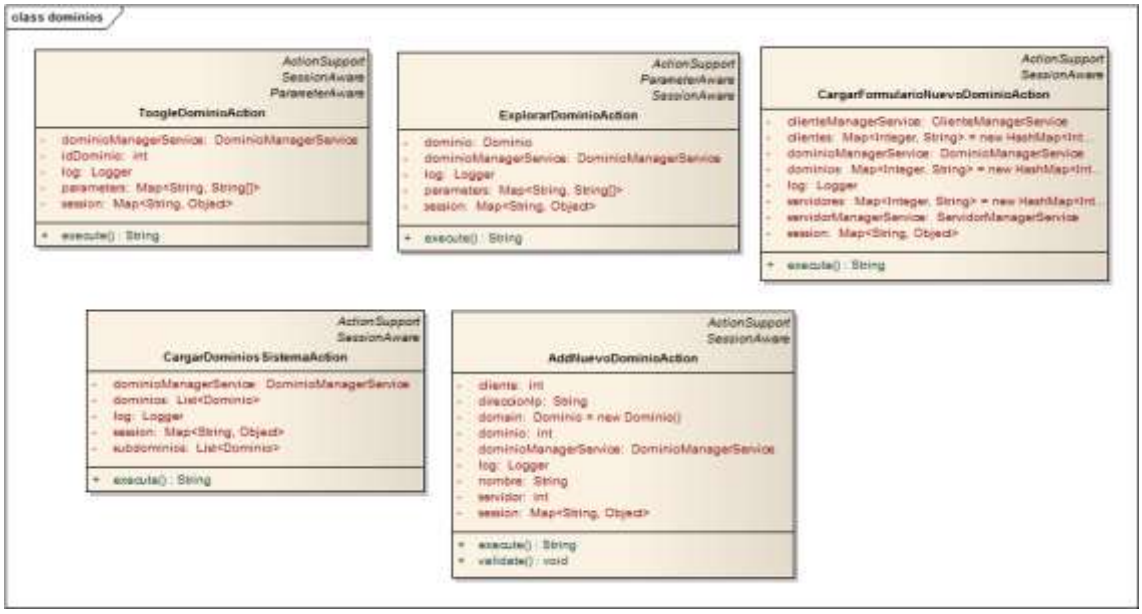


Diagrama 6.19: Clases-Presentación dominios (dominios)

6.3 Diagramas de Interacción y Estados

A continuación encontraremos una serie de diagramas en los que se pretenderá explicar el funcionamiento interno de las acciones más importantes del proyecto a lo largo de su ejecución paso a paso.

Mediante los diagramas de interacciones veremos el conjunto de pasos y métodos llamados desde que se inicia una operación hasta que finaliza, además de los elementos implicados en ello. A través de los diagramas de estados se pretende ilustrar qué fases atraviesan los objetos durante una u otra operación.

Se muestran los diagramas más representativos de la aplicación, ya que como hemos visto las operaciones son relativamente similares entre sí.

6.3.1 Caso de uso: Iniciar sesión

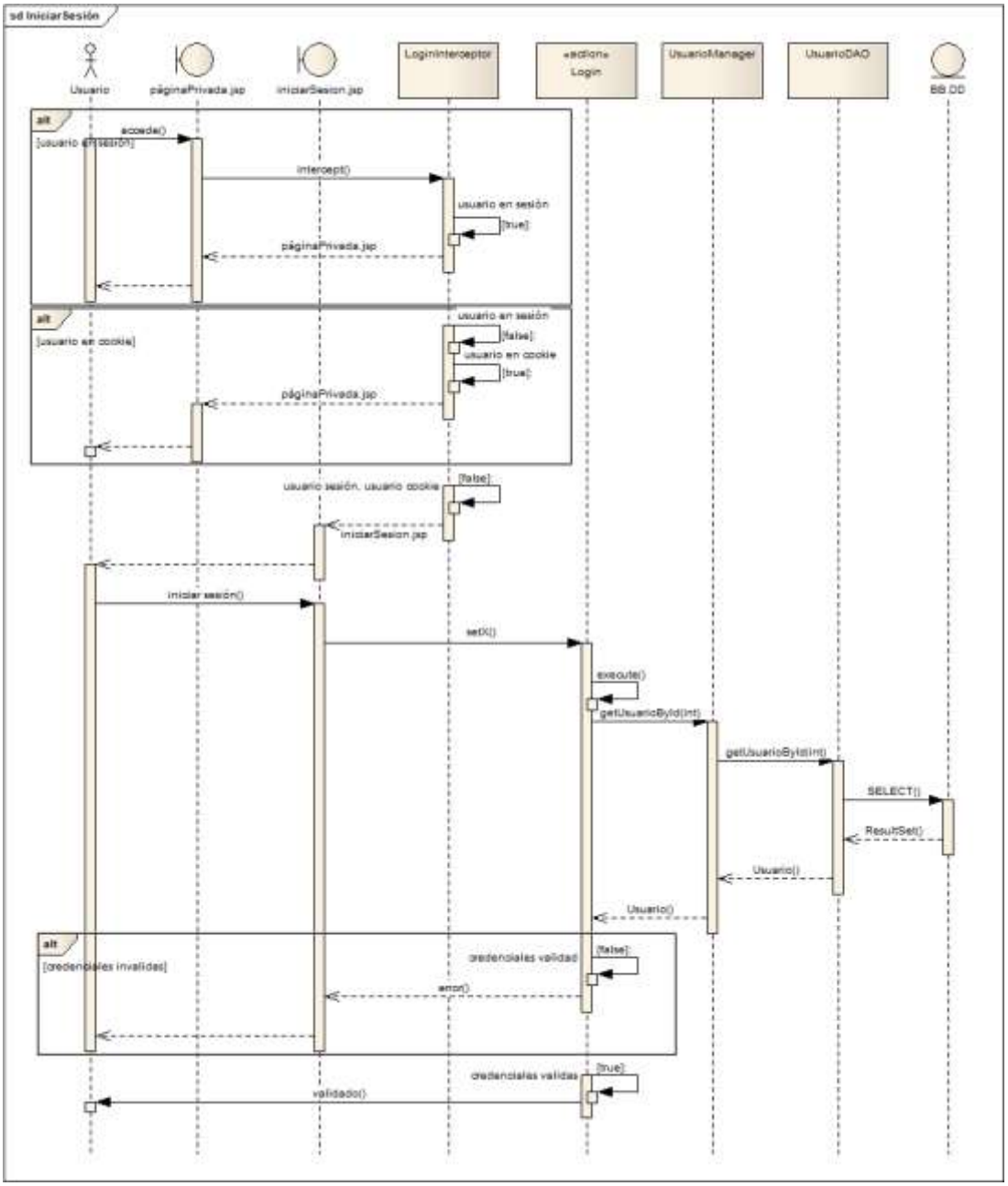


Diagrama 6.20: Interacción-Iniciar sesión

Con este diagrama se quiere describir el funcionamiento del interceptor que es como sigue:

Si el usuario intenta acceder a una página web de la parte privada de la aplicación el interceptor de login se invoca y realiza varias comprobaciones:

- Comprueba si el usuario está en sesión, si es así le permite seguir navegando.
- Comprueba si existe una cookie con la información del usuario, en caso de que exista, genera el objeto usuario y lo coloca en sesión para siguientes comprobaciones.

- Si se da cualquiera de las condiciones anteriores el interceptor deja transcurrir la petición del usuario y le garantiza acceso a la web. En el caso contrario se le devuelve a la página de inicio de sesión.

Como podemos observar se podría decir que el usuario pasa por varios estados a la hora de encontrarlo dentro de la aplicación, estos se intentarán dejar más claros con un diagrama de estados a continuación.

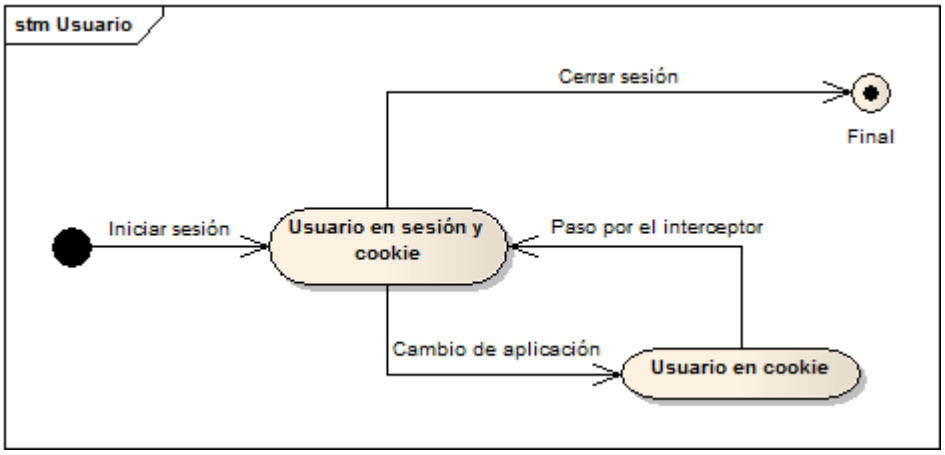


Diagrama 6.21: Estado-Usuario

6.3.2 Añadir objeto a base de datos: Añadir usuario

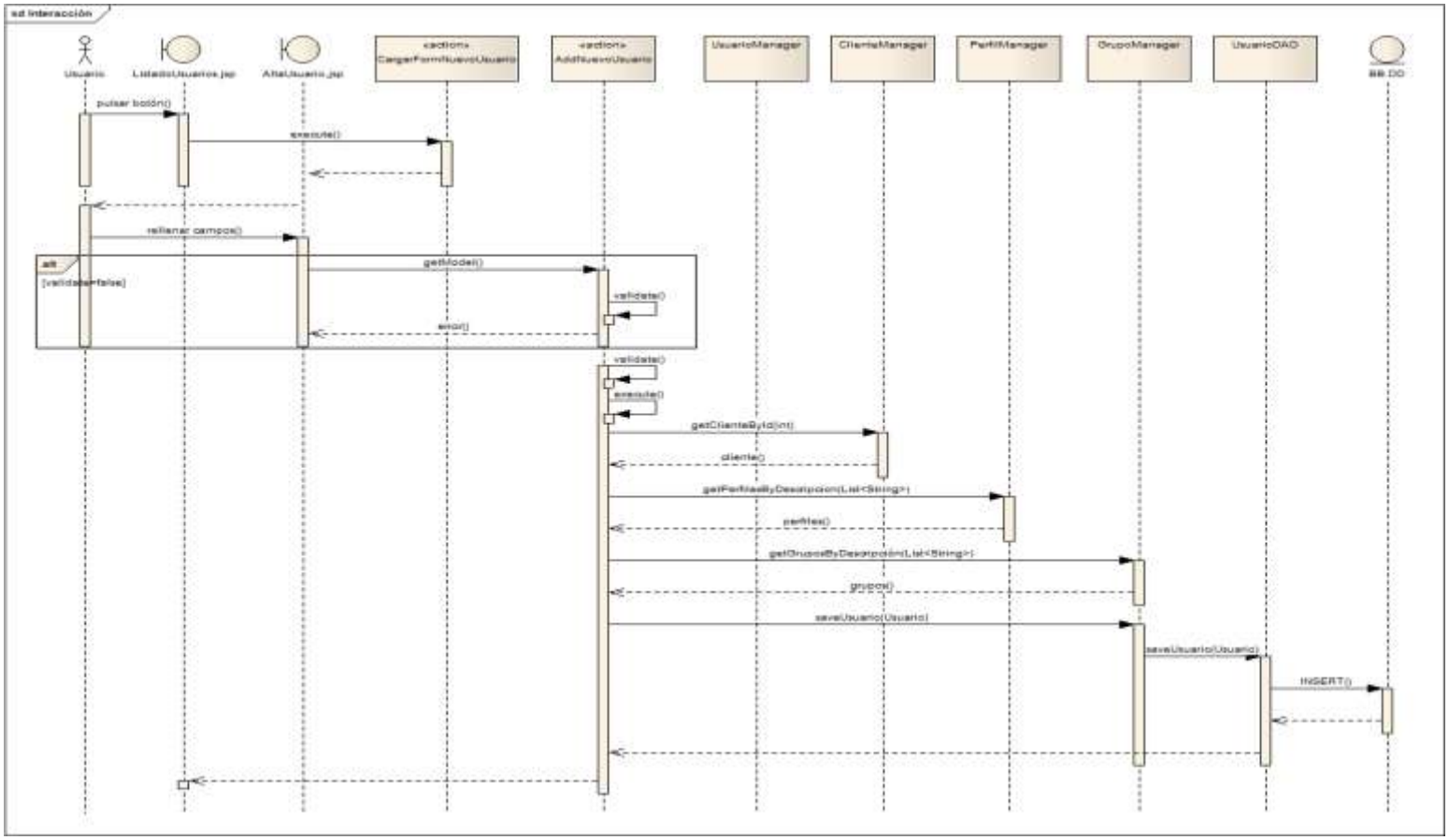


Diagrama 6.22: Interacción-Insertar usuario

Con el diagrama anterior se pretende explicar el funcionamiento general de las operaciones de inserción, utilizando para ello la más compleja de todo el proyecto desarrollado, la inserción de un usuario, debido a todas las relaciones que tiene la clase.

Cuando en un "action" falla la validación se devuelve el control a la interfaz de usuario con un mensaje de error para el usuario propiamente dicho, en el caso de que la validación sea correcta se continúa con la ejecución del "action".

Para simplificar el diagrama se ha omitido todos los DAOs a los que llaman los manejadores secundarios de este proceso, pero todos ellos tienen que realizar accesos de consulta a la base de datos.

Es importante en este tipo de operaciones indicar cómo funciona el manejo de conexiones por medio del servidor, el cual se hará mediante un diagrama de estados.

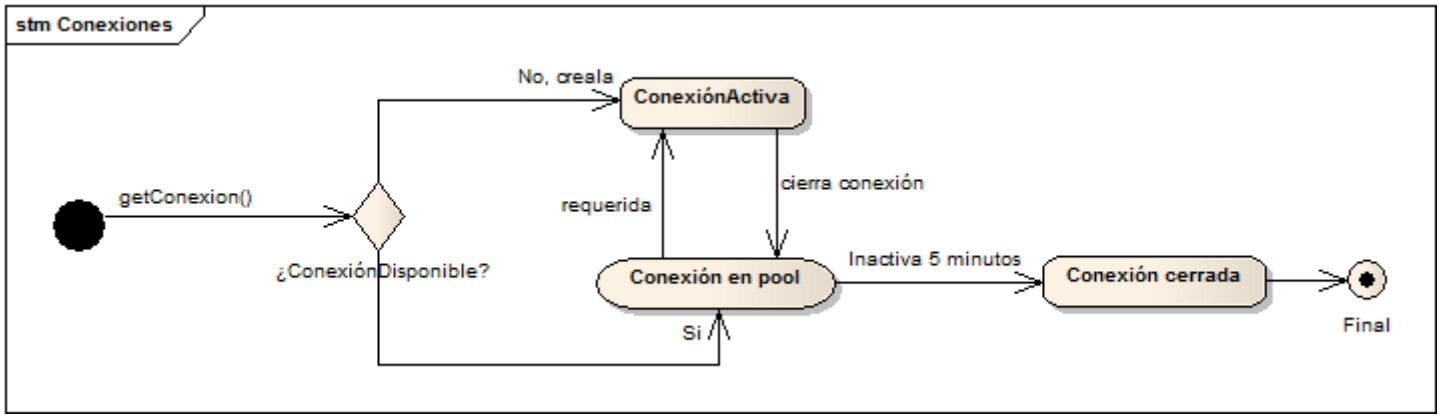


Diagrama 6.23: Estado-Conexión

6.3.3 Añadir objeto al servidor ISP: Dominio

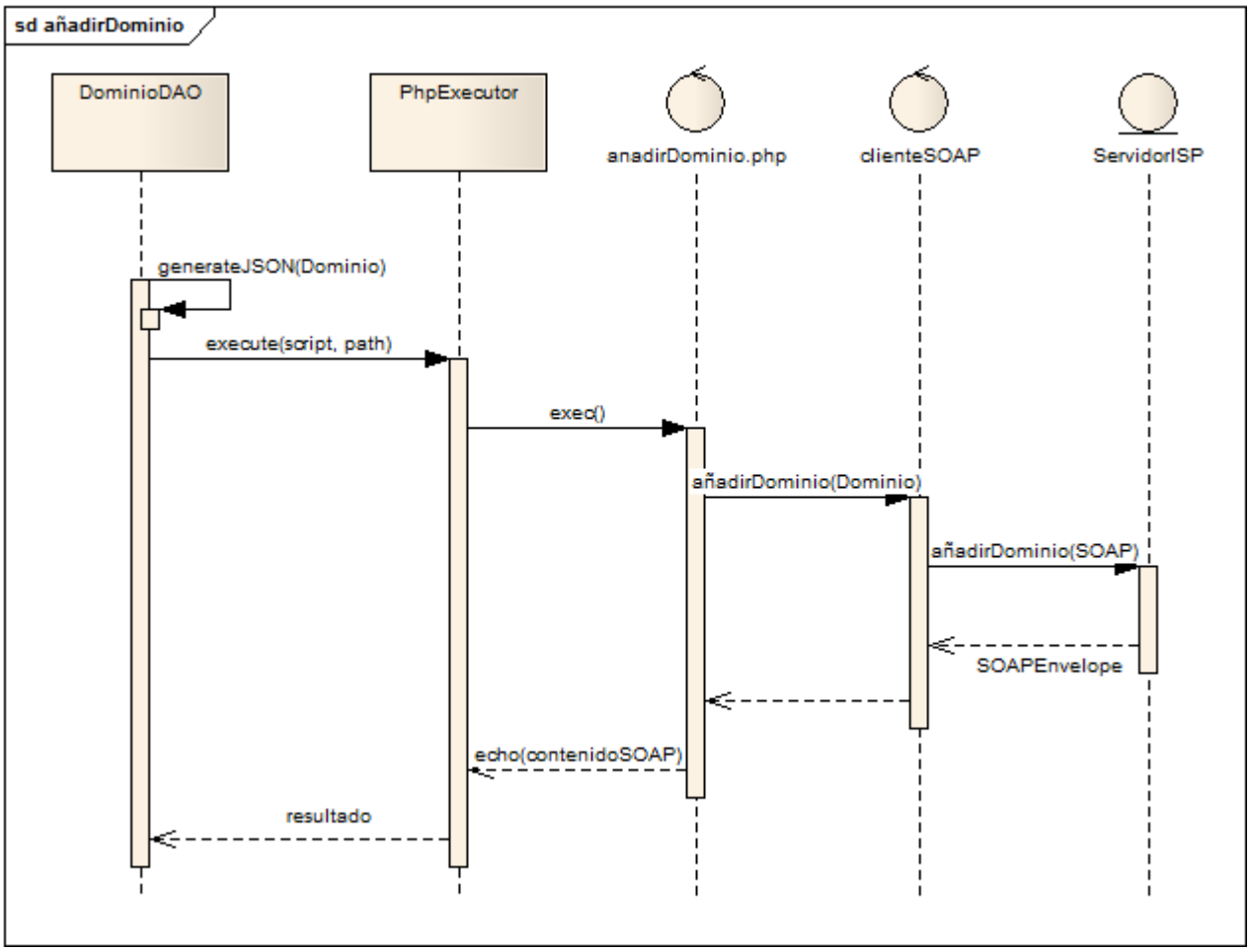


Diagrama 6.24: Interacción-Insertar dominio

Este diagrama se ha simplificado pues la mayor parte de lo que acontece en este caso de uso es similar a la inserción de usuarios y lo que se quiere reseñar en esta es la invocación de un servicio SOAP mediante PHP, debido a la dificultad que suponía consumirlo en Java acarreada por la falta de un descriptor de servicios y una documentación bien proporcionada.

Hay que tener en cuenta que en este tipo de inserciones o accesos al servidor ISP se utiliza JSON para conseguir que el objeto llegue al script PHP.

6.4 Diseño de la Base de Datos

En la aplicación podemos distinguir una única base de datos en la que se almacenará toda la información necesaria para el correcto funcionamiento del proyecto, además de otras tablas y datos existentes, utilizadas por otras aplicaciones y secciones de la intranet general pero que no tienen relación con lo aquí expuesto.

Es por esto que se hablará únicamente de las tablas empleadas por este proyecto y no de la totalidad de la base de datos, que no ha sido diseñada ni gestionada por el alumno.

6.4.1 Descripción del SGBD Usado

El Sistema de Gestión de Base de Datos utilizado es un Microsoft SQL Server 2005, impuesto por la empresa cliente como condición necesaria, ya que toda su infraestructura está construida sobre este sistema en sus últimas versiones.

Este SGBD contempla soporte para transacciones, procedimientos almacenados, dispone de un entorno gráfico para su administración y la posibilidad de trabajar en modo cliente-servidor. Gracias al cliente nativo de SQL que tiene incorporado se pueden ejecutar las sentencias sobre dicho lenguaje de consulta.

6.4.2 Integración del SGBD en Nuestro Sistema

La Java Database Connectivity empleada en el proyecto dispone de los mecanismos y controladores necesarios para conectarse y trabajar con un SQL Server 2005, proporcionando su localización, nombre de la base de datos y un usuario y contraseña con acceso a la misma.

Las conexiones se realizan a través de un connection pool, que mantiene una colección de conexiones abiertas que pueden ser reutilizadas, agilizando así los mecanismos de conexión para con ella y evitando la sobrecarga de recursos, puesto que en una aplicación multitarea con varios usuarios simultáneos crear nuevas conexiones continuamente para cada petición ralentiza mucho la ejecución general.

6.4.3 Diagrama de la base de datos

El diagrama de base de datos se ha dividido en dos secciones para ilustrar de manera separada las diferentes partes que conforman el total.

6.4.3.1 *Tablas concernientes al usuario*

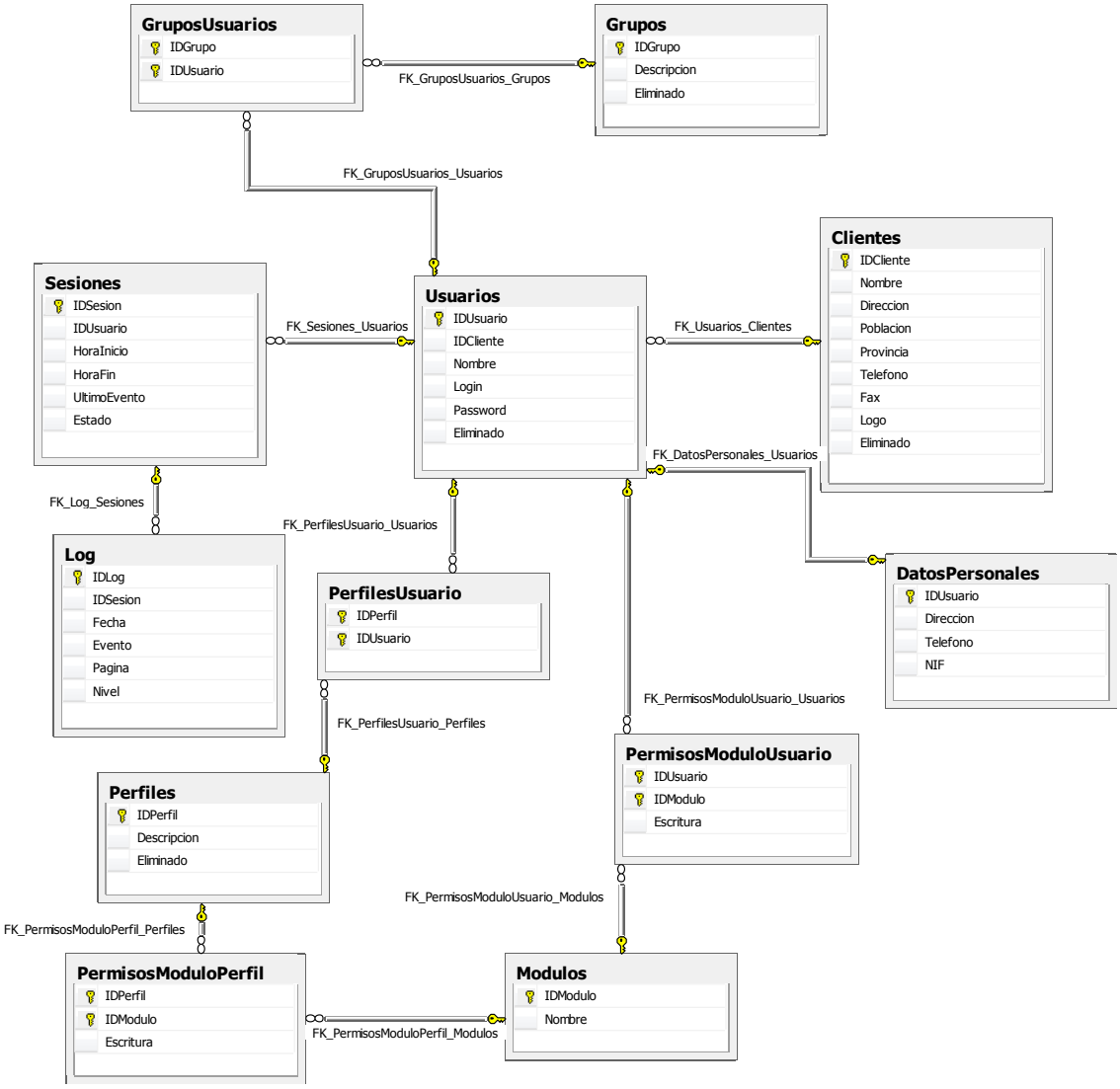


Diagrama 6.25: Entidad-Relación

Nombre de la tabla
Usuarios
Descripción
Tabla que almacena los datos relevantes a los usuarios del sistema.
Atributos
• **IDUsuario (PK, int, no NULL):** Identificador único del usuario. • **IDCliente (FK, int):** Identificador del cliente vinculado al usuario. • **Nombre (varchar(50), no NULL):** Nombre del usuario. • **Login (varchar(50), no NULL):** Nombre con el que el usuario inicia sesión. • **Password (varchar(50), no NULL):** Contraseña con la que el usuario inicia sesión. • **Eliminado (bit, no NULL):** Indica si el usuario ha sido eliminado o no.

Nombre de la tabla
Datos personales
Descripción
Tabla que almacena los datos personales y de contacto de los usuarios.

Atributos
• **IDUsuario (FK, PK, int, not NULL):** Identificador único del usuario. • **Dirección (varchar(150)):** Dirección física del usuario. • **Teléfono (varchar(20)):** Teléfono de contacto del usuario. • **NIF (varchar(10)):** Documento de identificación nacional del usuario.

Nombre de la tabla
Clientes
Descripción
Tabla que almacena los datos relevantes a los clientes del sistema.
Atributos
• **IDCliente (PK, int, not NULL):** Identificador único del cliente en el sistema. • **Nombre (varchar(50)):** Nombre del usuario. • **Dirección (varchar(150)):** Dirección física del cliente o la empresa. • **Población (varchar(50)):** Población del cliente o de la empresa. • **Provincia (varchar(50)):** Provincia del cliente o de la empresa. • **Teléfono (varchar(50)):** Teléfono de contacto del cliente o de la empresa. • **Fax (varchar(50)):** Fax del cliente o de la empresa. • **Logo (varchar(50)):** Ruta a la imagen logo de la empresa. • **Eliminado (bit, not NULL):** Indica si el cliente ha sido eliminado o no.

Nombre de la tabla
Perfiles
Descripción
Tabla que almacena los datos relevantes a los diferentes perfiles del sistema.
Atributos
• **IDPerfil (PK, int, no NULL):** Identificador único del perfil. • **Descripcion (varchar(50), no NULL):** Descripción y nombre del perfil. • **Eliminado (bit, no NULL):** Indica si el usuario ha sido eliminado o no.

Nombre de la tabla
PerfilesUsuario
Descripción
Tabla que relaciona a un usuario con su perfil.
Atributos
• **IDPerfil (PK, FK, int, no NULL):** Identificador único del perfil. • **IDUsuario (PK, FK, int, no NULL):** Identificador único del usuario.

Nombre de la tabla
Grupos
Descripción
Tabla que relaciona los datos relevantes a los grupos del sistema
Atributos
• **IDGrupo (PK, int, no NULL):** Identificador único del grupo. • **Descripcion (varchar(50), no NULL):** Descripción y nombre del grupo. • **Eliminado (bit, no NULL):** Indica si el usuario ha sido eliminado o no.

Nombre de la tabla
GruposUsuarios
Descripción
Tabla que relaciona un usuario con su grupo
Atributos
• **IDGrupo (PK, FK, int, no NULL):** Identificador único del grupo. • **IDUsuario (PK, FK, int, no NULL):** Identificador único del usuario.

Nombre de la tabla
Módulos
Descripción
Tabla que almacena los datos relevantes a los módulos o bloques de funcionalidad diferente del sistema.
Atributos
• **IDModulo (PK, int, no NULL):** Identificador único del módulo. • **Nombre (varchar(50), no NULL):** Nombre del módulo. • **Eliminado (bit, no NULL):** Indica si el módulo ha sido eliminado o no.

Nombre de la tabla
PermisosModuloPerfil
Descripción
Tabla que especifica qué permisos tiene un perfil determinado sobre un módulo concreto.
Atributos
• **IDPerfil (PK, FK, int, no NULL):** Identificador único del perfil. • **IDModulo (PK, FK, int, no NULL):** Identificador único del módulo. • **Escritura (bit, no NULL):** Indica si el perfil posee permisos de escritura (bit a 1) sobre el módulo o solo de lectura (bit a 0).

Nombre de la tabla
PermisosModuloUsuario
Descripción
Tabla que especifica qué permisos tiene un usuario determinado sobre un módulo concreto.
Atributos
• **IDPerfil (PK, FK, int, no NULL):** Identificador único del perfil. • **IDUsuario (PK, FK, int, no NULL):** Identificador único del usuario. • **Escritura (bit, no NULL):** Indica si el usuario posee permisos de escritura (bit a 1) sobre el módulo o solo de lectura (bit a 0).

Nombre de la tabla
Sesiones
Descripción
Tabla que almacena los datos de la sesión del usuario.
Atributos
• **IDSesion (PK, int, no NULL):** Identificador único de la sesión. • **IDUsuario (FK, int, no NULL):** Identificador único del usuario. • **HoraInicio (datetime, no NULL):** Fecha y hora a la que el usuario inició la sesión. • **HoraFin (datetime, NULL):** Fecha y hora a la que el usuario finalizó la sesión. • **UltimoEvento (datetime, no NULL):** Fecha y hora en la que el usuario realizó la última

operación.
- **Estado (bit, no NULL):** Indica el estado de la sesión.

Nombre de la tabla
Logs
Descripción
Tabla que funciona como registro o log de todas las acciones que el usuario realiza durante una sesión concreta.
Atributos
• **IDLog (PK, int, no NULL):** Identificador único de registro. • **IDSesion (FK, int, no NULL):** Identificador único de la sesión. • **Fecha (datetime, no NULL):** Fecha y hora a la que el usuario inició la sesión. • **Evento (varchar(200), no NULL):** Descripción del evento que causó la escritura en el registro. • **Página (varchar(100), no NULL):** Nombre de la página que causó la escritura en el registro. • **Nivel (bit, no NULL):** Nivel de abstracción del tipo de evento que causó la escritura en registro.

6.4.3.2 *Tablas concernientes a los dominios*

En este caso, en la base de datos solo se almacena una relación del ID del cliente con el ID del usuario del servidor ISPConfig vinculado. Todos los demás datos se almacenan dentro del servidor ISPConfig, del cual no se ha podido encontrar ningún diagrama de la base de datos interna, por lo cual el diagrama a continuación será el presupuesto por el alumno según el cual se ha desarrollado el dominio.

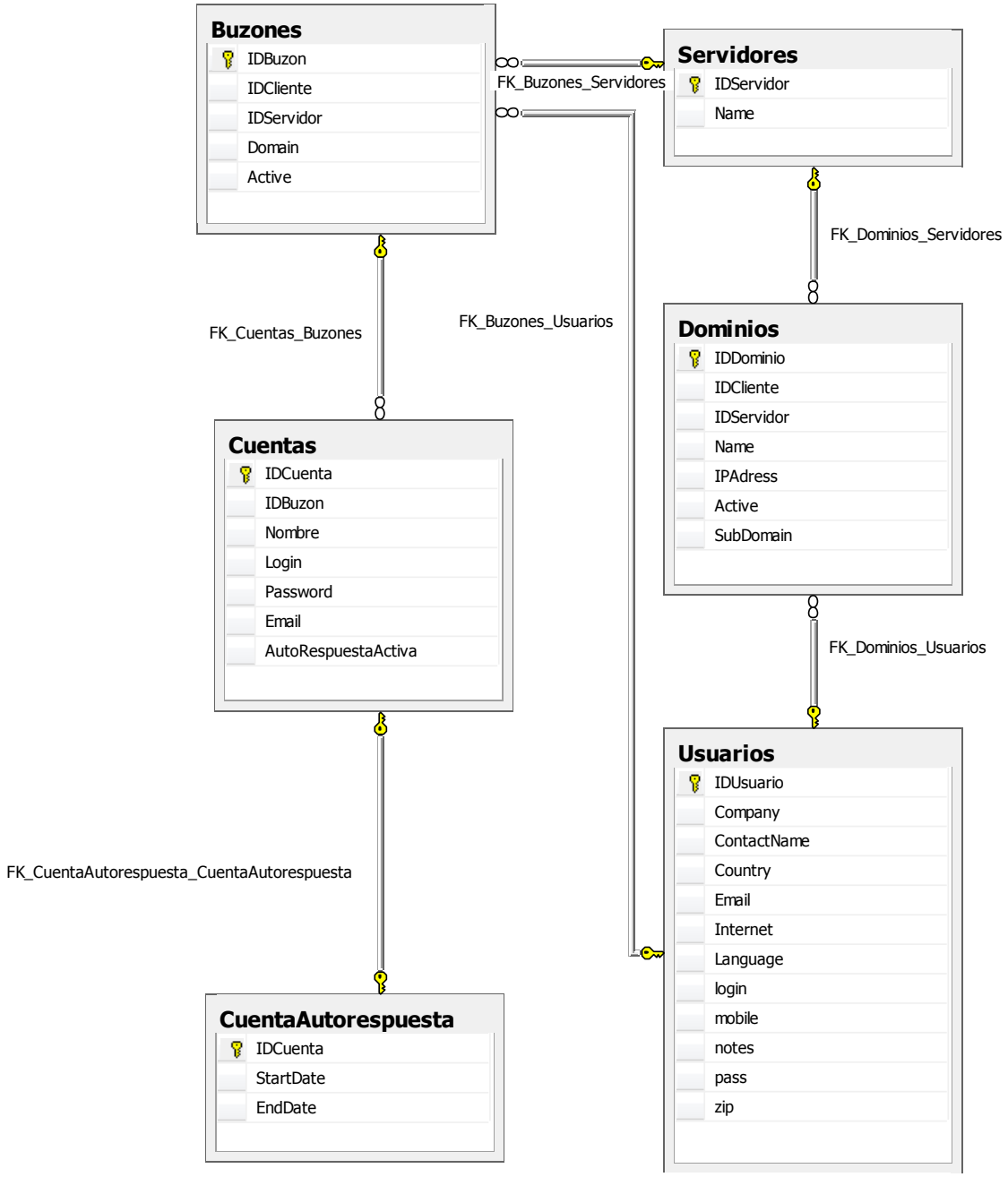


Diagrama 6.26: Entidad-Relación (ISPConfig)

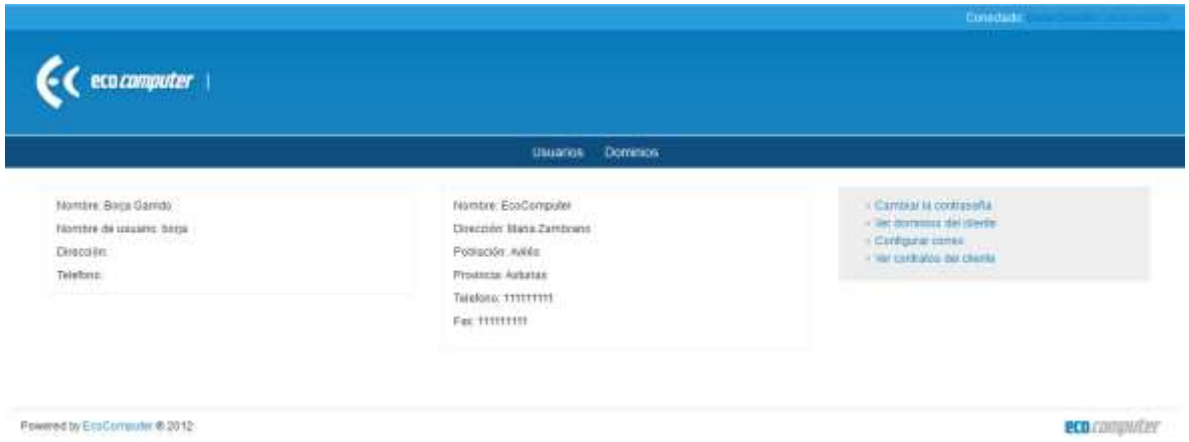
6.5 Diseño de la Interfaz

Como ya se comentó en la fase de análisis, fue responsabilidad de la empresa el diseñar las interfaces de la aplicación, distribuir sus contenidos, establecer su gama de colores y todo lo relacionado con la visualización de los contenidos.

El desarrollador solo tiene la responsabilidad de adaptar la información de la manera más semejante posible a los deseos de la empresa. Como apenas se ha participado en el diseño de la interfaz, se muestran a continuación las capturas enviadas, sin justificar las mismas debido a que se desconocen las razones.

Cabe señalar que por motivos técnicos o logísticos quizá no sea posible adaptar las interfaces en su totalidad a lo especificado por el cliente. Estas posibles discrepancias, no obstante, solo afectarán a la distribución de los contenidos y no a su funcionalidad.

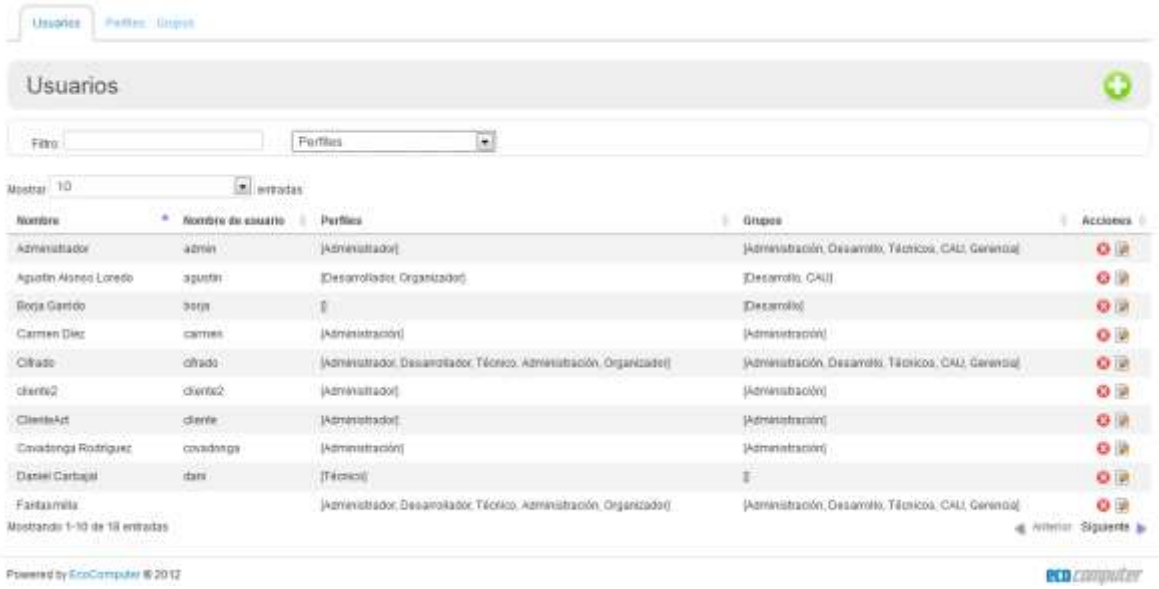
6.5.1 Intranet



Interfaz 6.1: Página personal

La única interfaz reseñable de la Intranet es la página personal del usuario, en ella podemos distinguir los datos de usuario, del cliente asociados y diferentes acciones que el usuario puede realizar. No se hace hincapié en la página de Login pues no aporta nada especial.

6.5.2 Módulo de gestión de usuarios

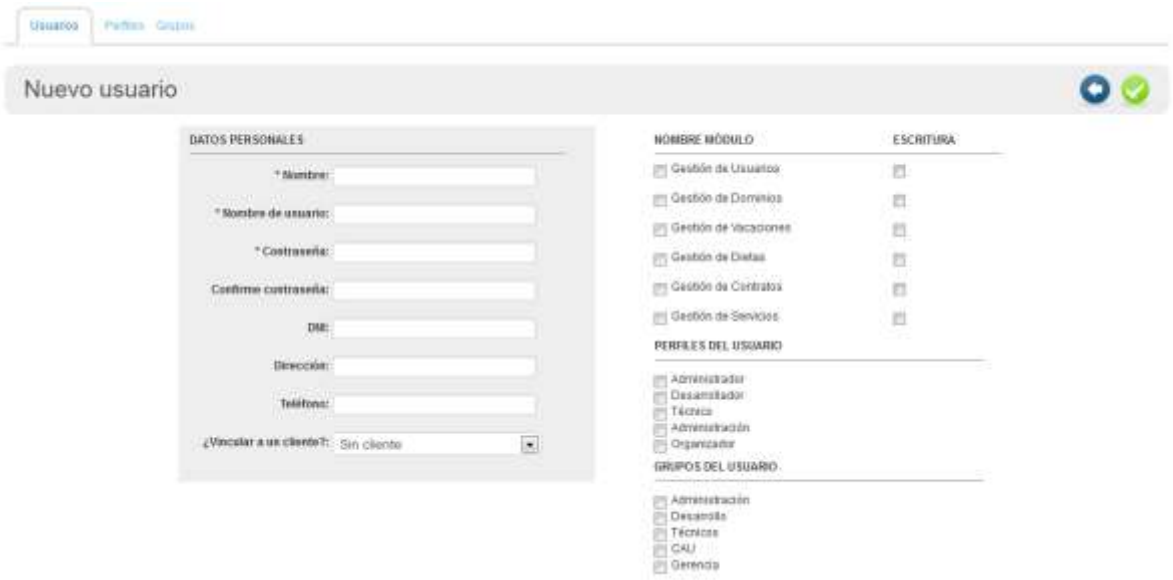


Interfaz 6.2: Listado de usuarios

Todas las páginas de listado son similares y por ello no se pondrán las correspondientes a perfiles y grupos.

Como se puede observar, en la página tenemos una lista de pestañar para navegar entre módulos, una barra de herramientas con el nombre del módulo en el que nos encontramos y botones que nos permiten realizar acciones sobre los elementos del módulo. Un filtro de búsqueda que funciona sobre la tabla de listado y la propia tabla.

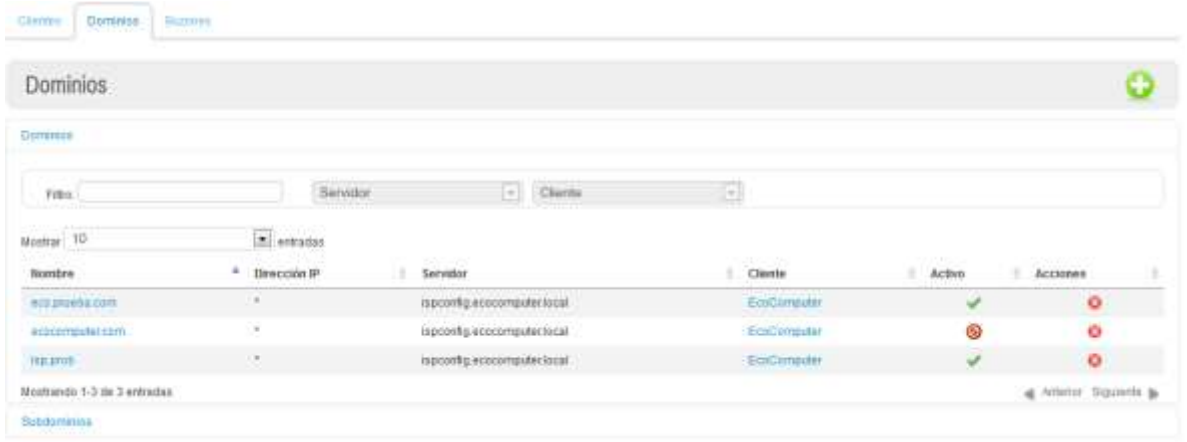
Dentro de la tabla podemos pinchar dentro de las filas para editar al usuario, esto desde luego no resulta usable, sin embargo es un requisito impuesto por la empresa, en la columna acciones podemos eliminar o cambiar la contraseña del usuario que seleccionamos.



Interfaz 6.3: añadir usuario

Al igual que ocurre con el listado, las páginas de adición son todas similares, un formulario, en muchos casos dividido en dos partes y con botón de cancelación y aceptación en la barra de herramientas.

6.5.3 Módulo de gestión de usuarios

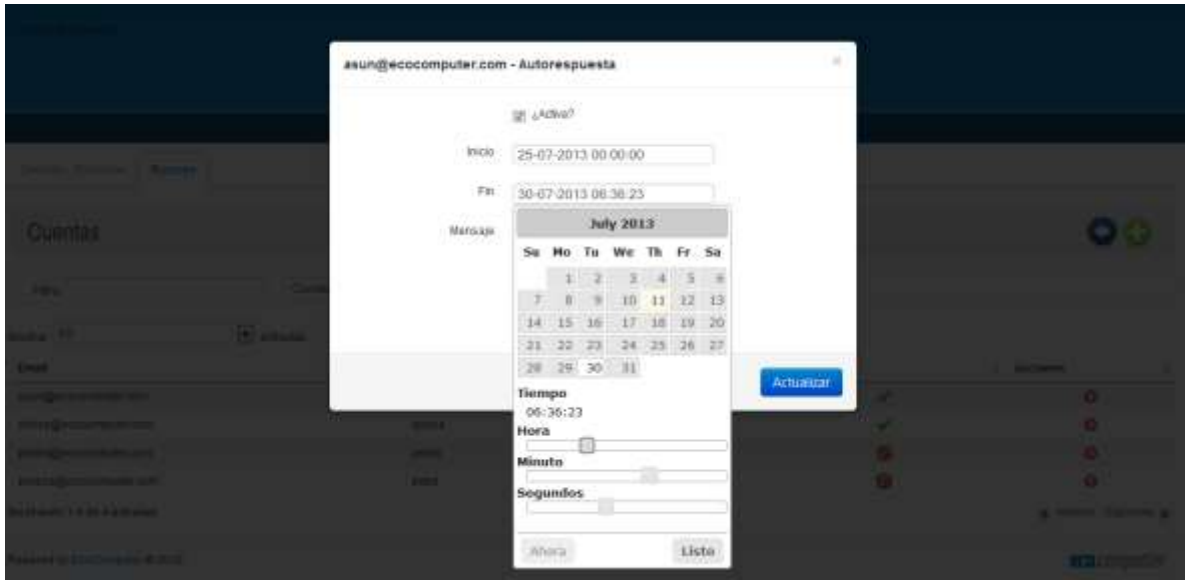


Interfaz 6.4: Listado dominios

Como vemos estás páginas tampoco difieren en exceso de las de los otros módulos, esto es debido a que con motivo de guardar consistencia entre sí todos los módulos han sido desarrollados de manera similar. Cabe destacar en esta interfaz que presionando sobre la columna activa se modifica el estado de los dominios y que las listas de dominios y subdominios están dentro de listas desplegables para facilitar su lectura.

Interfaz 6.5: Ver cliente

Esta es la interfaz de la página web de explorar cliente, como podemos ver, se muestran todos los datos relativos al cliente en el sistema así como todos los dominios asociados. En la barra de herramientas se proporciona un enlace al servidor ISPConfig de manera que el usuario pueda acceder a él de manera rápida y sencilla en caso de necesidad.



Interfaz 6.6: Configurar autorespuesta

Por último se hace reseña sobre la manera de configurar la autorespuesta, para ello se utiliza un dialogo modal que impide al usuario interactuar con el fondo de la pantalla, esto es debido a que la configuración es tan pobre que no se veía necesario la creación de una página para el formulario, por lo que se tomó la decisión de realizarlo así.

Este mismo método se utiliza en otros sitios como el cambio de contraseña del usuario desde el módulo de usuarios.

6.6 Especificación Técnica del Plan de Pruebas

6.6.1 Pruebas Unitarias

Se realizarán pruebas unitarias sobre las clases del dominio que realiza las operaciones importantes de agregado de datos, modificación y demás casos de uso relevantes. Con ello pretendemos corroborar que los procesos serealizan correctamente, sin errores bajo los diferentes escenarios y que el resultado final es el esperado.

No se han utilizado herramientas de test como JUnit debido entre otras cosas a la falta de tiempo y a que las pruebas no son lo suficiente complejas como para necesitarlos, pues el plan sería poco más que comprobar que las operaciones de base de datos se hacen de manera correcta.

6.6.2 Pruebas de Integración y del Sistema

Mediante estas pruebas de integración se pretende comprobar que la combinación de cada módulo individual en el total funciona como es esperado. Se realizarán las definidas previamente en la fase de análisis. Habrá que tener en cuenta que las pruebas de visualización de interfaz son más directas y basadas en la experiencia del usuario: se comprobará su buen funcionamiento y eficacia de manera directa, observando la interfaz y los posibles cambios que en ella provoquen las operaciones de los demás módulos.

Al haber desarrollado un módulo sobre el que trabajarán otros alumnos la mayoría de las pruebas las realizarán ellos por medio del método ensayo y error, ya que al integrarlo con su sistema detectarán fallos que habrán de ser seleccionados.

6.6.3 Pruebas de Usabilidad y Accesibilidad

Considerando la naturaleza puramente web de esta aplicación y que está supeditada a los requisitos, se prioriza la adaptación a las preferencias del cliente por encima de la accesibilidad y usabilidad. No obstante, se realizarán las comprobaciones propuestas por el WCAG para intentar obtener la máxima accesibilidad posible sin sacrificar los requisitos del cliente en la medida de lo posible.

Por esto no se ve la necesidad de desarrollar un plan de pruebas de usabilidad para usuarios de distintos niveles, pues esta aplicación será utilizada puramente por la empresa y sus componentes y en menor medida por algún cliente pero solo teniendo acceso a acciones muy reducidas.

6.6.4 Pruebas de Rendimiento

Aunque el sistema se sepa utilizado por pocos usuarios de manera concurrente, se realizarán pruebas de acceso sobre la aplicación y del tiempo de respuesta de la misma, teniendo en cuenta que muchos usuarios podrían verse agobiados por tiempos de espera muy grandes.

Capítulo 7. Implementación del Sistema

7.1 Estándares y Normas Seguidos

7.1.1 XTML 1.1

Acrónimo en inglés de eXtensible Hypertext Markup Language, se trata de un lenguaje de marcado pensado para ser el estándar que sustituye a HTML. Concretamente es la versión XML de HTML, tiene la misma funcionalidad pero cumple con las especificaciones más estrictas de XML. Tiene como objeto la obtención de una web semántica, en la que estén claramente separadas la información a mostrar de la forma de mostrarla.

Durante el desarrollo de la aplicación se ha intentado seguir lo máximo posible las normas establecidas por este estándar. Sin embargo, por la naturaleza de Struts2 y los deseos del cliente es muy posible que no se consiga un marcado que cumpla totalmente con la norma. Al no ser un requisito del cliente no se perseguirá la total perfección del estándar.

7.1.2 CSS 2

Acrónimo en inglés de Cascading Style Sheets, u “hojas de estilo en cascada”, es un lenguaje formal cuya función es la definir la presentación de un documento escrito en HTML o XML. Dichas hojas se pueden aplicar mediante un documento CSS externo llamado desde el archivo asociado, desde una hoja de estilo interna en el mismo documento, o insertando las propiedades CSS directamente dentro de las etiquetas HTML, si bien la opción recomendada por ser la más potente es la primera de ellas. Aplicando hojas de estilo conseguimos las siguientes ventajas:

- Control centralizado de la presentación de un sitio web, agilizando considerablemente su actualización.
- Posibilidad de que el usuario pueda especificar desde el navegador su propia hoja de estilos para cubrir algunas posibles necesidades como, por ejemplo, aumentar el tamaño del texto o cambiarlo de color.
- Las páginas pueden disponer de diferentes hojas de estilo en función del dispositivo que las lea. Así, la visualización de la web no será la misma desde un ordenador de sobremesa que en un dispositivo móvil, por ejemplo.
- Aumenta la legibilidad del código HTML por contener menos información, además de reducir considerablemente el tamaño del documento.

7.1.3 Estándares de programación

Pese a que no es necesario para la correcta ejecución del código fuente, sí que es muy recomendable aplicar los estándares de cada lenguaje de programación utilizado. Seguir las normas dictadas aporta principalmente mantenibilidad, entendimiento y legibilidad del código. Esto permite que la aplicación sea más fácil de entender y mantener por el propio autor original del código u otros programadores que deban hacerlo posteriormente.

Durante el desarrollo del proyecto se seguirán los estándares propios de cada lenguaje (organización física y lógica de ficheros, correcta declaración de objetos, adecuada indentación y separación…).

7.1.4 Modelo Vista Controlador

Este patrón de arquitectura de software se encarga de separar la los datos de la aplicación, la lógica de negocio y la interfaz de usuario que representa la información y recoge las interacciones del usuario. Aplicar este patrón supone aplicar las ideas de reutilización de código y separación de conceptos, lo que facilita el desarrollo y posterior mantenimiento de una aplicación software.

La realización del proyecto se ha planteado bajo la propuesta de este patrón, como ya se ha visto en los capítulos anteriores. Por un lado las JavaServer Pages muestran la información (vista), las Action recogen las interacciones del usuario (controlador) y los Manager, Service, DAO y demás clases se encargan de la lógica de negocio (modelo).

7.1.5 Patrón fachada

Aunque en un principio puede parecer que el patrón fachada no tiene aplicación en este proyecto, supone un gran gado de flexibilidad para el mismo, pues su objetivo es proporcionar a las distintas capas una fachada de manera que las capas superiores sean ajenas a la lógica interna de las clases, lo que a su vez permite ampliar dicha lógica o cambiarla por completo sin que las capas superiores se den cuenta de ello.

7.1.6 Objeto de Acceso a Datos (DAO)

Utilizar objetos de acceso a datos significa agregar una capa nueva a la capa de modelo del modelo vista controlador, separando esta en negocio y acceso a datos (persistencia). Su utilización viene motivada nuevamente por la flexibilidad, ya que aplicando una fachada a esta nueva capa se permite que puedan coexistir varias implementaciones distintas de estos objetos y que utilicen distintas formas de acceso a datos.

7.1.7 Método de Factorías

Aunque con una primera mirada al código puede parecer que no se utilizan factorías para la obtención de objetos, esté método viene intrínseco con la utilización de Spring para la inyección de dependencias.

Se escribirá la configuración en un fichero XML en el cual se definirán los beans y la implementación que deberá inyectarse en las clases que lo especifiquen, esto es el objetivo principal de utilizar factorías además la configuración de XML da soporte a utilizar el patrón singleton, de manera que solo se cree una instancia de las clases para ser utilizada por aquellas que la implementen.

7.2 Lenguajes de Programación

7.2.1 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un código intermedio, aunque la compilación en código máquina nativo es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución.

La implementación original y de referencia del compilador, la máquina virtual (JVM) y las librerías de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Java es el lenguaje de programación utilizado en la mayor parte de lógica del proyecto que nos atañe. Sin embargo se utilizan varios frameworks del lenguaje de manera que se facilite la implementación del proyecto y se evite cometer errores de implementación que sin usarlos sería posible que se diesen.

7.2.1.1 *Struts*

Struts es una herramienta de soporte para el desarrollo de aplicaciones web bajo el patrón modelo-vista-controlador para Java. Su utilización permite reducir considerablemente el tiempo de desarrollo, separando automáticamente las capas de interfaz de usuario, lógica de negocio y el acceso a datos. Esto lo realiza mediante Servlets, que se encargan de realizar las operaciones desencadenadas por los Actions del usuario en la interfaz HTML.

Este software libre fue creado originalmente por Craig McClanahan y donado a la Apache Foundation en el año 2000, pero no fue hasta 2005 cuando se convirtió en un proyecto de alta importancia para Apache. La versión empleada en este proyecto es la 2.2.1 por cuestiones de compatibilidad con el resto de software del proyecto.

Con el fin de agilizar, facilitar y asegurar el correcto desarrollo del proyecto y su separación en capas, se emplearán las librerías de Struts en el entorno de desarrollo eclipse.

7.2.1.2 *Spring Framework*

Spring es un conjunto de herramientas para el desarrollo de aplicaciones en Java. Aunque Spring no impone ningún modelo de programación específico, se ha popularizado entre los programadores al ser considerado una alternativa al modelo de Enterprise JavaBeans.

Sus módulos proveen una gran variedad de servicios, técnicas y paradigmas, como un contenedor de Inversión de Control, programación orientada a aspectos, acceso a datos, gestor transaccional, modelo-vista-controlador, framework de acceso remoto, convención sobre configuración, procesamiento por lotes, autenticación y autorización, administración remota, mensajería y herramientas de testado.

Se trata de un software libre desarrollado por Rod Johnson cuya primera versión fue lanzada en 2004. En el desarrollo del proyecto se utilizará la versión 2.2.1 distribuida junto con las librerías de Struts.

Las librerías de Spring se incorporan en el entorno de desarrollo de eclipse para proporcionar de manera más directa las herramientas que ofrece. Concretamente, se utiliza en combinación con Struts para proporcionar las ya citadas funcionalidades de inyección de dependencias.

Aunque se explicará más adelante, durante el desarrollo del proyecto se llegó alternativamente a ver Spring como una alternativa más acertada para su desarrollo, sin embargo la falta de tiempo y la dependencia de otros compañeros hizo que no fuese viable cambiar por completo la implementación del proyecto.

7.2.1.3 *Java Database Connectivity*

Más conocido por sus siglas JDBC, se trata de una API que permite realizar de manera sencilla operaciones sobre una base de datos empleando el lenguaje Java, independientemente del sistema operativo y base de datos a la que se quiera acceder. Una vez establecida la conexión mediante los manejadores pertinentes, se puede realizar cualquier tipo de tarea con la base de datos, siempre y cuando se tengan en la misma los permisos adecuados, utilizando el conjunto de objetos e interfaces que pone a disposición del programador

Su desarrollo depende de Sun Microsystems puesto que se integra en la Java Development Kit desde su versión 1.1 de febrero de 1997. Su actual versión, JDBC 4.1, está incluida en la Java SE 7 de julio de 2011.

La utilización de las librerías de JDBC en el desarrollo del proyecto se limita a la conectividad de la aplicación con la base de datos SQL Server 2005 para solicitar o escribir información.

Nuevamente y al igual que ocurre con Struts, durante el desarrollo del proyecto se tomó como una mala elección ya que de haber utilizado alguna implementación de JPA como Hibernate y todas sus herramientas de compatibilidad para Spring como Spring-Data se hubiese simplificado en gran medida todo el tema del acceso a datos. Aunque en un principio se puede pensar que el uso de fachadas y el patrón DAO permite exactamente el tener varías configuraciones el hecho principal que lo imposibilito es el formar parte de una serie de módulos que han de ser mantenidos por la misma persona y por lo cual se intentó que no difiriesen demasiado unos de otros.

7.2.2 JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se trata de un lenguaje orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Aunque existe una forma de JavaScript ejecutada en el servidor, normalmente se ejecuta en la máquina del cliente a través del navegador web, otorgando funcionalidad extra a aquellas páginas web que tengan componentes desarrollados con este lenguaje. También es significativo su uso en aplicaciones externas a la web, como widgets de escritorio o documentos PDF.

Aunque JavaScript pueda ser relacionado con el lenguaje Java, lo cierto es que no es así: aunque es de Java de quien adopta nombres y convenciones, tienen semánticas y propósitos diferentes, siendo el lenguaje C con quien comparte una sintaxis similar. También tiene influencias de otros lenguajes como Perl, Self o Python. Fue desarrollado originalmente por Brendan Eich de Netscape Communications Corp. en 1995 bajo el nombre de "Mocha", siendo el término "JavaScript" una marca registrada de Oracle Corporation. Su última versión estable es la 1.8.5 de marzo de 2011.

JavaScript se utiliza para otorgar dinamismo y funcionalidad extra a las JavaServer Pages que comprenden el proyecto. Como módulos adicionales, directamente vinculados a JavaScript, se emplea también:

7.2.2.1 *JQuery*

jQuery es una biblioteca, creada inicialmente por John Resig y presentada en enero de 2006, que simplifica notablemente la manera de interactuar con documentos HTML, su árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción AJAX, entre otras.

En este proyecto se utiliza la versión 1.9.1 para facilitar y mejorar la ejecución del JavaScript empleado en las JavaServer Pages.

7.2.2.2 *JQueryUI*

jQueryUI se trata de una biblioteca creada como adición a jQuery, anunciada en septiembre de 2007, que le otorgan un conjunto de plugins, widgets y efectos visuales para mejorar la interacción del usuario con la página web.

La aplicación de su versión 1.10.2 en este proyecto pasa por añadir calendarios desplegables, tooltips más ricos, efectos visuales varios…

7.2.2.3 *JSON*

El formato JSON o JavaScript Object Notation surge como una manera de transmitir objetos, basada en texto y entendible por humanos. Su utilización en este proyecto pasa por varios puntos, desde utilizarlo para obtener información de actions que pueda ser procesada por

JavaScript para rellenar automáticamente datos en las JSPs. Como transmitir objetos entre dos lenguajes de programación tan distintos como Java y PHP.

7.2.3 PHP

PHP es un lenguaje de script ejecutado en el lado del servidor (Server-Side) diseñado originalmente como un lenguaje para el desarrollo web que sin embargo se ha ido abriendo paso, convirtiéndose finalmente en un lenguaje de propósito general.

Creado originalmente en 1995 por Rasmus Lerdorf el control de implementación está ahora a cargo de “The PHP Group”.

Su utilización en el proyecto viene condicionada por la necesidad de comunicación con el software de terceros ISPConfig, el cual ofrece para la comunicación una capa de servicios SOAP, sin embargo se reserva el ofrecer el descriptor WSDL de los mismos y de ofrecer una documentación relativamente extensa sobre como consumir dichos servicios.

7.3 Herramientas y Programas Usados para el Desarrollo

7.3.1 Eclipse Juno

Eclipse es un entorno de desarrollo multiplataforma utilizado para la realización de proyectos de cierta envergadura, ya que da soporte a una gran cantidad de lenguajes de programación, como son C, C++, Fortran, Java, Perl, PHP, Python o Ruby, entre otros, gracias a las herramientas de desarrollo que incorpora y la posibilidad de instalar plugins adicionales.

El entorno de desarrollo integrado emplea módulos que pueden ser activados y desactivados en función de las necesidades de cada usuario, lo que le confiere una ligereza de ejecución considerable. Eclipse provee al programador de una serie de frameworks para realizar una gran cantidad de tareas relacionadas con el modelado y desarrollo de aplicaciones, además de un compilador interno de Java muy eficaz.

Se trata de una aplicación de software libre mantenida por la Eclipse Foundation, fundada en 2003, y su comunidad de usuarios, pese a que inicialmente fue producto de un proyecto de IBM. Su última versión estable, 4.2 y llamada Juno, es la empleada en el desarrollo de este proyecto.

Eclipse es la herramienta elegida para el desarrollo del proyecto, debido a que soporta todas las tecnologías y librerías necesarias, su ligereza, eficacia y la familiaridad del programador con el entorno.

7.3.2 SQL Server Management Studio Express

La aplicación SQL Server Management Studio Express es una herramienta empleada para configurar, manejar y administrar todos los componentes de una base de datos Microsoft SQL Server. Soporta tanto la entrada de datos por comandos como mediante las interfaces gráficas proporcionadas.

Fue desarrollada por el propio equipo de Microsoft y lanzada conjuntamente con SQL Server 2005. En este proyecto se utiliza la versión 9 “Express”, gratuita, puesto que es la última compatible con la versión del servidor de base de datos empleado. El motivo de su empleo aquí es obvio: la fácil administración de la base de datos del sistema y la generación de diagramas de la misma.

7.3.3 Enterprise Architect

La aplicación Enterprise Architect, desarrollada por Sparx Systems, es una completa herramienta con la que crear todo tipo de diagramas UML. Utiliza la última especificación de UML y dispone de gran cantidad de funcionalidades extra además de las posibilidades de diseño que ofrece. Se trata de una herramienta comercial, si bien existe una versión de prueba de 30 días de uso.

Incluye soporte para los diagramas de comportamiento, compuestos por casos de uso, actividades, estado, interacción, secuencia y comunicación; además de diagramas estructurales como lo son los de paquetes, clases, objetos, composición, componentes y despliegue. Entre otras características, ofrece la creación automática de código fuente en más de diez lenguajes diferentes partiendo de los diagramas, y viceversa; reportes en formato HTML y RTF; manejo de requisitos; integración con entornos de desarrollo como Microsoft Visual Studio o Eclipse…

Enterprise Architect ha sido usado en el desarrollo de este proyecto como herramienta de modelaje de la mayoría de diagramas, los cuales se pueden ver en las diferentes secciones de esta documentación. Concretamente, la versión utilizada de la aplicación ha sido la 6.0.

7.3.4 Microsoft Office Project

Project se trata de una aplicación desarrollada por Microsoft cuya función es la de administrar proyectos para poder mantener un control sobre el trabajo, la evolución del desarrollo, los plazos temporales y las finanzas, entre otros. Se trata de un software comercial que fue creado para DOS en 1984 a partir de las especificaciones de Alan M. Boyd de Microsoft, aunque fue una empresa de Seattle quien lo desarrolló para que Microsoft comprase los derechos en 1985.

Entre sus tareas se pueden destacar la posibilidad de organizar diferentes tareas, personal dedicado a cada una y el presupuesto previsto; la visualización y gestión del progreso, solucionando problemas del proceso y prediciendo posibles escenarios; y el control de las finanzas de los proyectos pudiendo elaborar presupuestos

Se ha utilizado la versión 2010 (12.0) para realizar la planificación temporal del proyecto, así como la definición y duración de sus diferentes fases. El resultado de ello puede verse en el capítulo 4 de esta documentación.

7.4 Creación del Sistema

A continuación se enumeran y explican brevemente algunos de los problemas más importantes encontrados durante el desarrollo del proyecto, y qué se hizo para solucionarlos.

7.4.1 Problemas Encontrados

7.4.1.1 *Reducción desmesurada del alcance del proyecto*

Uno de los principales problemas encontrados durante el desarrollo del siguiente fue la reducción en alcance del mismo, esto se debió a durante la fase de análisis el descubrimiento de un software de terceros denominado ISPConfig. Este software cubría prácticamente todo el alcance que se pretendía con el proyecto y se propuso su implementación dentro de la empresa, produciendo que la mitad del proyecto perdiese su sentido. Si bien es cierto que se prosiguió con el desarrollo del proyecto este cambio casi por completo, pues no se veía necesario reinventar la rueda. Sin embargo algo que si se buscaba y era la integración de todos los sistemas dentro de la Intranet seguía en pie, por lo que se decidió continuar con el proyecto enfocándolo a proporcionar una Intranet reforzada sobre la cual se integrasen tanto los módulos desarrollados por otros alumnos como de una manera mínima la interacción con el software antes mencionado.

Obviamente esto supuso un gran problema pues ya no era momento de descartar el proyecto y por lo tanto digamos que este se transformó en lo ya comentado en el párrafo anterior. Las decisiones de este tipo durante el desarrollo del proyecto suponen un riesgo y por lo tanto pueden producir nuevos problemas como en este caso un resultado final quizás no tan ambicioso como el que se perseguía en un principio.

7.4.1.2 *Retrasos por cambios de los requisitos*

Uno de los problemas intrínsecos del desarrollo de un proyecto para un cliente son los cambios de requisitos. Aunque se intentó hacerles frente mediante la utilización de una metodología ágil para el desarrollo del proyecto la cual amortiguaba en gran medida dichos cambios y por lo tanto retrasos, sí que se produjo un cambio significativo al poco de comenzar con el proyecto.

En un principio todos los proyectos se plantearon como individuales y en acuerdo con el resto de los compañeros se decidió realizar una integración de todos ellos en un mismo sistema, esto se le propuso a la empresa la cual acepto el cambio, lo que desemboco en que muchos de los aspectos que ya se tenían desarrollados en el módulo de usuarios como los controles de acceso o sin ir más lejos las acciones encargadas de identificarlo debían ser hechas de manera más general para poder ser aprovechadas por todos los módulos. Esto produjo que se desechase mucho del trabajo que se llevaba realizado hasta la fecha y la aparición de nuevos requisitos tanto para mí como para el resto de compañeros.

No siendo el cambio más significativo explicado anteriormente el único, si es cierto que en los otros casos de cambios de requisitos muchas veces la metodología ágil los amortiguaba y por lo tanto no se puede hablar de que produjesen grandes retrasos.

7.4.1.3 *Indisponibilidad de los recursos proporcionados por el cliente*

Realizar un proyecto para una empresa de informática conlleva tener que contar con ciertos recursos tanto personas como hardware y software que la empresa debe proporcionar.

En el caso de las personas, son trabajadores de la empresa y tienen sus obligaciones por lo que no disponen de todo el tiempo que sería necesario para dedicarse al proyecto. Esto ocasionó que las reuniones tuviesen que hacerse con un tiempo mayor de periodicidad, incrementando la posibilidad de que surjan cambios. De igual manera las pruebas de los módulos que recaían en gran parte sobre ellos por ser a su vez los usuarios finales a veces no se podían llevar a cabo según la planificación.

En cuanto a los recursos Hardware, el cliente debía encargarse de proporcionarnos un servidor de pruebas y otro de producción, pues no era posible saber si el proyecto avanzaba como se requería sin probarlo en conjunto con todos los módulos de los demás alumnos. Finalmente esto no pudo hacerse consiguiendo solamente un servidor de pruebas-producción, al ser este proyecto el encargado de preparar el entorno común me quedo asignada la carga de mantenimiento de dicho servidor (lo que como veremos más adelante produjo más retrasos). El tener solo un servidor ocasionaba a su vez grandes percances pues ciertos cambios en el proyecto producían que su integración con módulos de otros alumnos se viese afectada, lo que ocasionaba que estos dejasen de funcionar de forma correcta, como el cliente utilizaba este servidor para comprobar el estado de desarrollo de los proyectos no se podía dejar nunca un problema a medias pues podía afectar directamente a la planificación de los compañeros.

Finalmente, los recursos software, como es obvio la empresa tienen versiones comerciales de ciertos programas a los que se pedía que nos adaptásemos, concretamente SQLServer, para el desarrollo hubo que conformarse con una versión antigua pues era gratuita, pero obviamente no proporcionaba las mismas prestaciones que la versión de la empresa, una de las que más se echó en falta era la capacidad de exportar a un script la estructura de la base de datos y todos sus datos almacenados.

7.4.1.4 *Aparición de clientes internos*

Como ya se ha comentado anteriormente el alcance de este proyecto se vio relegado casi por completo a la construcción de la Intranet y a la integración de todos los módulos desarrollados por otros alumnos, a su vez estos alumnos dependían del módulo de usuarios desarrollado en este proyecto y de toda la estructura de la Intranet.

Esto ocasionó que hubiese que empezar a tratar a los compañeros como clientes internos, pues tenían requisitos que debían ser aplicados a la Intranet para el desarrollo de sus módulos, había que ofrecerles cierto entrenamiento sobre cómo utilizar los componentes que se les proporcionaba y finalmente había que tener en cuenta sus pruebas del sistema. Todo esto

acabo convirtiéndose en un sistema de desarrollo a petición prácticamente, pues los cambios había que hacerlos de manera rápida y efectiva para no afectar a su planificación, lo que acababa afectando a la mía.

7.4.1.5 *Integración de software de terceros*

Uno de los grandes baches de este proyecto ha sido la integración de software de terceros, concretamente de ISPConfig, debido sobre todo a que este software no está especialmente preparado para su utilización desde otro. Si bien es cierto que proporcionan una capa de servicios WEB SOAP estos no son suficientes para una integración completa, provocando que se hayan tenido que hilar varios de estos servicios para poder conseguir cosas tan sencillas como listar todos los dominios. Otro de los problemas es que no ofrecen un descriptor de servicios WSDL, por lo que desarrollar un cliente para los mismos se convertía en una tarea muy ardua y finalmente la falta de una documentación extensa sobre cómo utilizarlos.

Este tema produjo que se perdiese mucho tiempo dando palos de ciego averiguando como consumir los servicios SOAP del servidor desde Java, llegando a un punto en que se decidió esto como inviable y se echó mano de lo ofrecido por el servició que eran unos ejemplos de consumo en PHP.

Para poder ejecutar PHP en Tomcat7 es necesaria cierta configuración tanto del servidor como de la aplicación pues es necesario un componente que transforme dicho código en algo que entienda el servidor, en este caso Java. Está configuración así como el estudio de que herramienta era la más correcta utilizar conllevo nuevamente unos retrasos con los que no se contaban.

Surgía entonces un nuevo problema y era la integración con lo que ya se tenía, pues las páginas PHP no podían ser tratadas de igual manera por el sistema, finalmente la decisión que se tomo era tener los ficheros PHP almacenados en un subdirectorio del proyecto y ejecutarlo mediante la propiedad de Java “exec” que permite ejecutar ordenes como si se hiciese desde línea de comandos.

7.4.1.6 *Asignación de tareas no contempladas en un principio*

En este apartado recae el hecho de haber tenido que hacerse cargo de la administración del servidor común de todos los alumnos, ya que es algo inherente al propio funcionamiento del mismo el que de vez en cuando haya problemas, haya que actualizar software, instalar software nuevo y tener a más de una persona haciendo esto conlleva en la mayoría de los casos más problemas.

7.4.2 Descripción Detallada de las Clases

Para facilitar la lectura se dividirá nuevamente toda la descripción de las clases en los diferentes módulos que se han implementado y a su vez en paquetes.

7.4.2.1 *Intranet*

7.4.2.1.1 Modelos

7.4.2.1.1.1 *Cliente*

Nombre	Tipo	Extiende	Implementa
Cliente	Class		Serializable
Descripción			
Clase que representa el objeto cliente dentro de la aplicación			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public		Cliente	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		int	idCliente
private		String	nombre
private		String	direccion
private		String	poblacion
private		String	provincia
private		String	telefono
private		String	fax
private		String	logo
Observaciones			
Se han eliminado los métodos de acceso a atributos			

7.4.2.1.1.2 *Grupo*

Nombre	Tipo	Extiende	Implementa
Grupo	Class		Serializable
Descripción			
Clase que representa el objeto grupo dentro de la aplicación			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public		Grupo	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		int	idGrupo
private		String	descripcion
private		List<Usuario>	usuarios
Observaciones			
Se han eliminado los métodos de acceso a atributos			

7.4.2.1.1.3 *Módulo*

Nombre	Tipo	Extiende	Implementa
Modulo	Class		Serializable
Descripción			
Clase que representa el objeto módulo dentro de la aplicación, indica que tipo de acceso tienen los usuarios y perfiles a un módulo concreto.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public		Modulo	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	idModulo
Private		String	nombre
Private		boolean	escritura
Observaciones			
Se han eliminado los métodos de acceso a datos			

7.4.2.1.1.4 *Perfil*

Nombre	Tipo	Extiende	Implementa
Perfil	Class		Serializable
Descripción			
Clase que representa el objeto perfil dentro de la aplicación.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public		Perfil	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	idPerfil
Private		String	descripcion
Private		List<Usuario>	usuarios
Private		List<Modulo>	modulos
Observaciones			
Se han eliminado los métodos de acceso a datos			

7.4.2.1.1.5 *Sesión*

Nombre	**Tipo**	**Extiende**	**Implementa**
Sesion	Class		Serializable
Descripción			
Clase que representa una sesión de usuario			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public		Sesion	int idUsuario
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private	static final	int	LOG_ERROR
private	static final	int	LOG_WARNING
private	static final	int	LOG_INFO
private		int	idSesion
private		int	idUsuario
private		Timestamp	horaInicio
private		Timestamp	horaFin
private		Timestamp	ultimoEvento
private		Log	log
Observaciones			
Se han eliminado los métodos de acceso a datos			

En su interior se encuentra la clase privada Log:

Nombre	**Tipo**	**Extiende**	**Implementa**
Log	Class		Serializable
Descripción			
Clase que representa una entrada de log dentro de la sesión.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public		Log	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		String	evento
private		String	pagina
private		int	nivel
Observaciones			
Se han eliminado los métodos de acceso a datos			

7.4.2.1.1.6 *Usuario*

Nombre	Tipo	Extiende	Implementa
Usuario	Class		Serializable
Descripción			
Clase que representa un usuario dentro de la aplicación.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public		Usuario	
Public	Usuario	deserializar	String usuarioSerializado
Public	boolean	getAccesoModulo	int idModulo
Public	boolean	getEscrituraModulo	int idModulo
Public	boolean	pertenecePerfil	int idPerfil
Public	String	serializar	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	idUsuario
Private		String	nombre
Private		String	login
Private		String	password
Private		String	nif
Private		String	direccion
Private		String	telefono
Private		Cliente	cliente
Private		List<Perfil>	perfiles
Private		List<Grupo>	grupos
Private		List<Modulo>	modulos
Private		Sesion	sesion
Observaciones			
Se han eliminado los métodos de acceso a datos			

7.4.2.1.2 Paquete de persistencia

7.4.2.1.2.1 *ConnectionManager*

Nombre	Tipo	Extiende	Implementa
ConnectionManager	Class		
Descripción			
Clase utilizada para gestionar las conexiones a la base de datos.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
private		ConnectionManager	
public static	Connection	getConnection	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		Connection	connection
private	static	ConnectionManager	moDbConnect
private		DataSource	dataSource
Observaciones			
Tiene un constructor privado por que solo se puede acceder de manera estática.			

7.4.2.1.2.2 *GenericDAO*

Nombre	Tipo	Extiende	Implementa
GenericDAO	Class		
Descripción			
Clase de la que heredan todos los DAO, contiene métodos y atributos comunes.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
protected		closeConnections	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
protected		Connection	con
protected	static	PreparedStatement	ps
protected		ResultSet	rs
Observaciones			
-			

7.4.2.1.2.3 *ClienteDAO*

Nombre	Tipo	Extiende	Implementa
ClienteDAO	Class	GenericDAO	ClienteDataService
Descripción			
Clase que gestiona la persistencia del objeto Cliente.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	createCliente	Cliente cliente
Public	int	deleteCliente	int idCliente
Public	int	updateCliente	Cliente cliente
Public	Cliente	getClienteById	int idCliente
Public	List<Cliente>	getClientes	
Public	List<String>	getNombresCliente	
Private	Cliente	extractClienteFromResultSet	Cliente cliente ResultSet rs Connection con
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.2.4 *GrupoDAO*

Nombre	Tipo	Extiende	Implementa
GrupoDAO	Class	GenericDAO	GrupoDataService
Descripción			
Clase que gestiona la persistencia del objeto Grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	createGrupo	Grupo grupo
Public	int	deleteGrupo	int idGrupo
Public	int	updateGrupo	Grupo grupo
Public	Grupo	getGrupoById	int idGrupo
Public	Grupo	getGrupoByDescripcion	String descripcion
Public	List<Grupo>	getGrupos	
Public	List<Grupo>	getGruposByDescripcion	List<String> descripciones
Public	List<Grupo>	getGruposByUsuario	int idUsuario
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.2.5 *ModuloDAO*

Nombre	**Tipo**	**Extiende**	**Implementa**
ModuloDAO	Class	GenericDAO	ModuloDataService
Descripción			
Clase que gestiona la persistencia del objeto Módulo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	Modulo	getModuloById	int idModulo
public	List<Modulo>	getModulos	
public	List<Modulo>	getModulosByNombre	List<String> nombres
public	List<Modulo>	getModulosByPerfil	int idPerfil
public	List<Modulo>	getModulosByUsuario	int idUsuario
public	List<String>	getNombresModulos	
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.2.6 *PerfilDAO*

Nombre	**Tipo**	**Extiende**	**Implementa**
PerfilDAO	Class	GenericDAO	PerfilDataService
Descripción			
Clase que gestiona la persistencia del objeto Perfil.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	int	createPerfil	Perfil perfil
public	int	deletePerfil	int idPerfil
public	int	updatePerfil	Perfil perfil
public	Perfil	getPerfilById	int idPerfil
public	Perfil	getPerfilByDescripcion	String descripcion
public	List<Perfil>	getPerfiles	
public	List<Perfil>	getPerfilesByDescripcion	List<String> descripciones
public	List<Perfil>	getPerfilesByUsuario	int idUsuario
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.2.7 *SesionDAO*

Nombre	Tipo	Extiende	Implementa
SesionDAO	Class	GenericDAO	SesionDataService
Descripción			
Clase que gestiona la persistencia del objeto Sesion.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	Sesion	saveSesion	Sesion sesion
Public	int	closeSesion	Sesion sesion
Public	int	updateSesion	Sesion sesion
Private	int	writeLog	Connection con Sesion sesion
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.2.8 *UsuarioDAO*

Nombre	Tipo	Extiende	Implementa
UsuarioDAO	Class	GenericDAO	UsuarioDataService
Descripción			
Clase que gestiona la persistencia del objeto Usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	createUsuario	Sesion sesion
Public	int	deleteUsuario	Sesion sesion
Public	int	updateUsuario	Sesion sesion
Public	Usuario	getUsuarioById	int idUsuario
Public	Usuario	getUsuarioByLogin	String login
Public	List<Usuario>	getUsuarios	
Public	List<Usuario>	getUsuariosByGrupo	int idGrupo
Public	List<Usuario>	getUsuariosByPerfil	int idPerfil
Public	List<Usuario>	getUsuariosByLogin	List<String> logins
Public	List<String>	getLoginUsuarios	
Private	Usuario	extractUsuarioFromResultSet	Usuario usuario ResultSet rs Connection con
Private	Usuario	loadDatosPersonales	Usuario usuario Connection con
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.3 Paquete de negocio

7.4.2.1.3.1 *UsuarioManager*

Nombre	Tipo	Extiende	Implementa
UsuarioManager	Class		UsuarioManagerService
Descripción			
Se encarga de gestionar las acciones relacionadas con los usuarios, llamando a unos u otros métodos para la operación.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	int	deleteUsuario	int idUsuario
public	int	saveNuevoUsuario	Usuario usuario
public	int	updateUsuario	Usuario usuario
public	Usuario	getUsuarioById	int idUsuario
public	Usuario	getUsuarioByLogin	String login
public	List<Usuario>	getUsuarios	
public	List<Usuario>	getUsuariosByLogin	List<String> logins
private	void	loadRelacionesUsuario	Usuario usuario
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteDataService	clienteDataService
private		GrupoDataService	grupoDataService
private		ModuloDataService	moduloDataService
private		PerfilDataService	perfilDataService
private		UsuarioDataService	usuarioDataService
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.1.4 Presentación

7.4.2.1.4.1 *CerrarSesionAction*

Nombre	Tipo	Extiende	Implementa
CerrarSesionAction	Class	ActionSupport	SessionAware ServletRequestAware ServletResponseAware
Descripción			
Action que se encarga de sacar al usuario de sesión y eliminar todos los datos relativos al mismo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Private	void	eliminarcookies	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		SesionDataService	sesionDataService
Private		Map<String, Object>	session
Private		HttpServletRequest	request
Private		HttpServletResponse	response
Observaciones			
-			

7.4.2.1.4.2 *ChangePasswordAction*

Nombre	Tipo	Extiende	Implementa
ChangePasswordAction	Class	ActionSupport	SessionAware
Descripción			
Action que se encarga de realizar el cambio de contraseña de un usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		UsuarioManagerService	usuarioManagerService
Private		String	confirmPassword
Private		String	oldPassword
Private		String	password
Private		Map<String, Object>	session
Observaciones			
-			

7.4.2.1.4.3 *DeterminativeAction*

Nombre	Tipo	Extiende	Implementa
DeterminativeAction	Class	ActionSupport	SessionAware
Descripción			
Action que se encarga de decidir hacia donde enviar al usuario en su página personal.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		Cliente	cliente
private		Usuario	usuario
private		Map<String, Object>	session
Observaciones			
-			

7.4.2.1.4.4 *LoginAction*

Nombre	Tipo	Extiende	Implementa
LoginAction	Class	ActionSupport	SessionAware RequestAware ResponseAware
Descripción			
Action que se encarga de validar las credenciales e introducir al usuario en sesión.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
private	void	generarCookies	Usuario usuario
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		usuarioManagerService	usuarioManagerService
private		sesionDataService	sesionDataService
private		Map<String, Object>	session
private		HttpServletRequest	request
private		HttpServletResponse	response
private		String	login
private		String	password
Observaciones			
-			

7.4.2.1.4.5 *TransactionAction*

Nombre	Tipo	Extiende	Implementa
TransactionAction	Class	ActionSupport	
Descripción			
Action que se utiliza cuando la invocación de un JSP no necesita ejecutar ningún tipo de acción.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Private	void	generarCookies	Usuario usuario
Observaciones			
-			

7.4.2.1.5 Security

7.4.2.1.5.1 *AccessMethodInterceptor*

Nombre	Tipo	Extiende	Implementa
AccessMethodInterceptor	Class	AbstractInterceptor	
Descripción			
Interceptor que se dispara cuando se intenta acceder a ciertas páginas que necesitan una carga previa de datos sin haberla realizado.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	intercept	ActionInvocation invocation
Observaciones			
-			

7.4.2.1.5.2 *LoginInterceptor*

Nombre	Tipo	Extiende	Implementa
LoginInterceptor	Class	AbstractInterceptor	
Descripción			
Interceptor que se dispara cuando se intenta acceder a la parte privada de la aplicación, comprueba si el usuario está identificado o existe algún dato del mismo en el sistema.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	intercept	ActionInvocation invocation
private	Usuario	getUsuarioFromCookie	
Observaciones			
-			

7.4.2.1.5.3 *ReadAccessInterceptor*

Nombre	Tipo	Extiende	Implementa
ReadAccessInterceptor	Class	AbstractInterceptor	
Descripción			
Interceptor que se dispara cuando se intenta acceder a una parte del módulo que requiere permisos de lectura.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	intercept	ActionInvocation invocation
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		int	idModulo
Observaciones			
-			

7.4.2.1.5.4 *WiteAccessInterceptor*

Nombre	Tipo	Extiende	Implementa
WriteAccessInterceptor	Class	AbstractInterceptor	
Descripción			
Interceptor que se dispara cuando se intenta acceder a una parte del módulo que requiere permisos de escritura.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	intercept	ActionInvocation invocation
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	idModulo
Observaciones			
-			

7.4.2.1.6 **Utils**

7.4.2.1.6.1 *Cipher*

Nombre	Tipo	Extiende	Implementa
Cipher	Class		
Descripción			
Clase utilizada para cifrar las contraseñas a la base de datos			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public static	String	cipherPassword	String password
Observaciones			
-			

7.4.2.1.6.2 *Logger*

Nombre	**Tipo**	**Extiende**	**Implementa**
Logger	Class		
Descripción			
Clase utilizada para llevar un registro de logs en la base de datos.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public		Logger	String className Map<String, Object> sesion
public		error	String evento
public		info	String evento
public		ewarning	String evento
public		updateSesion	String className String evento int nivel
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		String	pagina
private		Sesion	sesion
private		SesionDataService	sesionDataService
Observaciones			
-			

7.4.2.2 *Gestión de usuarios*

7.4.2.2.1 Negocio

7.4.2.2.1.1 *ClienteManager*

Nombre	Tipo	Extiende	Implementa
ClienteManager	Class		ClienteManagerService
Descripción			
Clase utilizada para el tratamiento de los datos relacionados con la clase Cliente.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	Cliente	getClienteById	int idCliente
Public	List<Cliente>	getClientesSistema	
Public	Map<Integer, String>	getHashMapClientes	
Public	List<String>	getNombresClientes	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		ClienteDataService	clienteDataService
Observaciones			
-			

7.4.2.2.1.2 *GrupoManager*

Nombre	Tipo	Extiende	Implementa
GrupoManager	Class		GrupoManagerService
Descripción			
Clase utilizada para el tratamiento de los datos relacionados con la clase Grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	deleteGrupo	int idGrupo
Public	int	saveNuevoGrupo	grupo Grupo
Public	int	updateGrupo	grupo Grupo
Public	Grupo	getGrupoById	int idGrupo
Public	Grupo	getGrupoByDescripcion	String descripcion
Public	List<Grupo>	getGruposSistema	
Public	List<Grupo>	getGruposByDescripcion	List<String> descripciones
Public	List<String>	getDescripcionGrupos	
Private		loadRelacionesGrupo	Grupo grupo
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		GrupoDataService	grupoDataService
Private		UsuarioDataService	usuarioDataService
Observaciones			
-			

7.4.2.2.1.3 *ModuloManager*

Nombre	Tipo	Extiende	Implementa
ModuloManager	Class		ModuloManagerService
Descripción			
Clase utilizada para el tratamiento de los datos relacionados con la clase Modulo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	List<Modulo>	getModulosByNombre	List<String> nombres
public	List<Modulo>	getModulosSistema	
public	List<String>	getNombresModulos	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ModuloManagerService	moduloManagerService
Observaciones			
-			

7.4.2.2.1.4 *PerfilManager*

Nombre	Tipo	Extiende	Implementa
PerfilManager	Class		PerfilManagerService
Descripción			
Clase utilizada para el tratamiento de los datos relacionados con la clase Perfil.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	Int	deletePerfil	int idPerfil
public	Int	saveNuevoPerfil	perfil Perfil
public	Int	updatePerfil	perfil Perfil
public	Perfil	getPerfilById	int idPerfil
public	Perfil	getPerfilByDescripcion	String descripcion
public	List<Perfil>	getPerfiles	
public	List<Perfil>	getPerfilesByDescripcion	List<String> descripciones
public	List<String>	getDescripcionPerfiles	
private		loadRelacionesPerfil	perfil Perfil
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		PerfilDataService	perfilDataService
private		UsuarioDataService	usuarioDataService
private		ModuloDataService	moduloDataService
Observaciones			
-			

7.4.2.2.1.5 *UsuarioManager*

Nombre	Tipo	Extiende	Implementa
UsuarioManager	Class		UsuarioManagerService
Descripción			
Se encarga de gestionar las acciones relacionadas con los usuarios, llamando a unos u otros métodos para la operación.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	deleteUsuario	int idUsuario
Public	int	saveNuevoUsuario	Usuario usuario
Public	int	updateUsuario	Usuario usuario
Public	Usuario	getUsuarioById	int idUsuario
Public	Usuario	getUsuarioByLogin	String login
Public	List<Usuario>	getUsuarios	
Public	List<Usuario>	getUsuariosByLogin	List<String> logins
Private	void	loadRelacionesUsuario	Usuario usuario
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		ClienteDataService	clienteDataService
Private		GrupoDataService	grupoDataService
Private		ModuloDataService	moduloDataService
Private		PerfilDataService	perfilDataService
Private		UsuarioDataService	usuarioDataService
Observaciones			
Como atributos de clase solo tiene los heredados.			

7.4.2.2.2 Presentación

7.4.2.2.2.1 *AddNuevoGrupoAction*

Nombre	Tipo	Extiende	Implementa
AddNuevoGrupoAction	Class	ActionSupport	SessionAware ModelDriven<Grupo>
Descripción			
Action encargado de comenzar la adición de un Grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
private	boolean	checkDescripcionRepetida	String descripcion
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		GrupoManagerService	grupoManagerService
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Logger	log
private		List<String>	loginUsuarios
Observaciones			
-			

7.4.2.2.2.2 *CargarFormularioNuevoGrupoAction*

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevoGrupoAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de cargar los datos necesarios para el formulario de añadir grupo nuevo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Logger	log
private		List<String>	usuarios
Observaciones			
-			

7.4.2.2.2.3 *CargarGrupoSistemaAction*

Nombre	Tipo	Extiende	Implementa
CargarGrupoAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de proporcionar los datos para visualizar un grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Private	void	loadUsuariosGrupo	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		GrupoManagerService	grupoManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		int	idGrupo
Private		Grupo	grupo
Private		List<String>	usuariosGrupo
Private		List<String>	usuarios
Private		List<Modulo>	modulos
Observaciones			
-			

7.4.2.2.2.4 *CargarGruposSistemaAction*

Nombre	Tipo	Extiende	Implementa
CargarGruposSistemaAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de proporcionar los datos para visualizar un grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		GrupoManagerService	grupoManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		List<Grupo>	grupos
Observaciones			
-			

7.4.2.2.2.5 *DeleteGrupoAction*

Nombre	Tipo	Extiende	Implementa
DeleteGrupoAction	Class	ActionSupport	SessionAware ParametersAware
Descripción			
Action encargado de iniciar el proceso de eliminación de un grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		GrupoManagerService	grupoManagerService
private		Map<String, String>	parameters
private		Map<String, Object>	session
private		Logger	log
private		int	idGrupo
Observaciones			
-			

7.4.2.2.2.6 *UpdateGrupoAction*

Nombre	Tipo	Extiende	Implementa
UpdateGrupoAction	Class	ActionSupport	SessionAware ModelDriven<Grupo>
Descripción			
Action encargado de iniciar la actualización de un grupo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		GrupoManagerService	grupoManagerService
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Logger	log
private		Grupo	grupo
private		List<String>	loginUsuarios
Observaciones			
-			

7.4.2.2.2.7 *AddNuevoPerfilAction*

Nombre	Tipo	Extiende	Implementa
AddNuevoPerfilAction	Class	ActionSupport	SessionAware ModelDriven<Perfil>
Descripción			
Action encargado de iniciar la acción de añadir nuevo perfil.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Public	void	validate	
Private	List<Modulo>	createPermisosModulos	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		PerfilManagerService	perfilManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		Perfil	perfil
Private		List<String>	loginUsuarios
Private		List<Integer>	modulosActivos
Private		List<Integer>	permisosEscritura
Observaciones			
-			

7.4.2.2.2.8 *CargarFormularioNuevoPerfilAction*

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevoPerfilAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de iniciar la acción de añadir nuevo perfil.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		ModuloManagerService	moduloManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		List<String>	usuarios
Private		List<Modulo>	modulos
Observaciones			

-

7.4.2.2.2.9 *CargarPerfilesSistemaAction*

Nombre	**Tipo**	**Extiende**	**Implementa**
CargarPerfilesSistemaAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar todos los datos de perfiles del sistema.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		PerfilManagerService	perfilManagerService
private		Map<String, Object>	session
private		Logger	log
private		List<Perfil>	perfiles
Observaciones			
-			

7.4.2.2.2.10 *CargarPerfilSistemaAction*

Nombre	Tipo	Extiende	Implementa
CargarPerfilSistemaAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar los datos de un perfil para pasarselos a la interfaz.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Private	void	loadModulosPerfil	
Private	void	loadUsuariosPerfil	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		PerfilManagerService	perfilManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		ModuloManagerService	moduloManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		int	idPerfil
Private		Perfil	perfil
Private		List<String>	usuariosPerfil
Private		List<Modulo>	modulosPerfil
Private		List<String>	usuarios
Private		List<Modulo>	modulos
Observaciones			
-			

DeletePerfilAction

Nombre	**Tipo**	**Extiende**	**Implementa**
DeletePerfilAction	Class	ActionSupport	SessionAware ParametersAware
Descripción			
Action encargado de iniciar el proceso de eliminación de un perfil.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		PerfilManagerService	perfilManagerService
private		Map<String, String>	parameters
private		Map<String, Object>	session
private		Logger	Log
private		int	idPerfil
Observaciones			
-			

UpdatePerfilAction

Nombre	Tipo	Extiende	Implementa
UpdatePerfilAction	Class	ActionSupport	SessionAware ModelDriven<Perfil>
Descripción			
Action encargado de preparar los datos de un perfil para pasarselos a la interfaz.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Public	void	validate	
Private	boolean	checkDescripcionRepetida	String descripcion
Private	List<Modulo>	createPermisosModulos	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		PerfilManagerService	perfilManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		Perfil	perfil
Private		List<String>	loginUsuarios
Private		List<Integer>	modulosActivos
Private		List<Integer>	permisosEscritura
Observaciones			
-			

AddNuevoUsuarioAction

Nombre	**Tipo**	**Extiende**	**Implementa**
AddNuevoUsuarioAction	Class	ActionSupport	SessionAware ModelDriven<Usuario>
Descripción			
Action encargado de iniciar la creación de un nuevo usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
private	boolean	checkLoginRepetido	String login
private	List<Modulo>	createPermisosModulos	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		PerfilManagerService	perfilManagerService
private		UsuarioManagerService	usuarioManagerService
private		GrupoManagerService	grupoManagerService
private		Map<String, Object>	session
private		Logger	log
private		int	idCliente
private		Usuario	Usuario
private		String	confirmPassword
private		List<String>	descripciónGrupos
private		List<String>	descripciónPerfiles
private		List<Modulo>	permisosEscritura
private		List<Modulo>	modulosActivos
Observaciones			
-			

CargarFormularioNuevoUsuarioAction

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevoUsuarioAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de iniciar la creación de un nuevo usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		PerfilManagerService	perfilManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		GrupoManagerService	grupoManagerService
Private		clienteManagerService	clienteManagerService
Private		moduloManagerService	moduloManagerService
Private		Map<String, Object>	session
Private		Logger	log
Observaciones			
-			

7.4.2.2.2.11 *CargarUsuarioSistemaAction*

Nombre	**Tipo**	**Extiende**	**Implementa**
CargarUsuarioSistemaAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar los datos de un usuario para ser usados por las JSPs.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
private	List<String>	loadGruposUsuario	
private	List<Modulo>	loadModulosUsuario	
private	List<String>	loadPerfilesUsuario	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		PerfilManagerService	perfilManagerService
private		UsuarioManagerService	usuarioManagerService
private		GrupoManagerService	grupoManagerService
private		clienteManagerService	clienteManagerService
private		moduloManagerService	moduloManagerService
private		Map<String, Object>	session
private		Logger	log
private		int	idUsuario
private		Usuario	usuario
private		List<String>	perfilesUsuario
private		List<String>	gruposUsuario
private		List<Modulo>	modulosUsuario
Observaciones			
-			

CargarUsuariosSistemaAction

Nombre	Tipo	Extiende	Implementa
CargarUsuariosSistema	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar los datos de un usuario para ser usados por las JSPs.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		List<Usuario>	usuarios
Observaciones			
-			

7.4.2.2.2.12 *ChangeUsuarioPasswordAction*

Nombre	Tipo	Extiende	Implementa
ChangeUsuarioPassword Action	Class	ActionSupport	SessionAware
Descripción			
Action encargado de iniciar el proceso de cambio de contraseña para un usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		UsuarioManagerService	usuarioManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		int	idUsuario
Private		String	newPassword
Private		String	confirmPassword
Observaciones			
-			

7.4.2.2.2.13 *DeleteUsuarioAction*

Nombre	**Tipo**	**Extiende**	**Implementa**
DeleteUsuarioAction	Class	ActionSupport	SessionAware ParametersAware
Descripción			
Action encargado de iniciar el proceso de eliminación de un usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		UsuarioManagerService	usuarioManagerService
private		Map<String, String>	parameters
private		Map<String, Object>	session
private		Logger	Log
private		int	idUsuario
Observaciones			
-			

7.4.2.2.2.14 *UpdateUsuarioAction*

Nombre	Tipo	Extiende	Implementa
UpdateUsuarioAction	Class	ActionSupport	SessionAware ModelDriven<Usuario>
Descripción			
Action encargado de iniciar el proceso de actualización de un usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Private	List<Modulo>	createPermisosModulo	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		PerfilManagerService	perfilManagerService
Private		UsuarioManagerService	usuarioManagerService
Private		GrupoManagerService	grupoManagerService
Private		clienteManagerService	clienteManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		int	idCliente
Private		Usuario	usuario
Private		List<String>	descripcionPerfiles
Private		List<String>	descripcionGrupos
Private		List<Integer>	modulosActivos
Private		List<Integer>	permisosEscritura
Observaciones			
-			

7.4.2.3 *Gestión de dominios*

7.4.2.3.1 **Modelo**

7.4.2.3.1.1 *AutoRespuesta*

Nombre	**Tipo**	**Extiende**	**Implementa**
AutoRespuesta	Class		
Descripción			
Clase encargada de servir como modelo al objeto Autorespuesta.			
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		String	endDate
private		String	startDate
private		String	message
Observaciones			
No se han incluido los getters y setters.			

7.4.2.3.1.2 *Buzon*

Nombre	**Tipo**	**Extiende**	**Implementa**
Buzon	Class		
Descripción			
Clase encargada de servir como modelo al objeto Buzon.			
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		boolean	active
private		ISPConfigClient	cliente
private		List<Cuenta>	cuentas
private		String	dominio
private		int	id
private		Servidor	servidor
Observaciones			
No se han incluido los getters y setters.			

7.4.2.3.1.3 *Cuenta*

Nombre	Tipo	Extiende	Implementa
Cuenta	Class		
Descripción			
Clase encargada de servir como modelo al objeto Cuenta.			
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		boolean	activeAutoResponse
Private		AutoRespuesta	autoRespuesta
Private		int	cuota
Private		Buzon	buzon
Private		String	email
Private		int	id
Private		String	login
Private		String	name
Private		String	password
Observaciones			
No se han incluido los getters y setters.			

7.4.2.3.1.4 *Dominio*

Nombre	Tipo	Extiende	Implementa
Dominio	Class		
Descripción			
Clase encargada de servir como modelo al objeto Dominio.			
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		boolean	active
Private		ISPConfigClient	cliente
Private		String	direccionIp
Private		int	id
Private		String	nombre
Private		Dominio	padre
Private		Servidor	servidor
Private		active	subDominio
Private		List<Dominio>	subDominios
Observaciones			
No se han incluido los getters y setters.			

7.4.2.3.1.5 *ISPConfigClient*

Nombre	Tipo	Extiende	Implementa
ISPConfigClient	Class		
Descripción			
Clase encargada de servir como modelo al objeto Cliente del servidor ISP.			
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		Cliente	cliente
private		String	company
private		String	nombreContacto
private		String	pais
private		List<Dominio>	dominios
private		String	email
private		String	internet
private		int	id
private		String	idioma
private		String	login
private		String	telefono
private		String	notas
private		String	password
Observaciones			
No se han incluido los getters y setters.			

7.4.2.3.1.6 *Servidor*

Nombre	Tipo	Extiende	Implementa
Servidor	Class		
Descripción			
Clase encargada de servir como modelo al objeto Servidor.			
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		int	id
private		String	nombre
Observaciones			
No se han incluido los getters y setters.			

7.4.2.3.2 Persistencia

7.4.2.3.2.1 *BuzonDAO*

Nombre	Tipo	Extiende	Implementa
BuzonDAO	Class		BuzonDataService
Descripción			
Clase encargada de la persistencia del objeto Buzon y sus objetos asociados.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	createCuenta	Cuenta cuenta
Public	int	createBuzon	Buzon buzon
Public	List<Cuenta>	getAllCuentas	
Public	List<Buzon>	getAllBuzones	
Public	Cuenta	getCuentaById	int idCuenta
Public	Buzon	getBuzonById	int idBuzon
Public	void	setAutorRespuesta	Cuenta cuenta
Private	Cuenta	extractCuentaFromJSON	JSONObject json
Private	Buzon	extractBuzonFromJSON	JSONObject json
Private	String	getPhpPath	
Observaciones			
-			

7.4.2.3.2.2 *ClienteDAO*

Nombre	Tipo	Extiende	Implementa
ClienteDAO	Class	GenericDAO	ClienteDataService
Descripción			
Clase encargada de la persistencia del objeto Cliente tanto en base de datos como en el servidor.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	createClient	ISPConfigClient cliente
Public	ISPConfigClient	getClienteById	int it
Public	ISPConfigClient	getClienteByNombre	String nombre
Public	List<ISPConfigClient>	getClientes	
Public	int	getISPEquivalente	int idCliente
Private	ISPConfigClient	extractClienteFromJSON	JSONObject json
Private	String	getPhpPath	
Observaciones			
-			

7.4.2.3.2.3 *DominioDAO*

Nombre	Tipo	Extiende	Implementa
DominioDAO	Class		DominioDataService
Descripción			
Clase encargada de la persistencia del objeto Dominio.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	int	addDominio	Dominio dominio
public	List<Dominio>	getAllDomains	
public	List<Dominio>	getByClient	int idCliente
public	Dominio	getDominioById	int idDominio
public	List<Dominio>	getSubDomains	
public	void	toggle	Dominio dominio
private	Dominio	extractDominioFromJSON	JSONObject json
private	List<Dominio>	loadSubDominios	int idDominio
private	String	getPhpPath	
Observaciones			
-			

7.4.2.3.2.4 *ServidorDAO*

Nombre	Tipo	Extiende	Implementa
ServidorDAO	Class		ServidorDataService
Descripción			
Clase encargada de la persistencia del objeto Servidor.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	List<Servidor>	findAll	
public	Servidor	getById	int idServidor
private	Servidor	extractServidorFromJSON	JSONObject json
private	List<Dominio>	loadSubDominios	int idDominio
private	String	getPhpPath	
Observaciones			
-			

7.4.2.3.3 Negocio

7.4.2.3.3.1 *BuzonManager*

Nombre	Tipo	Extiende	Implementa
BuzonManager	Class		BuzonManagerService
Descripción			
Clase encargada de la lógica relacionada con el objeto Buzón.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	void	createBuzon	Buzon buzon
Public	void	createCuenta	Cuenta cuenta
Public	List<Buzon>	getAllBuzones	
Public	Cuenta	getCuentaById	int idCuenta
Public	Cuenta	getCuentaByName	String nombre
Public	Buzon	getBuzonById	int idBuzon
Public	void	setAutoRespuesta	Cuenta cuenta
Private	Buzon	getBuzonByNombre	String nombre
Private	void	loadRelacionesBuzon	Buzon buzon
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		BuzonDataService	buzonDataService
Private		ClienteDataService	clienteDataService
Private		ServidorDataService	servidorDataService
Observaciones			
-			

7.4.2.3.3.2 *ClienteManager*

Nombre	**Tipo**	**Extiende**	**Implementa**
ClienteManager	Class		ClienteManagerService
Descripción			
Clase encargada de la lógica relacionada con el objeto ISPClient.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	ISPConfigClient	getClienteISPById	int id
public	Cliente	getClienteById	int id
public	List<Cliente>	getClientes	
public	List<ISPConfigClient>	getClientesISP	
public	ISPConfigClient	getClienteISPByCliente	int idCliente
public	boolean	isISPClient	int idCienteISP
public	int	saveNuevoCliente	Cliente cliente
public	int	saveNuevoClienteISP	ClienteISP cliente
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteDataService	clienteDataService
private		ClienteISPDataService	clienteISPDataService
private		ServidorDataService	servidorDataService
Observaciones			
-			

7.4.2.3.3.3 *DominioManager*

Nombre	Tipo	Extiende	Implementa
DominioManager	Class		DominioManagerService
Descripción			
Clase encargada de la lógica relacionada con el objeto Dominio.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	int	addDominio	Dominio dominio
Public	List<Dominio>	getAllDomains	
Public	List<Dominio>	getByClient	int idCliente
Public	Dominio	getDominioById	int id
Public	List<Dominio>	getSubDomains	
Public	void	toggle	Dominio dominio
Private	void	loadRelacionesDominio	Dominio dominio
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		ClienteDataService	clienteDataService
Private		DominioDataService	dominioDataService
Private		ServidorDataService	servidorDataService
Observaciones			
-			

7.4.2.3.3.4 *ServidorManager*

Nombre	Tipo	Extiende	Implementa
ServidorManager	Class		ServidorManagerService
Descripción			
Clase encargada de la lógica relacionada con el objeto Servidor.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	List<Servidor>	findAll	
Public	Servidor	getById	int idServidor
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		ServidorDataService	servidorDataService
Observaciones			
-			

7.4.2.3.4 Presentación

7.4.2.3.4.1 *ActualizarAutorespuestAction*

Nombre	Tipo	Extiende	Implementa
ActualizarAutorespuestaAction	Class	ActionSupport	SessionAware ModelDriven<AutoRespuesta>
Descripción			
Action encargado de iniciar el proceso de configuración de la autorespuesta de una cuenta.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		BuzonManagerService	buzonManagerService
private		Map<String, Object>	session
private		Logger	log
private		AutoRespuesta	respuesta
private		int	idBuzon
Observaciones			
-			

7.4.2.3.4.2 *AddNuevaCuentaAction*

Nombre	Tipo	Extiende	Implementa
AddNuevaCuentaAction	Class	ActionSupport	SessionAware ModelDriven<Cuenta>
Descripción			
Action encargado de iniciar el proceso de adición de una cuenta.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		BuzonManagerService	buzonManagerService
private		Map<String, Object>	session
private		Logger	log
private		Cuenta	cuenta
Observaciones			
-			

7.4.2.3.4.3 *AddNuevoBuzonAction*

Nombre	Tipo	Extiende	Implementa
AddNuevoBuzonAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de iniciar el proceso de adición de un buzón.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		BuzonManagerService	buzonManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		int	idCliente
Private		int	servidor
Private		String	nombre
Observaciones			
-			

7.4.2.3.4.4 *CargarBuzonesSistemaAction*

Nombre	Tipo	Extiende	Implementa
CargarBuzonesSistemaAction	Class	ActionSupport	SessionAware
Descripción			
Action encaargado de preparar todos los datos de buzones para ser usados por la interfaz de usuario.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		BuzonManagerService	buzonManagerService
Private		ClienteManagerService	clienteManagerService
Private		ServidorManagerService	servidorManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		Map<Integer, String>	clientes
Private		List<Buzon>	buzones
Private		Map<Integer, String>	servidores
Observaciones			
-			

7.4.2.3.4.5 *CargarFormularioNuevaCuentaAction*

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevaCuentaAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar la interfaz de la JSP añadir cuenta nueva.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		Map<String, Object>	session
private		Logger	log
private		Buzon	buzon
Observaciones			
-			

7.4.2.3.4.6 *CargarFormularioNuevoBuzonAction*

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevoBuzonAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar la interfaz de la JSP añadir buzón nuevo.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		DominioManagerService	dominioManagerService
private		ServidorManagerService	servidorManagerService
private		Map<String, Object>	session
private		Logger	log
private		Map<Integer, String>	clientes
private		Map<Integer, String>	dominios
private		Map<Integer, String>	servidores
Observaciones			
-			

7.4.2.3.4.7 *ExplorarBuzonAction*

Nombre	Tipo	Extiende	Implementa
ExplorarBuzonAction	Class	ActionSupport	SessionAware ParametersAware
Descripción			
Action encargado de cargar los datos de un Buzón.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		BuzonManagerService	buzonManagerService
Private		Map<String, Object>	session
Private		Map<String, String>	parameters
Private		Logger	log
Private		Buzon	buzon
Observaciones			
-			

7.4.2.3.4.8 *ObtenerDatosBuzonAction*

Nombre	Tipo	Extiende	Implementa
ObtenerDatosBuzonAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de cargar los datos de un buzón para ser utilizados por javascript.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		Map<String, Object>	session
Private		Logger	log
Private		int	idCuenta
Private		Cuenta	cuenta
Observaciones			
-			

7.4.2.3.4.9 *AddNuevoClienteAction*

Nombre	**Tipo**	**Extiende**	**Implementa**
AddNuevoClienteAction	Class	ActionSupport	SessionAware ModelDriven<ISPConfigClient>
Descripción			
Action encargado de comenzar el proceso de adición de un cliente.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
private	Cliente	composeClienteData	
private	Usuario	composeUserData	Cliente cliente
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Logger	log
private		ISPConfigClient	client
private		int	usuario
private		int	cliente
private		String	login
private		String	password
private		String	nif
private		String	calle
private		String	ciudad
private		String	provincia
private		String	telefono
private		String	fax
Observaciones			
-			

7.4.2.3.4.10 *CargarClientesSistemaAction*

Nombre	Tipo	Extiende	Implementa
CargarClientesSistemaAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de cargar los datos de los clientes ISP.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		ClienteManagerService	clienteManagerService
Private		Map<String, Object>	session
Private		Logger	log
Private		List< ISPConfigClient >	clientes
Observaciones			
-			

7.4.2.3.4.11 *CargarFormularioNuevoClienteAction*

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevoClienteAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de preparar los datos necesarios por la interfaz de añadir cliente.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Logger	log
private		Map<Integer, String>	clientes
private		Map<Integer, String>	usuarios
Observaciones			
-			

ExplorarClienteAction

Nombre	Tipo	Extiende	Implementa
ExplorarClienteAction	Class	ActionSupport	SessionAware ParametersAware
Descripción			
Action encargado de preparar los datos de un cliente para ser mostrados.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Map<String, String>	parameters
private		Logger	log
private		ISPConfigClient	cliente
Observaciones			
-			

7.4.2.3.4.12 *ObtenerDatosClienteAction*

Nombre	Tipo	Extiende	Implementa
ObtenerDatosClienteAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de obtener los datos de un cliente para ser utilizados por el javascript.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		UsuarioManagerService	usuarioManagerService
private		Map<String, Object>	session
private		Logger	log
private		int	idUsuario
private		Cliente	cliente
private		Usuario	usuario
Observaciones			
-			

7.4.2.3.4.13 *VerMiCuentaCorreoAction*

Nombre	Tipo	Extiende	Implementa
VerMiCuentaCorreoAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de devolver los datos de la cuenta asociados al usuario en sesión.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		BuzonManagerService	buzonManagerService
private		Map<String, Object>	session
private		Logger	log
private		Cuenta	cuenta
Observaciones			
-			

7.4.2.3.4.14 *VerMisDominiosAction*

Nombre	Tipo	Extiende	Implementa
VerMisDominiosAction	Class	ActionSupport	SessionAware
Descripción			
Action encargado de devolver los dominios asociados al usuario en sesión.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		ClienteManagerService	clienteManagerService
private		Map<String, Object>	session
private		Logger	log
private		ISPConfigClient	cliente
Observaciones			
-			

7.4.2.3.4.15 *AddNuevoDominioAction*

Nombre	Tipo	Extiende	Implementa
AddNuevoDominioAction	Class	ActionSupport	SessionAware
Action			
Action encargado de comenzar la adición de un nuevo dominio.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		DominioManagerService	dominioManagerService
private		Map<String, Object>	session
private		Logger	log
private		int	servidor
private		int	cliente
private		int	dominio
private		String	direccionIp
private		String	nombre
Observaciones			
-			

7.4.2.3.4.16 *CargarDominiosSistemaAction*

Nombre	Tipo	Extiende	Implementa
CargarDominiosSistemaAction	Class	ActionSupport	SessionAware
Action			
Action encargado de cargar todos los dominios y subdominios del servidor.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
public	void	validate	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		DominioManagerService	dominioManagerService
private		Map<String, Object>	session
private		Logger	log
private		List<Dominio>	dominios
private		List<Dominio>	subdominios
Observaciones			
-			

7.4.2.3.4.17 *CargarFormularioNuevoDominioAction*

Nombre	Tipo	Extiende	Implementa
CargarFormularioNuevo DominioAction	Class	ActionSupport	SessionAware
Action			
Action encargado de preparar los datos necesarios para el formulario añadir dominio.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	Execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		DominioManagerService	dominioManagerService
private		ClienteManagerService	clienteManagerService
private		ServidorManagerService	servidorManagerService
private		Map<String, Object>	session
private		Logger	log
private		Map<Integer, String>	clientes
private		Map<Integer, String>	dominios
private		Map<Integer, String>	servidores
Observaciones			
-			

7.4.2.3.4.18 *ExplorarDominioAction*

Nombre	Tipo	Extiende	Implementa
ExplorarDominioAction	Class	ActionSupport	SessionAware
Action			
Action encargado de cargar los datos de un Dominio del sistema para ser visualizados en la JSP.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	Execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		DominioManagerService	dominioManagerService
private		Map<String, Object>	session
private		Map<String, String>	parameters
private		Logger	log
private		Dominio	dominio
Observaciones			
-			

7.4.2.3.4.19 *ToogleDominioAction*

Nombre	**Tipo**	**Extiende**	**Implementa**
ToggleDominioAction	Class	ActionSupport	SessionAware ParametersAware
Action			
Action encargado de iniciar la acción de activar o desactivar un dominio.			
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
public	String	execute	
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
private		DominioManagerService	dominioManagerService
private		Map<String, Object>	session
private		Map<String, String>	parameters
private		Logger	log
private		int	idDominio
Observaciones			
-			

Capítulo 8. Desarrollo de las Pruebas

8.1 Pruebas Unitarias

Han sido realizadas una serie de pruebas unitarias sobre ciertas clases para comprobar su correcto funcionamiento. Debido a que el proyecto es en esencia un panel de control se han priorizado las operaciones relacionadas con persistencia, ya sea en la base de datos de la intranet o en el servidor ISPConfig.

8.1.1 Insertar usuario

- **Entrada:** Se invoca el método pasándole como parámetros un objeto Usuario con todos sus datos especificados.
- **Salida:** El usuario aparece en base de datos con los datos tal y como deben.
- **Resultado:** Correcto.

8.1.2 Obtener usuario por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del usuario a recuperar.
- **Salida:** Se recupera el objeto Usuario con los datos reflejados en la base de datos.
- **Resultado:** Correcto.

8.1.3 Obtener todos los usuarios

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de usuarios con todos los datos reflejados en la base de datos, los marcados como eliminados no aparecerán.
- **Resultado:** Correcto.

8.1.4 Obtener usuario por criterio

- **Entrada:** Se invoca uno de los distintos métodos para devolver el usuario por criterio (login, perfil, grupo).
- **Salida:** Se devuelve el objeto u objetos de la base de datos que coincidan con el parámetro especificado.
- **Resultado:** Correcto.

8.1.5 Modificar usuario

- **Entrada:** Se invoca el método pasándole como parámetro un objeto Usuario con todos sus datos y con el identificador al cual hace referencia en base de datos.
- **Salida:** Los nuevos datos indicados para el usuario aparecen en la base de datos, sobreescribiendose los antiguos.
- **Resultado:** Se módifica la contraseña aún cuando no está especificada.
- **Solución:** Se ha introducido un condicional para el caso de la contraseña, pues al ser cifrada quizás no deba sobreescribirse.

8.1.6 Eliminar usuario

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que representa el identificador del usuario a eliminar.
- **Salida:** El usuario se marca como eliminado en la base de datos.
- **Resultado:** Correcto.

8.1.7 Obtener cliente por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del cliente a recuperar.
- **Salida:** Se recupera el objeto Cliente con los datos reflejados en la base de datos.
- **Resultado:** Correcto.

8.1.8 Obtener todos los clientes

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de clientes con todos los datos reflejados en la base de datos, los marcados como eliminados no aparecerán.
- **Resultado:** Correcto.

8.1.9 Insertar grupo

- **Entrada:** Se invoca el método pasándole como parámetros un objeto grupo con todos sus datos especificados.
- **Salida:** El grupo aparece en base de datos con los datos tal y como deben.
- **Resultado:** Correcto.

8.1.10 Obtener grupo por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del grupo a recuperar.
- **Salida:** Se recupera el objeto Grupo con los datos reflejados en la base de datos.

- **Resultado:** Correcto.

8.1.11 Obtener todos los grupos

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de grupos con todos los datos reflejados en la base de datos, los marcados como eliminados no aparecerán.
- **Resultado:** Correcto.

8.1.12 Obtener grupo por descripción

- **Entrada:** Se invoca el método para devolver grupos por descripción pasándole como parámetro esta última o una lista de ellas.
- **Salida:** Se devuelve el objeto u objetos de la base de datos que coincidan con el parámetro especificado.
- **Resultado:** Correcto.

8.1.13 Modificar grupo

- **Entrada:** Se invoca el método pasándole como parámetro un objeto Grupo con todos sus datos y con el identificador al cual hace referencia en base de datos.
- **Salida:** Los nuevos datos indicados para el grupo aparecen en la base de datos, sobreescribiendose los antiguos.
- **Resultado:** Correcto.

8.1.14 Eliminar grupo

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que representa el identificador del grupo a eliminar.
- **Salida:** El grupo se marca como eliminado en la base de datos.
- **Resultado:** Correcto.

8.1.15 Insertar perfil

- **Entrada:** Se invoca el método pasándole como parámetros un objeto perfil con todos sus datos especificados.
- **Salida:** El perfil aparece en base de datos con los datos tal y como deben.
- **Resultado:** Correcto.

8.1.16 Obtener perfil por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del perfil a recuperar.

- **Salida:** Se recupera el objeto Perfil con los datos reflejados en la base de datos.
- **Resultado:** Correcto.

8.1.17 Obtener todos los perfiles

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de perfiles con todos los datos reflejados en la base de datos, los marcados como eliminados no aparecerán.
- **Resultado:** Correcto.

8.1.18 Obtener perfil por descripción

- **Entrada:** Se invoca el método para devolver perfiles por descripción pasándole como parámetro esta última o una lista de ellas.
- **Salida:** Se devuelve el objeto u objetos de la base de datos que coincidan con el parámetro especificado.
- **Resultado:** Correcto.

8.1.19 Modificar perfil

- **Entrada:** Se invoca el método pasándole como parámetro un objeto Perfil con todos sus datos y con el identificador al cual hace referencia en base de datos.
- **Salida:** Los nuevos datos indicados para el perfil aparecen en la base de datos, sobreescribiendose los antiguos.
- **Resultado:** Correcto.

8.1.20 Eliminar perfil

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que representa el identificador del perfil a eliminar.
- **Salida:** El perfil se marca como eliminado en la base de datos.
- **Resultado:** Correcto.

8.1.21 Obtener todos los módulos

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de módulos con todos los datos reflejados en la base de datos, los marcados como eliminados no aparecerán.
- **Resultado:** Correcto.

8.1.22 Obtener módulo por nombre

- **Entrada:** Se invoca el método para devolver módulos por nombre pasándole como parámetro este último o una lista de ellos.
- **Salida:** Se devuelve el objeto u objetos de la base de datos que coincidan con el parámetro especificado.
- **Resultado:** Correcto.

8.1.23 Insertar dominio

- **Entrada:** Se invoca el método pasándole como parámetros un objeto dominio con todos sus datos especificados.
- **Salida:** El dominio aparece en el servidor ISP Config con sus datos tal y como deben.
- **Resultado:** Correcto.

8.1.24 Obtener dominio por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del dominio a recuperar.
- **Salida:** Se recupera el objeto Dominio con los datos reflejados en el servidor ISPConfig.
- **Resultado:** Correcto.

8.1.25 Obtener todos los dominios

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de dominios con todos los datos reflejados el servidor ISPConfig.
- **Resultado:** No se ofrece esté método en la capa de servicios web.
- **Solución:** Se llama varias veces al método que permite obtenerlos por identificador.

8.1.26 Obtener dominios por cliente

- **Entrada:** Se invoca el método para devolver dominios por cliente pasándole por parámetro el identificador del mismo en ISPConfig.
- **Salida:** Se devuelve el objeto u objetos del servidor que coincidan con el parámetro especificado.
- **Resultado:** Correcto.

8.1.27 Activar/Desactivar Dominio

- **Entrada:** Se invoca el método pasándole como parámetro un objeto dominio con todos sus datos y con el identificador al cual hace referencia en el servidor ISPConfig.
- **Salida:** El estado del dominio cambia en el servidor ISPConfig.

- **Resultado:** Correcto.

8.1.28 Eliminar dominio

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que representa el identificador del dominio a eliminar.
- **Salida:** El dominio se remueve del servidor ISPConfig.
- **Resultado:** Correcto.

8.1.29 Insertar buzón

- **Entrada:** Se invoca el método pasándole como parámetros un objeto buzón con todos sus datos especificados.
- **Salida:** El buzón aparece en el servidor ISP Config con sus datos tal y como deben.
- **Resultado:** Correcto.

8.1.30 Obtener buzón por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del buzón a recuperar.
- **Salida:** Se recupera el objeto buzón con los datos reflejados en el servidor ISPConfig.
- **Resultado:** Correcto.

8.1.31 Obtener todos los buzones

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de buzones con todos los datos reflejados el servidor ISPConfig.
- **Resultado:** No existe tal método en la capa de servicios web.
- **Solución:** Se llama varias veces al método que permite obtener los buzones por identificador.

8.1.32 Insertar cuenta

- **Entrada:** Se invoca el método pasándole como parámetros un objeto cuenta con todos sus datos especificados.
- **Salida:** La cuenta aparece en el servidor ISP Config con sus datos tal y como deben.
- **Resultado:** Correcto.

8.1.33 Obtener cuenta por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador de la cuenta a recuperar.

- **Salida:** Se recupera el objeto cuenta con los datos reflejados en el servidor ISPConfig.
- **Resultado:** Correcto.

8.1.34 Obtener cuenta por nombre

- **Entrada:** Se invoca el método para devolver cuentas por nombre pasándole por parámetro el nombre del mismo en ISPConfig.
- **Salida:** Se devuelve el objeto del servidor que coincida con el parámetro especificado.
- **Resultado:** No se dispone de ese tipo de métodos dentro de la capa de servicios web.
- **Solución:** Para conseguir esto se utiliza el dominio del nombre de cuenta para pidiendo todos los buzones del servidor comprobar cual coincide y dentro de las cuentas asociadas al mismo comprueba que nombre coincide.

8.1.35 Aplicar una autorespuesta

- **Entrada:** Se invoca el método pasándole como parámetro un objeto cuenta con todos sus datos y con el identificador al cual hace referencia en el servidor ISPConfig.
- **Salida:** El estado de la autorespuesta asociada a la cuenta cambia en el servidor ISPConfig.
- **Resultado:** Las fechas se reinician y se ponen a 0.
- **Solución:** Esto es un bug conocido del servidor ISPConfig, esperamos que en alguna versión nueva lo solucionen.

8.1.36 Insertar clienteISP

- **Entrada:** Se invoca el método pasándole como parámetros un objeto clienteISP con todos sus datos especificados.
- **Salida:** Aparece un usuario, vinculado a un cliente en la base de datos, el cual a su vez está vinculado con el id del clienteISP que aparecerá nuevo en el servidor.
- **Resultado:** Correcto.

8.1.37 Obtener clienteISP por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del cliente a recuperar.
- **Salida:** Se recupera el objeto cliente con los datos reflejados en el servidor ISPConfig.
- **Resultado:** Correcto.

8.1.38 Obtener todos los clientes

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de clientes con todos los datos reflejados el servidor ISPConfig siempre y cuando existan en base de datos vinculadas.

- **Resultado:** Correcto.

8.1.39 Eliminar cliente

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que representa el identificador del cliente a eliminar.
- **Salida:** El cliente se remueve del servidor ISPConfig y la entrada en la base de datos desaparece.
- **Resultado:** Correcto.

8.1.40 Obtener servidor por identificador

- **Entrada:** Se invoca el método pasándole como parámetro un número entero que represente el identificador del servidor a recuperar.
- **Salida:** Se recupera el objeto servidor con los datos reflejados en el servidor ISPConfig.
- **Resultado:** Correcto.

8.1.41 Obtener todos los servidores

- **Entrada:** Se invoca el método sin necesidad de especificar parámetro alguno.
- **Salida:** Se recupera una lista de servidores con todos los datos reflejados el servidor ISPConfig.
- **Resultado:** Correcto.

8.2 Pruebas de Integración y del Sistema

Una vez integrado el conjunto de clases y ficheros necesarios para ejecutar la aplicación al completo, nos disponemos a ejecutar las pruebas especificadas en el apartado 5.7 de esta documentación.

En esta ocasión, además de los resultados de la base de datos, que ya se especificó en las pruebas unitarias anteriores, también observaremos y vigilaremos los resultados tras todas las operaciones en la interfaz de usuario que es, en definitiva, el componente que debe reflejar los cambios de cara a que el usuario pueda trabajar adecuadamente.

Aunque no se especifica, todas ellas generán líneas de log en la base de datos para facilitar al administrador el rastreo de errores.

8.2.1 Intranet

Iniciar sesión	
Prueba	**Resultado Esperado**
Iniciar sesión con credenciales válidas.	El usuario se encuentra en el sistema.
	Resultado Obtenido
	Efectivamente el usuario se encuentra en el sistema y se pueden acceder a sus datos desde la sesión.
Prueba	**Resultado Esperado**
Iniciar sesión con credenciales no válidas.	El sistema no se encuentra dentro de la aplicación y se le muestra un dialogo notificándolo.
	Resultado Obtenido
	Efectivamente el usuario recibe un mensaje indicándole que sus credenciales no son válidas.

Cerrar sesión	
Prueba	**Resultado Esperado**
El usuario cierra su sesión.	El usuario se encuentra fuera del sistema, se eliminan las cookies y la sesión en la base de datos se cierra automáticamente.
	Resultado Obtenido
	El usuario se expulsa de la aplicación pero la cookie persite, esto se arreglo devolviéndolas en la request tras modificarlas.

Acceder a webs privadas	
Prueba	**Resultado Esperado**
El usuario está identificado	Se permite al usuario visualizar y navegar por la web que desea.
	Resultado Obtenido
	Efectivamente el usuario tiene acceso a todas las webs de la parte privada de la aplicación.
Prueba	**Resultado Esperado**
El usuario no está	Se intercepta al usuario y se le devuelve a la web de inicio de

identificado	sesión
	Resultado Obtenido
	Efectivamente se situa al usuario fuera de la aplicación.
Prueba	**Resultado Esperado**
El usuario no se ha identificado pero existe una cookie activa	Se permite al usuario visualizar y navegar por la web que desea y se le coloca en sesión.
	Resultado Obtenido
	Efectivamente se extrae el usuario de la cookie y se coloca en sesión.

Acceder a webs que necesitan permisos	
Prueba	**Resultado Esperado**
El usuario tiene permisos para acceder a la web	Se permite al usuario visualizar y navegar por la web que desea.
	Resultado Obtenido
	Efectivamente el usuario tiene acceso a todas las webs que sus permisos le permiten.
Prueba	**Resultado Esperado**
El usuario no tiene permisos para acceder a la web	Se intercepta al usuario y se le devuelve a la web de acceso restringido.
	Resultado Obtenido
	Efectivamente se notifica al usuario y no se le permite navegar.
Prueba	**Resultado Esperado**
El usuario no tiene permisos para acceder a la web pero uno de sus perfiles asociados sí.	Se permite al usuario visualizar y navegar por la web que desea.
	Resultado Obtenido
	Efectivamente el usuario tiene acceso a todas las webs que los permisos de sus perfiles le permiten.

8.2.2 Gestión de usuarios

Añadir usuario	
Prueba	**Resultado Esperado**
Añadir un usuario no existente	El sistema posee un usuario más
	Resultado Obtenido
	Efectivamente el sistema poseé un usuario más.
Prueba	**Resultado Esperado**
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo, se le guardan los datos que rellenó.
	Resultado Obtenido
	Efectivamente se notifica al usuario que ya existe.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Consultar usuario	
Prueba	**Resultado Esperado**
Consultar un usuario existente	Se carga el objeto usuario y se muestra en la interfaz.
	Resultado Obtenido
	Efectivamente los datos del usuario aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar un usuario no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Consultar un usuario marcado como eliminado.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Actualizar usuario	
Prueba	**Resultado Esperado**
Actualizar un usuario existente	Los datos del usuario se han modificado en base de datos.
	Resultado Obtenido
	Efectivamente se han modificado los datos del usuario excepto la contraseña.
Prueba	**Resultado Esperado**
Actualizar un usuario no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Eliminar usuario	
Prueba	**Resultado Esperado**
Eliminar un usuario existente	El usuario se marca como eliminado en la base de datos.
	Resultado Obtenido
	El usuario aparece como eliminado en la base de datos.
Prueba	**Resultado Esperado**
Eliminar un usuario no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Añadir perfil	
Prueba	**Resultado Esperado**
Añadir un perfil no existente	El sistema posee un perfil más
	Resultado Obtenido
	Efectivamente el sistema poseé un perfil más.
Prueba	**Resultado Esperado**
Añadir un perfil que ya existe	El sistema no posee un perfil más y se muestra un dialogo notificándolo, se le guardan los datos que rellenó.
	Resultado Obtenido
	Efectivamente se notifica al usuario que ya existe.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Consultar perfil	
Prueba	**Resultado Esperado**
Consultar un perfil existente	Se carga el objeto perfil y se muestra en la interfaz.
	Resultado Obtenido
	Efectivamente los datos del perfil aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar un perfil no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Consultar un perfil marcado como eliminado.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Actualizar perfil	
Prueba	**Resultado Esperado**
Actualizar un perfil existente	Los datos del perfil se han modificado en base de datos.
	Resultado Obtenido
	Efectivamente se han modificado los datos del perfil.
Prueba	**Resultado Esperado**
Actualizar un perfil no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Eliminar perfil	
Prueba	**Resultado Esperado**
Eliminar un perfil existente	El perfil se marca como eliminado en la base de datos.
	Resultado Obtenido
	El perfil aparece como eliminado en la base de datos.
Prueba	**Resultado Esperado**
Eliminar un perfil no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Añadir grupo	
Prueba	**Resultado Esperado**
Añadir un grupo no existente	El sistema posee un grupo más
	Resultado Obtenido
	Efectivamente el sistema poseé un grupo más.
Prueba	**Resultado Esperado**
Añadir un grupo que ya existe	El sistema no posee un grupo más y se muestra un dialogo notificándolo, se le guardan los datos que rellenó.
	Resultado Obtenido
	Efectivamente se notifica al usuario que ya existe.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Consultar grupo	
Prueba	**Resultado Esperado**
Consultar un grupo existente	Se carga el objeto grupo y se muestra en la interfaz.
	Resultado Obtenido
	Efectivamente los datos del grupo aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar un grupo no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Consultar un grupo marcado como eliminado.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Actualizar grupo	
Prueba	**Resultado Esperado**
Actualizar un grupo existente	Los datos del grupo se han modificado en base de datos.

	Resultado Obtenido
	Efectivamente se han modificado los datos del grupo.
Prueba	**Resultado Esperado**
Actualizar un grupo no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Eliminar grupo	
Prueba	**Resultado Esperado**
Eliminar un grupo existente	El grupo se marca como eliminado en la base de datos.
	Resultado Obtenido
	El grupo aparece como eliminado en la base de datos.
Prueba	**Resultado Esperado**
Eliminar un grupo no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

8.2.3 Gestión de dominios

Añadir Dominio	
Prueba	**Resultado Esperado**
Añadir un dominio no existente	El servidor ISP contiene un nuevo dominio.
	Resultado Obtenido
	Efectivamente el sistema poseé un dominio más.
Prueba	**Resultado Esperado**
Añadir un dominio que ya existe	El sistema no posee un dominio más y se muestra un dialogo notificándolo, se le guardan los datos que rellenó.
	Resultado Obtenido
	Efectivamente se notifica al usuario que ya existe.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Consultar Dominio	
Prueba	**Resultado Esperado**
Consultar un dominio existente	Se carga el objeto dominio y se muestra en la interfaz.

	Resultado Obtenido
	Efectivamente los datos del perfil aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar un dominio no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Eliminar dominio	
Prueba	**Resultado Esperado**
Eliminar un dominio existente	El dominio se elimina del servidor ISPConfig.
	Resultado Obtenido
	Efectivamente el perfil desaparece del servidor.
Prueba	**Resultado Esperado**
Eliminar un dominio no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Añadir Cliente	
Prueba	**Resultado Esperado**
Añadir un cliente sin vinculación ninguna	Se crea el cliente en el servidor ISPConfig y aparece en la tabla de relación de la base de datos, además se crea un objeto usuario y otro cliente en la base de datos.
	Resultado Obtenido
	Efectivamente aparecen todos los cambios.
Prueba	**Resultado Esperado**
Añadir un cliente para un cliente ya existente.	Se crea el cliente en el servidor ISPConfig y aparece en la tabla de relación de la base de datos, además se crea un objeto usuario.
	Resultado Obtenido
	Efectivamente aparecen todos los cambios.
Prueba	**Resultado Esperado**
Añadir un cliente para un usuario ya existente.	Se crea el cliente en el servidor ISPConfig y aparece en la tabla de relación de la base de datos, además se crea un objeto cliente.
	Resultado Obtenido
	Efectivamente aparecen todos los cambios.
Prueba	**Resultado Esperado**
Añadir un cliente para un usuario y un cliente ya existentes.	Se crea el cliente en el servidor ISPConfig y aparece en la tabla de relación de la base de datos.
	Resultado Obtenido
	Efectivamente aparecen todos los cambios.
Prueba	**Resultado Esperado**

Cancelar la operación	No se produce cambio alguno en el sistema.
	Resultado Obtenido
	Efectivamente se cancelan los cambios.

Consultar cliente	
Prueba	**Resultado Esperado**
Consultar un cliente existente	Se carga el objeto cliente y se muestra en la interfaz.
	Resultado Obtenido
	Efectivamente los datos del cliente aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar un cliente no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Eliminar cliente	
Prueba	**Resultado Esperado**
Eliminar un cliente existente	Desaparece el cliente del servidor y la entrada en la tabla de relación de la base de datos.
	Resultado Obtenido
	Se han producido todos los cambios.
Prueba	**Resultado Esperado**
Eliminar un cliente no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Añadir Buzón	
Prueba	**Resultado Esperado**
Añadir un buzón no existente	El servidor ISP contiene un nuevo buzón.
	Resultado Obtenido
	Efectivamente el sistema poseé un buzón más.
Prueba	**Resultado Esperado**
Añadir un buzón que ya existe	El sistema no posee un buzón más y se muestra un dialogo notificándolo, se le guardan los datos que rellenó.
	Resultado Obtenido
	Efectivamente se notifica al usuario que ya existe.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Consultar Buzón	
Prueba	**Resultado Esperado**

Consultar un buzón existente	Se carga el objeto buzón y se muestra en la interfaz.
	Resultado Obtenido
	Efectivamente los datos del perfil aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar un buzón no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Eliminar buzón	
Prueba	**Resultado Esperado**
Eliminar un buzón existente	El buzón se elimina del servidor ISPConfig.
	Resultado Obtenido
	Efectivamente el perfil desaparece del servidor.
Prueba	**Resultado Esperado**
Eliminar un buzón no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Añadir Cuenta	
Prueba	**Resultado Esperado**
Añadir una cuenta no existente	El servidor ISP contiene una nueva cuenta.
	Resultado Obtenido
	Efectivamente el sistema poseé un cuenta más.
Prueba	**Resultado Esperado**
Añadir una cuenta que ya existe	El sistema no posee un cuenta más y se muestra un dialogo notificándolo, se le guardan los datos que rellenó.
	Resultado Obtenido
	Efectivamente se notifica al usuario que ya existe.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

Consultar Cuenta	
Prueba	**Resultado Esperado**
Consultar una cuenta existente	Se carga el objeto cuenta y se muestra en la interfaz.
	Resultado Obtenido
	Efectivamente los datos del perfil aparecen en la interfaz.
Prueba	**Resultado Esperado**
Consultar una cuenta no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.

	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.

Eliminar cuenta	
Prueba	**Resultado Esperado**
Eliminar una cuenta existente	El cuenta se elimina del servidor ISPConfig.
	Resultado Obtenido
	Efectivamente el perfil desaparece del servidor.
Prueba	**Resultado Esperado**
Eliminar una cuenta no existente.	El sistema lo detecta y le indica al usuario que no se puede llevar a cabo la operación.
	Resultado Obtenido
	Efectivamente se devuelve un mensaje de error al usuario.
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.
	Resultado Obtenido

8.3 Pruebas de rendimiento

La aplicación desarrollada se trata de un intranet para la empresa Ecocomputer S.L y por lo tanto no se tratará de un entorno de alta explotación. Esto ocasiona que no se hayan previsto grandes problemas derivados del rendimiento.

En cuando al rendimiento del código, el desarrollo se ha realizado pensando en mantener un alto nivel de calidad y limpieza a la vez que se intentaba alcanzar la menor complejidad posible. Las conexiones con la base de datos, el aspecto más susceptible de presentar cadencias en las operaciones, forman parte de la Intranet, y por lo tanto las conexiones se realizan a través de un pool de conexiones, por lo que el rendimiento es óptimo en entornos con una alta concurrencia.

Uno de los aspectos que merece mención es el tamaño de las tablas de la base de datos relacionados con el registro del sistema, las sesiones de los usuarios y el log de sus operaciones, puesto que todo lo que hagan los usuarios queda registrado. Dichas tablas pueden llegar a contener una gran cantidad de información conforme la aplicación siga funcionando a lo largo de los meses. Es responsabilidad del cliente, administrador de dicha base de datos, el control del mencionado aspecto, este ha sido requerido por el cliente y por lo tanto no se ha cuestionado una forma mejor de llevarlo a cabo.

Finalmente se nota una gran cadencia de rendimiento cuando se trata de realizar conexiones con el servidor ISPConfig, esto es debido a la falta de unos servicios web que permitan obtener en muchos de los casos los datos necesitados por la aplicación, conllevando esto que se hayan de hace multiples peticiones con el tiempo de carga que esto conlleva.

8.4 Pruebas de Accesibilidad

Aunque no es requisito del cliente y al tratarse de una Intranet los usuarios no van formar parte de ningún perfil que presente conflictos de cara a la accesibilidad, sí que se ha intentado en la medida de lo posible adaptarse al nivel mínimo de accesibilidad requerido por las WCAG.

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 (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			**X**
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 técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.			**X**
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			**X**

cuando ello facilite la comprensión de la página.			
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**	

Como vemos aunque se ha intentado cumplir en la medida de lo posible las normas hay algunas que no se han podido ya sea por el diseño intrínseco de la web que viene dado por el cliente o por la manera en la que el cliente espera que se comporte.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

9.1.1 Servidor de base de datos

El proyecto se comunica con un servidor de base de datos SQLServer, en su versión 2005 para las pruebas y en la versión 2008 para producción. Debido a que hemos utilizado la primera versión mencionada será está la que se enseñe como instalar.

En primer lugar será necesario instalar el servidor de base de datos Microsoft SQL Server 2008. Para ello se necesita disponer de un equipo con Windows Server 2008 o superior y cumplir con los siguientes requisitos mínimos:

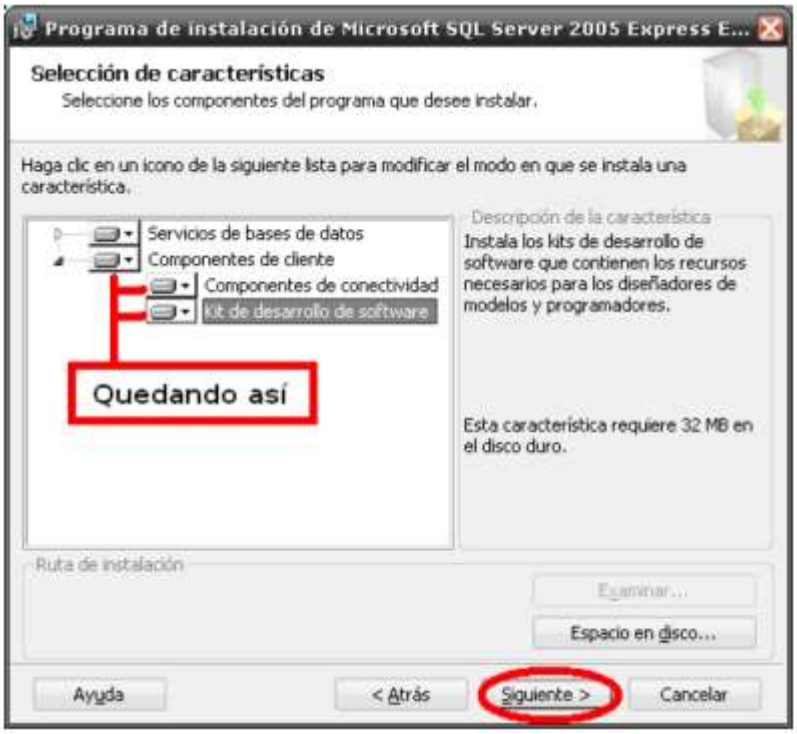
- Procesador Pentium III o superior a 600 MHz (recomendado 1 GHz o más);
- 512 MB de memoria RAM (recomendado 1 GB o más).
- Al menos 2 GB de disco duro.

Antes de comenzar con el proceso de instalación se recomienda tener el sistema operativo totalmente actualizado, mediante el uso de Windows Update. Seguidamente ejecutamos el instalador que comenzará el proceso.

Tras aceptar los habituales términos y condiciones de la licencia y pasar por las primeras pantallas de bienvenida, veremos cómo se ejecuta una comprobación de los requisitos hardware y software del sistema. Si todo está en orden podremos pasar al siguiente paso en el que se nos pedirán nuestros datos:

Manual instalación 9.1: SQLServer-Información de registro

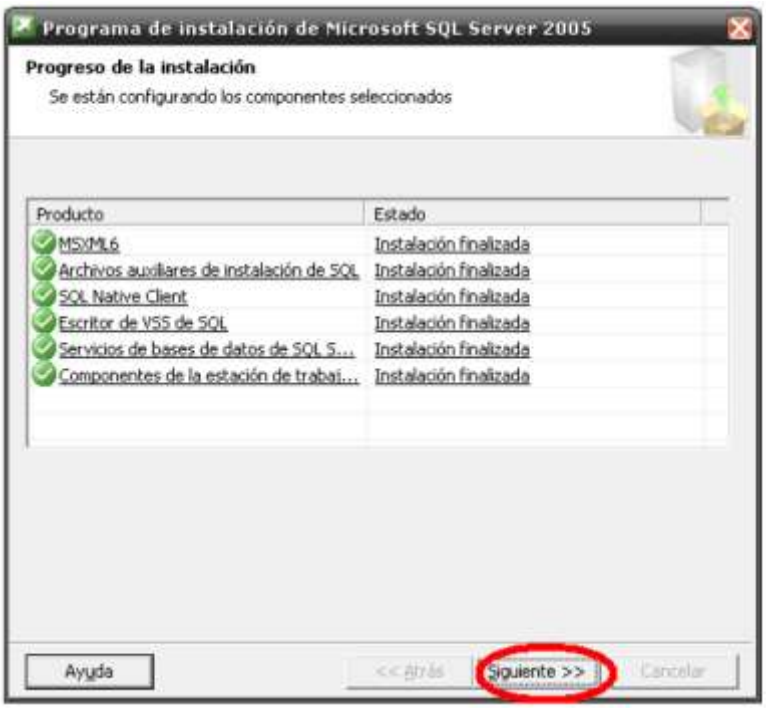
Seguidamente deberán seleccionarse qué componentes se quieren instalar. Para evitar posibles problemas se aconseja marcar todo:



Manual instalación 9.2: SQLServer-Componentes

En el siguiente paso deberemos indicar el nombre de la instancia, que podrá ser el que el usuario desee (en nuestro caso ha sido nombrado Intranet). Al dar a siguiente deberemos elegir modo de autenticación mixto para que la identificación del usuario no se limite únicamente al entorno Windows y sus usuarios. También deberemos escribir la contraseña para el usuario por defecto “sa”, a nuestra elección (en este caso la contraseña es Eco.Intranet).

Tras comprobar que estamos satisfechos con los parámetros de configuración establecidos, el servidor de base de datos se instalará y avisará cuando el proceso concluya:



Manual instalación 9.3: SQLServer-Finalización

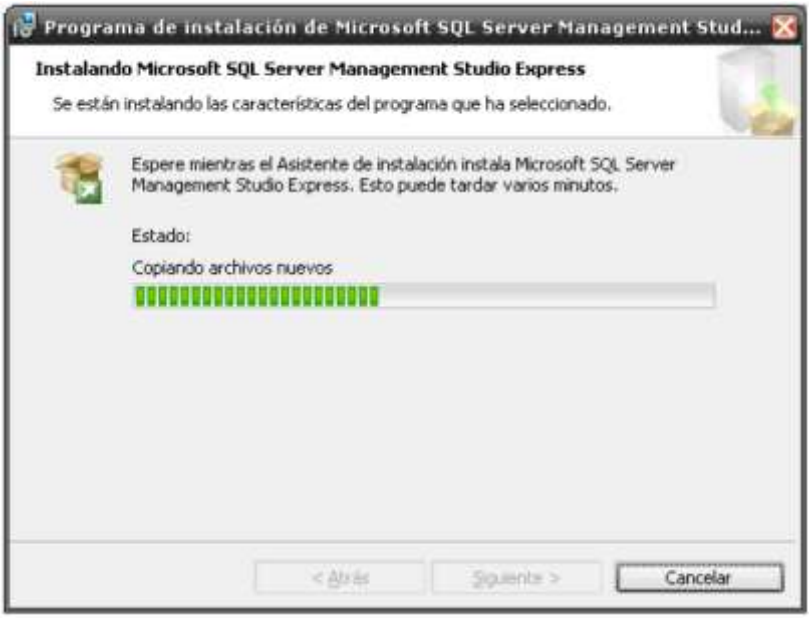
Para finalizar quizá sea necesario indicar que el cortafuegos del equipo debe estar configurado correctamente para recibir peticiones en el puerto en el que el servidor de base de datos esté escuchando (por defecto, 1433).

9.1.2 Sistema de gestión de base de datos

Una vez con el servidor de base de datos instalado necesitamos el programa que lo gestione de una manera sencilla para el usuario: el SGBD. Por comodidad y gratuidad será Microsoft SQL Server Management Studio Express.

Tras ejecutar el instalador y aceptar las ya comunes primeras pantallas de aceptación de condiciones y bienvenida, se nos pedirá nuevamente los mismos datos de usuario que ya pusimos en la anterior instalación.

En el siguiente paso debemos elegir los componentes a instalar, que por defecto ya está seleccionado el único disponible y necesario. Seguimos aceptando los pasos del instalador y comenzará el proceso:



Manual instalación 9.4: SQLServer management studio

Una vez instalado tanto el servidor de base de datos como su sistema de gestión se recomienda de nuevo actualizar todo mediante Windows Update, puesto que es probable que exista algún parche de seguridad o rendimiento para ambos paquetes.

9.1.3 Base de datos

Ya con el servidor de base de datos y el SGBD en el equipo procedemos a instalar la base de datos, sus tablas y datos mínimos para poder trabajar en la aplicación. Para ello accedemos a Microsoft SQL Server Management Studio Express y nos pedirá los datos de conexión a la instancia del servidor, que deberemos rellenar como sigue (o valores propios si se decidió utilizar otros parámetros):

Manual instalación 9.5: Conexión base de datos

Una vez conectados, en el explorador de objetos deberá crearse una nueva base de datos dejando los campos por defecto excepto, claro está, el nombre. En nuestro caso hemos usado "Ecocomputer". Una vez creada, damos clic derecho sobre ella y seleccionamos la opción "Nueva consulta". En el editor de texto que aparecerá pegamos y ejecutamos el siguiente código, que genera las tablas sus columnas de la base de datos (el script puede encontrarse también en el directorio de software a instalar bajo el nombre "script.sql").

```
IF NOT EXISTS (SELECT * FROM sys.database_principals WHERE name = N'pfmecocomputer')
CREATE USER [pfmecocomputer] FOR LOGIN [pfmecocomputer] WITH DEFAULT_SCHEMA=[dbo]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Modulos]')
AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Modulos](
	[IDModulo] [int] IDENTITY(1,1) NOT NULL,
	[Nombre] [varchar](50) NULL,
 CONSTRAINT [PK_Modulos] PRIMARY KEY CLUSTERED
(
	[IDModulo] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[TiposRelaciones]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[TiposRelaciones](
	[IDTipoRelacion] [int] NOT NULL,
	[Descripcion] [varchar](150) NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_TiposRelaciones] PRIMARY KEY CLUSTERED
(
	[IDTipoRelacion] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[TiposDieta]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[TiposDieta](
	[IDTipoDieta] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](200) NULL,
	[Unidades] [varchar](20) NULL,
 CONSTRAINT [PK_TiposDieta] PRIMARY KEY CLUSTERED
(
	[IDTipoDieta] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ComplementosNoSalariales]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[ComplementosNoSalariales](
	[IDComplementoNoSalarial] [int] IDENTITY(1,1) NOT NULL,
	[IDTipoDieta] [int] NOT NULL,
	[Descripcion] [varchar](200) NOT NULL,
	[Valor] [decimal](18, 2) NOT NULL,
```

```
	[FechaInicio] [datetime] NOT NULL,
	[FechaFinal] [datetime] NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_ComplementosNoSalariales] PRIMARY KEY CLUSTERED
(
	[IDComplementoNoSalarial] ASC,
	[IDTipoDieta] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[TiposIncidencia]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[TiposIncidencia](
	[IDTipoIncidencia] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](50) NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_TipoIncidencia] PRIMARY KEY CLUSTERED
(
	[IDTipoIncidencia] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Prioridades]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Prioridades](
	[IDPrioridad] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](50) NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_Prioridades] PRIMARY KEY CLUSTERED
(
	[IDPrioridad] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Servicios]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Servicios](
	[IDServicio] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](200) NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_Servicios] PRIMARY KEY CLUSTERED
(
	[IDServicio] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[TiposFacturacion]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[TiposFacturacion](
	[IDTipoFacturacion] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](200) NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_TiposFacturacion] PRIMARY KEY CLUSTERED
(
	[IDTipoFacturacion] ASC
```

```
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Clientes]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Clientes](
        [IDCliente] [int] IDENTITY(1,1) NOT NULL,
        [Nombre] [varchar](50) NULL,
        [Direccion] [varchar](200) NULL,
        [Poblacion] [varchar](50) NULL,
        [Provincia] [varchar](50) NULL,
        [Telefono] [varchar](50) NULL,
        [Fax] [varchar](50) NULL,
        [Logo] [varchar](max) NULL,
        [Eliminado] [bit] NULL,
 CONSTRAINT [PK_Cliente] PRIMARY KEY CLUSTERED
(
        [IDCliente] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Historial_backups]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Historial_backups](
        [ID_HISTORIAL_BACKUP] [int] NOT NULL,
        [ID_BACKUP] [int] NOT NULL,
        [ID_CLIENTE] [int] NOT NULL,
        [HOST_CLIENTE] [varchar](50) NOT NULL,
        [NOMBRE_BACKUP] [varchar](50) NOT NULL,
        [FECHA_INICIO] [datetime] NOT NULL,
        [FECHA_FIN] [datetime] NULL,
        [RESULTADO_BACKUP] [int] NOT NULL,
        [RESULTADO_COPIA] [varchar](50) NOT NULL,
        [RESULTADO_ENVIO] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Historial_backups] PRIMARY KEY CLUSTERED
(
        [ID_HISTORIAL_BACKUP] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Historial_Jobs]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Historial_Jobs](
        [ID_HISTORIAL_JOB] [int] NOT NULL,
        [ID_BACKUP] [int] NOT NULL,
        [NOMBRE_JOB] [varchar](50) NOT NULL,
        [FECHA_INICIO] [datetime] NOT NULL,
        [FECHA_FIN] [datetime] NULL,
        [TAMANO] [int] NOT NULL,
        [RESULTADO_JOB] [int] NOT NULL,
        [RESULTADO_COPIA] [varchar](50) NOT NULL,
        [RESULTADO_ENVIO] [varchar](50) NOT NULL,
        [ID_CLIENTE] [int] NOT NULL CONSTRAINT [DF_Historial_Jobs_ID_CLIENTE]  DEFAULT
((-1)),
        [HOST_CLIENTE] [varchar](150) NOT NULL CONSTRAINT
[DF_Historial_Jobs_HOST_CLIENTE]  DEFAULT ('Oficina sin determinar'),
 CONSTRAINT [PK_HISTORIAL_JOBS] PRIMARY KEY CLUSTERED
(
        [ID_HISTORIAL_JOB] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
```

```
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[TiposPermisos]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[TiposPermisos](
	[IDTipoPermiso] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](200) NOT NULL,
	[NumTotalDias] [int] NOT NULL,
	[DiaCompleto] [bit] NOT NULL,
	[Laborables] [bit] NOT NULL,
	[RequiereAprobacion] [bit] NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_TiposPermisos] PRIMARY KEY CLUSTERED
(
	[IDTipoPermiso] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Incidencias]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Incidencias](
	[IDIncidencia] [int] IDENTITY(1,1) NOT NULL,
	[IDInformador] [int] NOT NULL,
	[IDAsignada] [int] NOT NULL,
	[IDCliente] [int] NULL,
	[IDContrato] [int] NOT NULL,
	[IDTipoIncidencia] [int] NOT NULL,
	[IDPrioridad] [int] NOT NULL,
	[Titulo] [varchar](max) NOT NULL,
	[Descripcion] [varchar](max) NOT NULL,
	[RealizableRemoto] [bit] NULL,
	[TiempoEstimado] [float] NULL,
	[FechaAlta] [datetime] NULL,
	[Visibilidad] [bit] NOT NULL,
	[Estado] [varchar](50) NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_Incidencias] PRIMARY KEY CLUSTERED
(
	[IDIncidencia] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Perfiles]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Perfiles](
	[IDPerfil] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](50) NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_Perfiles] PRIMARY KEY CLUSTERED
(
	[IDPerfil] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Provincias]') AND type in (N'U'))
BEGIN
```

```
CREATE TABLE [dbo].[Provincias](
	[IDProvincia] [int] NOT NULL,
	[Provincia] [varchar](255) NOT NULL,
 CONSTRAINT [PK_Provincias] PRIMARY KEY CLUSTERED
(
	[IDProvincia] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Grupos]')
AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Grupos](
	[IDGrupo] [int] IDENTITY(1,1) NOT NULL,
	[Descripcion] [varchar](50) NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_Grupos] PRIMARY KEY CLUSTERED
(
	[IDGrupo] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Gestores]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Gestores](
	[IDGestor] [int] IDENTITY(1,1) NOT NULL,
	[Nombre] [varchar](50) NULL,
	[URL] [varchar](max) NULL,
	[Login] [varchar](50) NULL,
	[Password] [varchar](50) NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_Gestores] PRIMARY KEY CLUSTERED
(
	[IDGestor] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[PermisosModuloUsuario]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[PermisosModuloUsuario](
	[IDUsuario] [int] NOT NULL,
	[IDModulo] [int] NOT NULL,
	[Escritura] [bit] NULL,
 CONSTRAINT [PK_PermisosModuloUsuario] PRIMARY KEY CLUSTERED
(
	[IDUsuario] ASC,
	[IDModulo] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[PermisosModuloPerfil]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[PermisosModuloPerfil](
	[IDPerfil] [int] NOT NULL,
	[IDModulo] [int] NOT NULL,
	[Escritura] [bit] NULL,
```

```
 CONSTRAINT [PK_PermisosModuloPerfil] PRIMARY KEY CLUSTERED
(
	[IDPerfil] ASC,
	[IDModulo] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Log]') AND
type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Log](
	[IDLog] [int] IDENTITY(1,1) NOT NULL,
	[IDSesion] [int] NOT NULL,
	[Fecha] [datetime] NOT NULL,
	[Evento] [varchar](200) NOT NULL,
	[Pagina] [varchar](100) NULL,
	[Nivel] [int] NULL,
 CONSTRAINT [PK_Log] PRIMARY KEY CLUSTERED
(
	[IDLog] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[DietasDetalle]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[DietasDetalle](
	[IDDietaDetalle] [int] IDENTITY(1,1) NOT NULL,
	[IDDieta] [int] NOT NULL,
	[IDTipoDieta] [int] NOT NULL,
	[IDComplementoNoSalarial] [int] NULL,
	[Albaran] [varchar](50) NULL,
	[Valor] [decimal](6, 2) NULL,
	[ValorConvenio] [decimal](6, 2) NULL,
	[Descripcion] [varchar](200) NULL,
	[Observaciones] [varchar](500) NULL,
	[NombreImagen] [varchar](200) NULL
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[TarifasServicios]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[TarifasServicios](
	[IDTarifaServicio] [int] IDENTITY(1,1) NOT NULL,
	[IDServicio] [int] NULL,
	[Nombre] [varchar](50) NULL,
	[PrecioHora] [decimal](18, 2) NULL,
	[FechaInicio] [datetime] NULL,
	[FechaFin] [datetime] NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_TarifasServicios] PRIMARY KEY CLUSTERED
(
	[IDTarifaServicio] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Contratos]') AND type in (N'U'))
BEGIN
```

```
CREATE TABLE [dbo].[Contratos](
	[IDContrato] [int] IDENTITY(1,1) NOT NULL,
	[IDCliente] [int] NULL,
	[IDTipoFacturacion] [int] NULL,
	[Titulo] [varchar](150) NULL,
	[Descripcion] [varchar](500) NULL,
	[FechaInicio] [datetime] NULL,
	[FechaFin] [datetime] NULL,
	[FechaAlta] [datetime] NULL,
	[Estado] [varchar](150) NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_Contratos] PRIMARY KEY CLUSTERED
(
	[IDContrato] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ClientesISPConfig]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[ClientesISPConfig](
	[IDCliente] [int] NOT NULL,
	[IDISPConfig] [int] NOT NULL,
 CONSTRAINT [PK_ClientesISPConfig] PRIMARY KEY CLUSTERED
(
	[IDCliente] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Dominios]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Dominios](
	[IDDominio] [int] IDENTITY(1,1) NOT NULL,
	[IDGestor] [int] NOT NULL,
	[IDCliente] [int] NOT NULL,
	[Nombre] [varchar](50) NULL,
	[ClaveFTP] [varchar](50) NULL,
	[Usuario] [varchar](50) NULL,
	[Contraseña] [varchar](50) NULL,
	[FechaAlta] [datetime] NULL,
	[FechaVencimiento] [datetime] NULL,
	[FechaRenovacion] [datetime] NULL,
	[EspacioTotalCuentas] [float] NULL,
	[Observaciones] [varchar](max) NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_Dominios] PRIMARY KEY CLUSTERED
(
	[IDDominio] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Usuarios]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Usuarios](
	[IDUsuario] [int] IDENTITY(1,1) NOT NULL,
	[IDCliente] [int] NULL,
	[Nombre] [varchar](50) NULL,
	[Login] [varchar](50) NULL,
	[Password] [varchar](50) NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_Usuarios] PRIMARY KEY CLUSTERED
(
```

```
	[IDUsuario] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[ContratoServicios]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[ContratoServicios](
	[IDContrato] [int] NOT NULL,
	[IDServicio] [int] NOT NULL,
	[NumHoras] [int] NULL,
	[PrecioHora] [decimal](18, 2) NULL,
	[Desplazamiento] [bit] NULL,
	[Descripcion] [varchar](500) NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_ContratoServicios] PRIMARY KEY CLUSTERED
(
	[IDContrato] ASC,
	[IDServicio] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id = OBJECT_ID(N'[dbo].[Dietas]')
AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Dietas](
	[IDDieta] [int] IDENTITY(1,1) NOT NULL,
	[IDUsuario] [int] NOT NULL,
	[Estado] [int] NULL,
	[Fecha] [datetime] NULL,
	[TotalKM] [int] NULL,
	[TotalImporte] [decimal](6, 2) NULL,
	[TotalHoras] [decimal](6, 2) NULL,
 CONSTRAINT [PK_Dietas] PRIMARY KEY CLUSTERED
(
	[IDDieta] ASC,
	[IDUsuario] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'1' , N'SCHEMA',N'dbo',
N'TABLE',N'Dietas', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'1', @value=N'Sin Validar' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'Dietas',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'2' , N'SCHEMA',N'dbo',
N'TABLE',N'Dietas', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'2', @value=N'Validada por el usuario' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'Dietas',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'3' , N'SCHEMA',N'dbo',
N'TABLE',N'Dietas', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'3', @value=N'Autorizada' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'Dietas',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'4' , N'SCHEMA',N'dbo',
N'TABLE',N'Dietas', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'4', @value=N'Incompleta' ,
@level0type=N'SCHEMA',@level0name=N'dbo', @level1type=N'TABLE',@level1name=N'Dietas',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[PerfilesUsuario]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[PerfilesUsuario](
	[IDPerfil] [int] NOT NULL,
	[IDUsuario] [int] NOT NULL,
 CONSTRAINT [PK_PerfilesUsuario] PRIMARY KEY CLUSTERED
(
	[IDPerfil] ASC,
	[IDUsuario] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[GruposUsuarios]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[GruposUsuarios](
	[IDGrupo] [int] NOT NULL,
	[IDUsuario] [int] NOT NULL,
 CONSTRAINT [PK_GruposUsuarios] PRIMARY KEY CLUSTERED
(
	[IDGrupo] ASC,
	[IDUsuario] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[BolsaVacaciones]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[BolsaVacaciones](
	[IDUsuario] [int] NOT NULL,
	[DiasAcumulados] [smallint] NOT NULL
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[PermisosRetribuidos]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[PermisosRetribuidos](
	[IDPermisoRetribuido] [int] IDENTITY(1,1) NOT NULL,
	[IDUsuario] [int] NOT NULL,
	[IDTipoPermiso] [int] NOT NULL,
	[FechaInicio] [datetime] NOT NULL,
	[FechaFin] [datetime] NOT NULL,
	[Comentarios] [varchar](200) NOT NULL,
	[Estado] [smallint] NOT NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_PermisosRetribuidos] PRIMARY KEY CLUSTERED
(
	[IDPermisoRetribuido] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'Autorizado' , N'SCHEMA',N'dbo',
N'TABLE',N'PermisosRetribuidos', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'Autorizado', @value=N'2' ,
@level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'PermisosRetribuidos',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'Denegado' , N'SCHEMA',N'dbo',
N'TABLE',N'PermisosRetribuidos', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'Denegado', @value=N'3' ,
@level0type=N'SCHEMA',@level0name=N'dbo',
```

```
@level1type=N'TABLE',@level1name=N'PermisosRetribuidos',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
IF NOT EXISTS (SELECT * FROM ::fn_listextendedproperty(N'Solicitado' , N'SCHEMA',N'dbo',
N'TABLE',N'PermisosRetribuidos', N'COLUMN',N'Estado'))
EXEC sys.sp_addextendedproperty @name=N'Solicitado', @value=N'1' ,
@level0type=N'SCHEMA',@level0name=N'dbo',
@level1type=N'TABLE',@level1name=N'PermisosRetribuidos',
@level2type=N'COLUMN',@level2name=N'Estado'
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[DatosPersonales]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[DatosPersonales](
        [IDUsuario] [int] NOT NULL,
        [Direccion] [varchar](150) NULL,
        [Telefono] [varchar](20) NULL,
        [NIF] [varchar](10) NULL,
 CONSTRAINT [PK_DatosPersonales] PRIMARY KEY CLUSTERED
(
        [IDUsuario] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Sesiones]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Sesiones](
        [IDSesion] [int] IDENTITY(1,1) NOT NULL,
        [IDUsuario] [int] NOT NULL,
        [HoraInicio] [datetime] NOT NULL,
        [HoraFin] [datetime] NULL,
        [UltimoEvento] [datetime] NOT NULL,
        [Estado] [bit] NOT NULL,
 CONSTRAINT [PK_Sesiones] PRIMARY KEY CLUSTERED
(
        [IDSesion] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Intervenciones]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Intervenciones](
        [IDIntervencion] [int] IDENTITY(1,1) NOT NULL,
        [IDUsuario] [int] NOT NULL,
        [IDIncidencia] [int] NOT NULL,
        [Fecha] [datetime] NULL,
        [Parte] [varchar](50) NULL,
        [NumHoras] [float] NULL,
        [Motivo] [varchar](max) NULL,
        [Observaciones] [varchar](max) NULL,
        [Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_Intervenciones] PRIMARY KEY CLUSTERED
(
        [IDIntervencion] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[DocumentosIncidencia]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[DocumentosIncidencia](
	[IDDocumento] [int] IDENTITY(1,1) NOT NULL,
	[IDIncidencia] [int] NOT NULL,
	[Descripcion] [varchar](max) NULL,
	[Ruta] [varchar](max) NULL,
	[Eliminado] [bit] NOT NULL,
 CONSTRAINT [PK_DocumentosIncidencia] PRIMARY KEY CLUSTERED
(
	[IDDocumento] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[RelacionesIncidencias]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[RelacionesIncidencias](
	[IDRelacion] [int] IDENTITY(1,1) NOT NULL,
	[IDIncidenciaMain] [int] NOT NULL,
	[IDIncidenciaSub] [int] NOT NULL,
	[IDTipoRelacion] [int] NOT NULL,
	[Fecha] [datetime] NULL,
	[IDUsuario] [int] NULL,
	[Eliminado] [bit] NULL,
 CONSTRAINT [PK_RelacionesIncidencias] PRIMARY KEY CLUSTERED
(
	[IDRelacion] ASC
)WITH (PAD_INDEX  = OFF, IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PermisosModuloUsuario_Modulos]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PermisosModuloUsuario]'))
ALTER TABLE [dbo].[PermisosModuloUsuario]  WITH CHECK ADD  CONSTRAINT
[FK_PermisosModuloUsuario_Modulos] FOREIGN KEY([IDModulo])
REFERENCES [dbo].[Modulos] ([IDModulo])
GO
ALTER TABLE [dbo].[PermisosModuloUsuario] CHECK CONSTRAINT
[FK_PermisosModuloUsuario_Modulos]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PermisosModuloUsuario_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PermisosModuloUsuario]'))
ALTER TABLE [dbo].[PermisosModuloUsuario]  WITH CHECK ADD  CONSTRAINT
[FK_PermisosModuloUsuario_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[PermisosModuloUsuario] CHECK CONSTRAINT
[FK_PermisosModuloUsuario_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PermisosModuloPerfil_Modulos]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PermisosModuloPerfil]'))
ALTER TABLE [dbo].[PermisosModuloPerfil]  WITH CHECK ADD  CONSTRAINT
[FK_PermisosModuloPerfil_Modulos] FOREIGN KEY([IDModulo])
REFERENCES [dbo].[Modulos] ([IDModulo])
GO
ALTER TABLE [dbo].[PermisosModuloPerfil] CHECK CONSTRAINT
[FK_PermisosModuloPerfil_Modulos]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PermisosModuloPerfil_Perfiles]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PermisosModuloPerfil]'))
ALTER TABLE [dbo].[PermisosModuloPerfil]  WITH CHECK ADD  CONSTRAINT
[FK_PermisosModuloPerfil_Perfiles] FOREIGN KEY([IDPerfil])
REFERENCES [dbo].[Perfiles] ([IDPerfil])
GO
ALTER TABLE [dbo].[PermisosModuloPerfil] CHECK CONSTRAINT
[FK_PermisosModuloPerfil_Perfiles]
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Log_Sesiones]') AND parent_object_id = OBJECT_ID(N'[dbo].[Log]'))
ALTER TABLE [dbo].[Log]  WITH CHECK ADD  CONSTRAINT [FK_Log_Sesiones] FOREIGN
KEY([IDSesion])
REFERENCES [dbo].[Sesiones] ([IDSesion])
GO
ALTER TABLE [dbo].[Log] CHECK CONSTRAINT [FK_Log_Sesiones]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_DietasDetalle_TiposDieta]') AND parent_object_id =
OBJECT_ID(N'[dbo].[DietasDetalle]'))
ALTER TABLE [dbo].[DietasDetalle]  WITH CHECK ADD  CONSTRAINT
[FK_DietasDetalle_TiposDieta] FOREIGN KEY([IDTipoDieta])
REFERENCES [dbo].[TiposDieta] ([IDTipoDieta])
GO
ALTER TABLE [dbo].[DietasDetalle] CHECK CONSTRAINT [FK_DietasDetalle_TiposDieta]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_TarifasServicios_Servicios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[TarifasServicios]'))
ALTER TABLE [dbo].[TarifasServicios]  WITH CHECK ADD  CONSTRAINT
[FK_TarifasServicios_Servicios] FOREIGN KEY([IDServicio])
REFERENCES [dbo].[Servicios] ([IDServicio])
GO
ALTER TABLE [dbo].[TarifasServicios] CHECK CONSTRAINT [FK_TarifasServicios_Servicios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Contratos_TiposFacturacion]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Contratos]'))
ALTER TABLE [dbo].[Contratos]  WITH CHECK ADD  CONSTRAINT
[FK_Contratos_TiposFacturacion] FOREIGN KEY([IDTipoFacturacion])
REFERENCES [dbo].[TiposFacturacion] ([IDTipoFacturacion])
GO
ALTER TABLE [dbo].[Contratos] CHECK CONSTRAINT [FK_Contratos_TiposFacturacion]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_ClientesISPConfig_Clientes]') AND parent_object_id =
OBJECT_ID(N'[dbo].[ClientesISPConfig]'))
ALTER TABLE [dbo].[ClientesISPConfig]  WITH CHECK ADD  CONSTRAINT
[FK_ClientesISPConfig_Clientes] FOREIGN KEY([IDCliente])
REFERENCES [dbo].[Clientes] ([IDCliente])
GO
ALTER TABLE [dbo].[ClientesISPConfig] CHECK CONSTRAINT [FK_ClientesISPConfig_Clientes]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Dominios_Cliente]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Dominios]'))
ALTER TABLE [dbo].[Dominios]  WITH CHECK ADD  CONSTRAINT [FK_Dominios_Cliente] FOREIGN
KEY([IDCliente])
REFERENCES [dbo].[Clientes] ([IDCliente])
GO
ALTER TABLE [dbo].[Dominios] CHECK CONSTRAINT [FK_Dominios_Cliente]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Dominios_Dominios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Dominios]'))
ALTER TABLE [dbo].[Dominios]  WITH CHECK ADD  CONSTRAINT [FK_Dominios_Dominios] FOREIGN
KEY([IDDominio])
REFERENCES [dbo].[Gestores] ([IDGestor])
GO
ALTER TABLE [dbo].[Dominios] CHECK CONSTRAINT [FK_Dominios_Dominios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Usuarios_Clientes]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Usuarios]'))
ALTER TABLE [dbo].[Usuarios]  WITH CHECK ADD  CONSTRAINT [FK_Usuarios_Clientes] FOREIGN
KEY([IDCliente])
REFERENCES [dbo].[Clientes] ([IDCliente])
GO
ALTER TABLE [dbo].[Usuarios] CHECK CONSTRAINT [FK_Usuarios_Clientes]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_ContratoServicios_Contratos]') AND parent_object_id =
OBJECT_ID(N'[dbo].[ContratoServicios]'))
ALTER TABLE [dbo].[ContratoServicios]  WITH CHECK ADD  CONSTRAINT
[FK_ContratoServicios_Contratos] FOREIGN KEY([IDContrato])
REFERENCES [dbo].[Contratos] ([IDContrato])
GO
ALTER TABLE [dbo].[ContratoServicios] CHECK CONSTRAINT [FK_ContratoServicios_Contratos]
```

```
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Dietas_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Dietas]'))
ALTER TABLE [dbo].[Dietas]  WITH CHECK ADD  CONSTRAINT [FK_Dietas_Usuarios] FOREIGN
KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[Dietas] CHECK CONSTRAINT [FK_Dietas_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PerfilesUsuario_Perfiles]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PerfilesUsuario]'))
ALTER TABLE [dbo].[PerfilesUsuario]  WITH CHECK ADD  CONSTRAINT
[FK_PerfilesUsuario_Perfiles] FOREIGN KEY([IDPerfil])
REFERENCES [dbo].[Perfiles] ([IDPerfil])
GO
ALTER TABLE [dbo].[PerfilesUsuario] CHECK CONSTRAINT [FK_PerfilesUsuario_Perfiles]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PerfilesUsuario_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PerfilesUsuario]'))
ALTER TABLE [dbo].[PerfilesUsuario]  WITH CHECK ADD  CONSTRAINT
[FK_PerfilesUsuario_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[PerfilesUsuario] CHECK CONSTRAINT [FK_PerfilesUsuario_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_GruposUsuarios_Grupos]') AND parent_object_id =
OBJECT_ID(N'[dbo].[GruposUsuarios]'))
ALTER TABLE [dbo].[GruposUsuarios]  WITH CHECK ADD  CONSTRAINT
[FK_GruposUsuarios_Grupos] FOREIGN KEY([IDGrupo])
REFERENCES [dbo].[Grupos] ([IDGrupo])
GO
ALTER TABLE [dbo].[GruposUsuarios] CHECK CONSTRAINT [FK_GruposUsuarios_Grupos]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_GruposUsuarios_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[GruposUsuarios]'))
ALTER TABLE [dbo].[GruposUsuarios]  WITH CHECK ADD  CONSTRAINT
[FK_GruposUsuarios_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[GruposUsuarios] CHECK CONSTRAINT [FK_GruposUsuarios_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_DiasAcumulados_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[BolsaVacaciones]'))
ALTER TABLE [dbo].[BolsaVacaciones]  WITH CHECK ADD  CONSTRAINT
[FK_DiasAcumulados_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[BolsaVacaciones] CHECK CONSTRAINT [FK_DiasAcumulados_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PermisosRetribuidos_TiposPermisos]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PermisosRetribuidos]'))
ALTER TABLE [dbo].[PermisosRetribuidos]  WITH CHECK ADD  CONSTRAINT
[FK_PermisosRetribuidos_TiposPermisos] FOREIGN KEY([IDTipoPermiso])
REFERENCES [dbo].[TiposPermisos] ([IDTipoPermiso])
GO
ALTER TABLE [dbo].[PermisosRetribuidos] CHECK CONSTRAINT
[FK_PermisosRetribuidos_TiposPermisos]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_PermisosRetribuidos_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[PermisosRetribuidos]'))
ALTER TABLE [dbo].[PermisosRetribuidos]  WITH CHECK ADD  CONSTRAINT
[FK_PermisosRetribuidos_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[PermisosRetribuidos] CHECK CONSTRAINT
[FK_PermisosRetribuidos_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_DatosPersonales_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[DatosPersonales]'))
```

```
ALTER TABLE [dbo].[DatosPersonales]  WITH CHECK ADD  CONSTRAINT
[FK_DatosPersonales_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[DatosPersonales] CHECK CONSTRAINT [FK_DatosPersonales_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Sesiones_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Sesiones]'))
ALTER TABLE [dbo].[Sesiones]  WITH CHECK ADD  CONSTRAINT [FK_Sesiones_Usuarios] FOREIGN
KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[Sesiones] CHECK CONSTRAINT [FK_Sesiones_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Intervenciones_Incidencias]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Intervenciones]'))
ALTER TABLE [dbo].[Intervenciones]  WITH CHECK ADD  CONSTRAINT
[FK_Intervenciones_Incidencias] FOREIGN KEY([IDIncidencia])
REFERENCES [dbo].[Incidencias] ([IDIncidencia])
GO
ALTER TABLE [dbo].[Intervenciones] CHECK CONSTRAINT [FK_Intervenciones_Incidencias]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_Intervenciones_Usuarios]') AND parent_object_id =
OBJECT_ID(N'[dbo].[Intervenciones]'))
ALTER TABLE [dbo].[Intervenciones]  WITH CHECK ADD  CONSTRAINT
[FK_Intervenciones_Usuarios] FOREIGN KEY([IDUsuario])
REFERENCES [dbo].[Usuarios] ([IDUsuario])
GO
ALTER TABLE [dbo].[Intervenciones] CHECK CONSTRAINT [FK_Intervenciones_Usuarios]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_DocumentosIncidencia_Incidencias]') AND parent_object_id =
OBJECT_ID(N'[dbo].[DocumentosIncidencia]'))
ALTER TABLE [dbo].[DocumentosIncidencia]  WITH CHECK ADD  CONSTRAINT
[FK_DocumentosIncidencia_Incidencias] FOREIGN KEY([IDIncidencia])
REFERENCES [dbo].[Incidencias] ([IDIncidencia])
GO
ALTER TABLE [dbo].[DocumentosIncidencia] CHECK CONSTRAINT
[FK_DocumentosIncidencia_Incidencias]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_RelacionesIncidencias_Incidencias]') AND parent_object_id =
OBJECT_ID(N'[dbo].[RelacionesIncidencias]'))
ALTER TABLE [dbo].[RelacionesIncidencias]  WITH CHECK ADD  CONSTRAINT
[FK_RelacionesIncidencias_Incidencias] FOREIGN KEY([IDIncidenciaMain])
REFERENCES [dbo].[Incidencias] ([IDIncidencia])
GO
ALTER TABLE [dbo].[RelacionesIncidencias] CHECK CONSTRAINT
[FK_RelacionesIncidencias_Incidencias]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_RelacionesIncidencias_Incidencias1]') AND parent_object_id =
OBJECT_ID(N'[dbo].[RelacionesIncidencias]'))
ALTER TABLE [dbo].[RelacionesIncidencias]  WITH CHECK ADD  CONSTRAINT
[FK_RelacionesIncidencias_Incidencias1] FOREIGN KEY([IDIncidenciaSub])
REFERENCES [dbo].[Incidencias] ([IDIncidencia])
GO
ALTER TABLE [dbo].[RelacionesIncidencias] CHECK CONSTRAINT
[FK_RelacionesIncidencias_Incidencias1]
GO
IF NOT EXISTS (SELECT * FROM sys.foreign_keys WHERE object_id =
OBJECT_ID(N'[dbo].[FK_RelacionesIncidencias_RelacionesIncidencias]') AND
parent_object_id = OBJECT_ID(N'[dbo].[RelacionesIncidencias]'))
ALTER TABLE [dbo].[RelacionesIncidencias]  WITH CHECK ADD  CONSTRAINT
[FK_RelacionesIncidencias_RelacionesIncidencias] FOREIGN KEY([IDRelacion])
REFERENCES [dbo].[RelacionesIncidencias] ([IDRelacion])
GO
ALTER TABLE [dbo].[RelacionesIncidencias] CHECK CONSTRAINT
[FK_RelacionesIncidencias_RelacionesIncidencias]
```

Con la estructura de tablas ya establecida necesitamos introducir los datos básicos que la aplicación necesita para funcionar. Este paso no se puede incluir en el script de generación anterior puesto que la versión 2005 de Microsoft SQL Server no dispone de esta función. Estos datos son:

9.1.3.1 *Tabla Modulos*

IDModulo	Nombre
1	Gestión de Usuarios
2	Gestión de Dominios
3	Gestión de Vacaciones
4	Gestión de Dietas
5	Gestión de Contratos
6	Gestión de Servicios
7	Gestión de Incidencias
8	Gestión de Clientes
9	Gestión de Backups

9.1.3.2 *Tabla Perfiles*

IDPerfil	Descripcion	Eliminado
1	Administrador	0
2	Desarrollador	0
3	Técnico	0
4	Administración	0
5	Organizador	0

9.1.3.3 *Tabla Usuarios*

IDModulo	IDCliente	Nombre	Login	Password	Eliminado
1	1	Administrador	Admin		0

9.1.3.4 *Tabla PerfilesUsuario*

IDPerfil	IDUsuario
1	1

9.1.3.5 *Tabla PermisosModuloUsuario*

IDUsuario	IDModulo	Escritura
1	1	1
1	2	1
1	3	1
1	4	1
1	5	1
1	6	1
1	7	1
1	8	1

1	9	1

9.1.3.6 *Tabla PermisosModuloPerfil*

IDPerfil	IDModulo	Escritura
1	1	1
1	2	1
1	3	1
1	4	1
1	5	1
1	6	1
1	7	1
1	8	1
1	9	1

Con estos datos básicos introducidos podemos identificarnos en la aplicación y comenzar a introducir todos los datos necesarios desde la misma.

9.1.4 Servidor Web

El siguiente paso es preparar el entorno en el que se ejecutará la aplicación, es decir, el servidor web Apache Tomcat. Dado que es necesario modificar una serie de parámetros de configuración y adjuntar librerías adicionales, el proceso se torna muy tedioso de explicar y aplicar paso a paso. Por eso se incluye en el directorio de software del CD ya comprimido todo el entorno del Apache Tomcat.

Para ponerlo en funcionamiento basta con descomprimirlo en una carpeta cualquiera del disco duro (por ejemplo, `C:\apache-tomcat\`). Cabe destacar que está configurado para que las aplicaciones se ejecuten sobre el puerto 8080, por lo que el firewall del equipo debe permitir las conexiones entrantes en dicho puerto.

Las configuraciones adicionales que se han llevado a cabo en el servidor proporcionado son las siguientes:

```
<Resource auth="Container"
driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver" maxActive="100"
maxIdle="120" name="jdbc/Ecocomputer" removeAbandoned="true" type="javax.sql.DataSource"
url="jdbc:sqlserver://localhost:1433;DatabaseName=Ecocomputer;user=pfmecocomputer;passwo
rd=test123;"/>
```

Con estas líneas en el archive “Context.xml” de la aplicación conseguimos que el servidor sea el encargado de manejar las conexiones con la base de datos, creando un pool de conexiones.

```
<Connector port="8080" protocol="org.apache.coyote.http11.Http11Protocol"
               connectionTimeout="20000"
               redirectPort="8443" />

<Connector port="8443" SSLEnabled="true"
        protocol="org.apache.coyote.http11.Http11Protocol"
        maxThreads="150" scheme="https" secure="true"
        clientAuth="false" sslProtocol="TLS"
        keystoreFile="C:\keystore\ecocomputer.keystore"
        keystorePass="Eco.123" />
```

Estas líneas se colocan en el archivo “server.xml” y su función es dotar a nuestro servidor del protocolo SSL, de manera que la navegación por nuestra aplicación sea segura. Para conseguir

esto previamente será necesaria la creación de un certificado autofirmado, proceso que se detalla a continuación:

9.1.4.1 *Creación de un certificado autofirmado*

```
C:\Program Files\Java\jdk1.7.0_05\bin>keytool -genkey -alias eco -keyalg RSA -ke
ystore C:\certificados\ecocomputer.keystore
Introduzca la contrase±a del almacún de claves:
Volver a escribir la contrase±a nueva:
¬Cu¿les son su nombre y su apellido?
  [Unknown]:  Eco computer
¬Cu¿l es el nombre de su unidad de organizaci¾n?
  [Unknown]:  Ecocomputer
¬Cu¿l es el nombre de su organizaci¾n?
  [Unknown]:  Ecocomputer
¬Cu¿l es el nombre de su ciudad o localidad?
  [Unknown]:  Oviedo
¬Cu¿l es el nombre de su estado o provincia?
  [Unknown]:  Asturias
¬Cu¿l es el c¾digo de paýs de dos letras de la unidad?
  [Unknown]:  es
¬Es correcto CN=Eco computer, OU=Ecocomputer, O=Ecocomputer, L=Oviedo, ST=Asturi
as, C=es?
  [no]:  s

Introduzca la contrase±a de clave para <eco>
        (INTRO si es la misma contrase±a que la del almacún de claves):
```

Manual instalación 9.6: Creación de certificado autofirmado

Con las ordenes anteriores se crea un fichero autofirmado el cual solo deberemos situar dentro de la carpeta que veamos oportuna y vincularlo desde Tomcat con la configuración mostrada anteriormente.

Aunque para este proyecto el certificado que se ha creado es autofirmado, es aconsejable pagar a una empresa que nos lo genere.

Finalmente, la última parte de configuración a realizar para el servidor será la integración con PHPJavaBridge en el archivo “web.xml”, lo cual permitirá que se ejecute código PHP desde el servidor, cosa que de otra manera sería imposible al tratarse de un servidor Java.

```
        <listener>
                <listener-class>php.java.servlet.ContextLoaderListener</listener-class>
        </listener>

        <servlet>
                <servlet-name>PhpJavaServlet</servlet-name>
                <servlet-class>php.java.servlet.PhpJavaServlet</servlet-class>
        </servlet>

        <servlet>
                <servlet-name>PhpCGIServlet</servlet-name>
                <servlet-class>php.java.servlet.fastcgi.FastCGIServlet</servlet-class>
                <init-param>
                        <param-name>prefer_system_php_exec</param-name>
                        <param-value>On</param-value>
                </init-param>

                <init-param>
                        <param-name>php_include_java</param-name>
                        <param-value>Off</param-value>
                </init-param>
        </servlet>

        <servlet-mapping>
                <servlet-name>PhpJavaServlet</servlet-name>
                <url-pattern>*.phpjavabridge</url-pattern>
        </servlet-mapping>
```

```
        <servlet-mapping>
                <servlet-name>PhpCGIServlet</servlet-name>
                <url-pattern>*.php</url-pattern>
        </servlet-mapping>
```

9.1.5 Instalación del servidor ISPConfig

La instalación y configuración del servidor ISPConfig ha sido llevada a cabo por la empresa, proporcionándonos una máquina virtual con el ya instalado y configurado tal y como ellos lo necesitan. Además debido a que la información de instalación del ISPConfig así como muchos otros puntos se cubren en su manual, el cual es de pago, no se ve procedente utilizar aquí sus secciones.

9.2 Manual de Ejecución

9.2.1 Inicio del servidor web

El proceso de arranque del servidor web es tan sencillo como ejecutar el archivo de lotes `bin\startup.bat` del directorio de Tomcat, siempre y cuando el equipo tenga Java instalado.

9.2.2 Desplegar la aplicación

La configuración que ya tiene preparada el servidor Tomcat hacen que desplegar la aplicación se realice simplemente copiando los archivos `Intranet.war`, `GestionUsuarios.war` y `GestionDominios.war` en la carpeta `webapps\` del directorio donde hayamos descomprimido Tomcat en el apartado anterior. Tras unos segundos, el servidor leerá automáticamente los archivos de distribución de aplicaciones .war y los desplegará.

Requiere mención especial que el archivo Intranet.war no pertenece a este proyecto, pero que es necesario para poder tener acceso a las otras dos aplicaciones que sí pertenecen, puesto que se encarga de realizar el inicio de sesión del usuario.

Una vez desplegado, podemos acceder a la aplicación desde cualquier navegador web a través de la URL http://localhost:8080/Intranet, o bien desde el exterior, siempre y cuando el cortafuegos acepte conexiones externas, desde http://IP-de-la-máquina:8080/Intranet.

9.2.3 Parada del servidor web

De manera similar al arranque del servidor, detenerlo pasa por ejecutar el archivo de lotes `bin\shutdown.bat` del directorio donde previamente descomprimimos Tomcat.

9.3 Manual de Usuario

9.3.1 Inicio de sesión en la aplicación

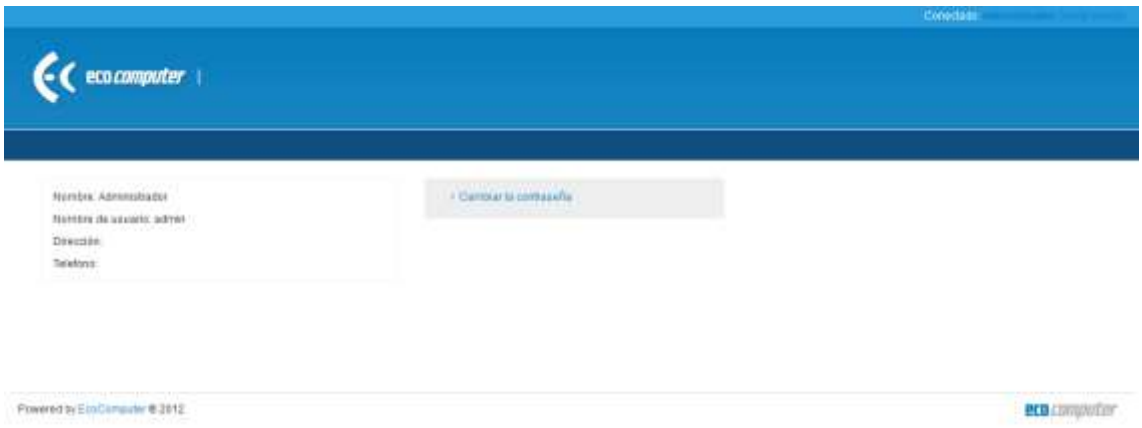
Lo primero que debe hacer un usuario para poder comenzar a utilizar las funcionalidades del proyecto es iniciar sesión en la aplicación. Para ello habrá que acceder desde un navegador web a la dirección en la que haya sido desplegada la Intranet, en nuestro caso http://localhost:8080/Intranet/.

Al acceder nos encontraremos una pantalla como la siguiente, en la cual el usuario deberá especificar su nombre de usuario y contraseña. En el caso de no disponer de ellos deberá ponerse en contacto con el administrador de la Intranet, pues es de acceso privado.

Manual usuario 9.1: Iniciar sesión

9.3.2 Opciones de usuario

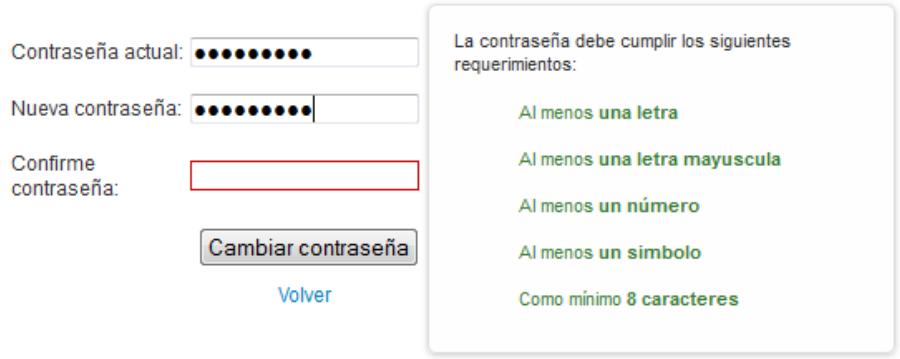
Una vez dentro de la aplicación el usuario se encontrará en su página principal, pudiendo tener disponibles las opciones de cliente o solamente las propias de usuario.



Manual usuario 9.2: Opciones de usuario

9.3.2.1 *Cambio de contraseña propia*

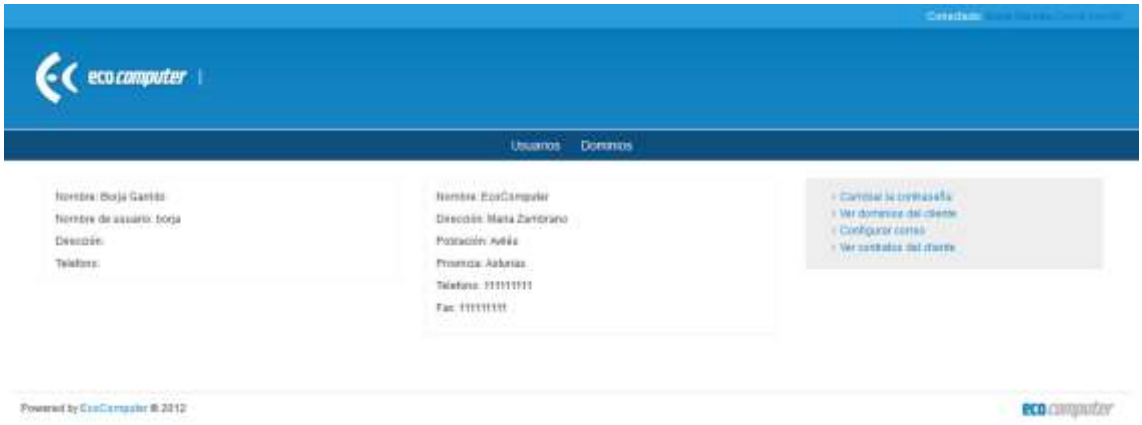
En este caso vemos que como opciones propias del usuario solo se le permite cambiar la contraseña, para ello nos aparece una nueva pantalla en la cual deberemos especificar la contraseña actual y la nueva, repitiéndola varias veces.



Manual usuario 9.3: Cambio de contraseña

9.3.3 Opciones de cliente

En el caso de que el usuario sea a su vez un cliente se le ofrecen opciones extra como podemos ver en esta imagen.

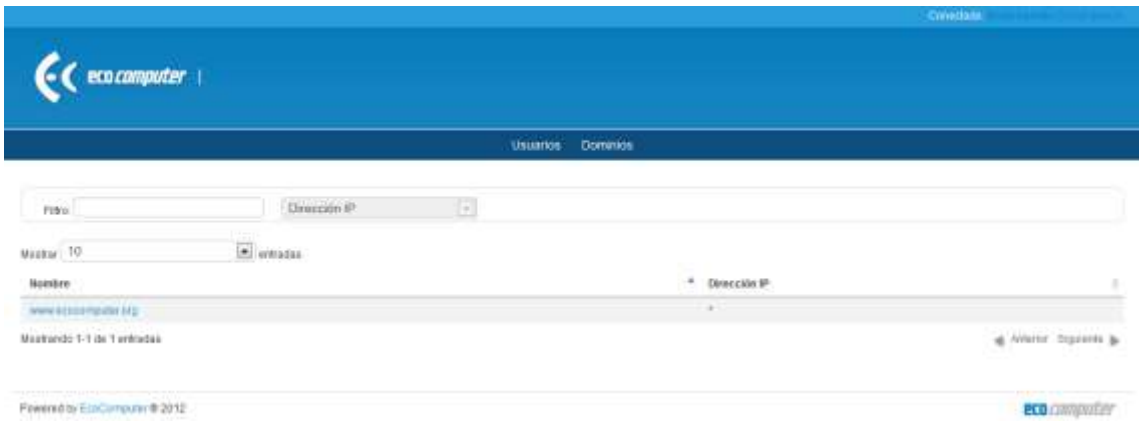


Manual usuario 9.4: Opciones del cliente

De todas las opciones hay una que no pertenece a este proyecto y que por lo tanto no se explicará. Como podemos ver sigue apareciendo la opción de cambiar la contraseña y dos opciones nuevas que se explicarán a continuación.

9.3.3.1 *Consultar dominios*

Un cliente puede o no tener asociados dominios dentro de la aplicación, en caso de ser así se le permite consultarlos y ver su estado.



Manual usuario 9.5: Consultar dominios propios

No está permitido sin embargo realizar ningún cambio sobre ellos, pues no es responsabilidad del cliente.

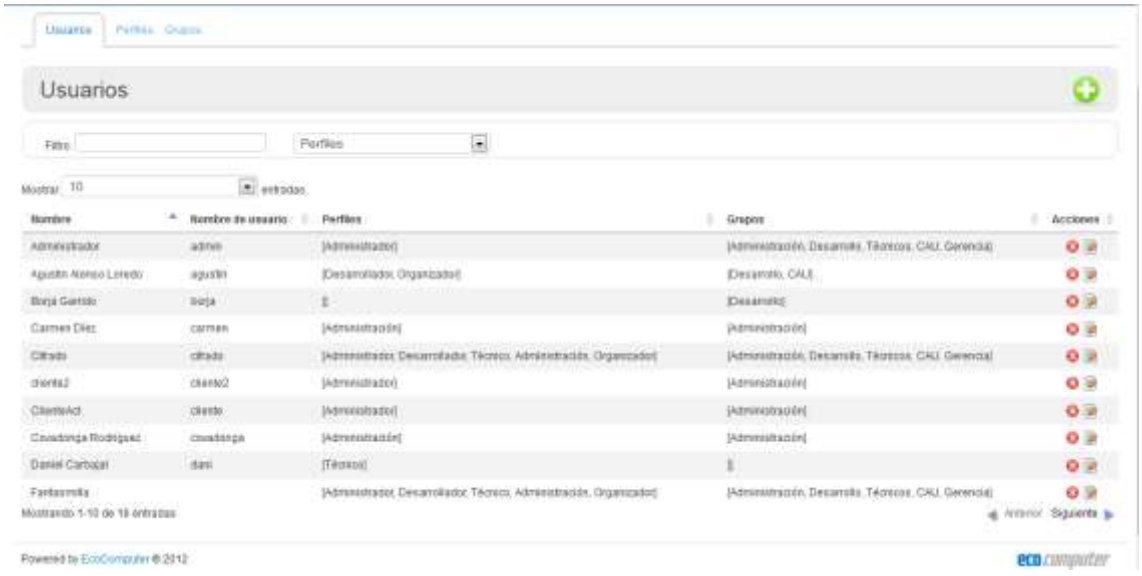
9.3.3.2 *Configurar cuenta de correo*

Además de consultar los dominios se permite al usuario consultar su cuenta de correo, siempre y cuando tenga asociada una al servidor ISP.

9.3.4 Manejo de usuarios

En el módulo de gestión de usuarios se permite la adición de usuarios, su modificación y eliminación, así como los perfiles y grupos del sistema. Nos centraremos en como interactuar con la parte de usuarios pues las demás son similares.

Lo primero que visualiza un usuario al entrar en la gestión de usuarios es la lista de los usuarios del sistema.



Manual usuario 9.6: Visualización de usuarios

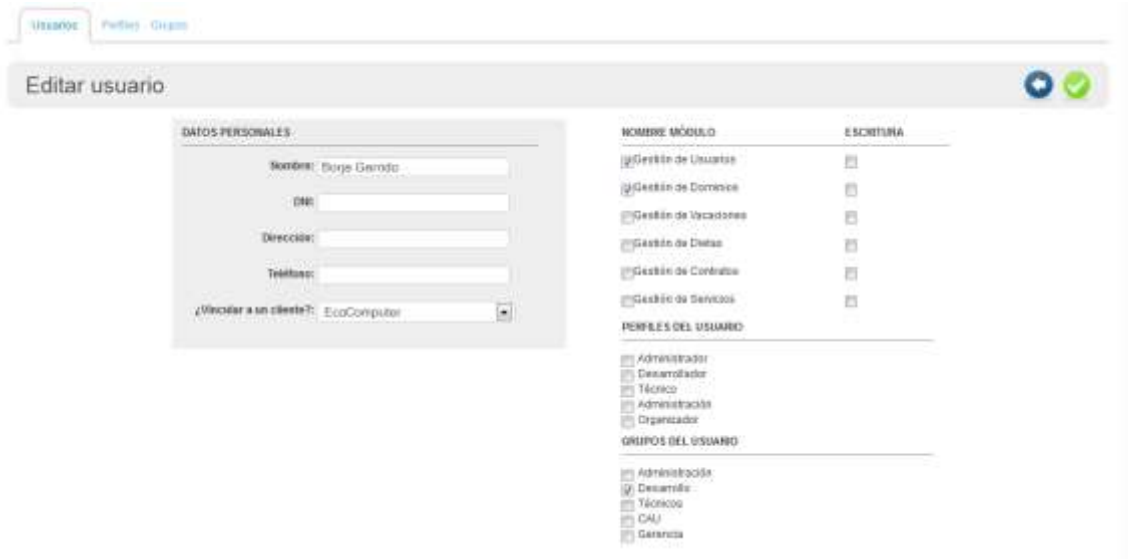
Dentro de esta pantalla un usuario puede añadir pulsando sobre el icono con el símbolo "+", filtrar los usuarios que se muestran en la barra de filtro, modificar los datos de un usuario pulsando sobre el mismo, o desde el menú de acciones eliminarlos y cambiar su contraseña.

9.3.4.1 *Insertar nuevo usuario*

Manual usuario 9.7: Añadir nuevo usuario

Como vemos el proceso de añadir un usuario no tiene nada reseñable, solo hay que rellenar sus datos y seleccionar sus permisos, perfiles y grupos. En cualquier momento se puede cancelar la acción pulsando sobre el botón de "volver" en la barra de tareas o confirmarla pulsando sobre el botón "aceptar".

9.3.4.2 *Modificar usuario*



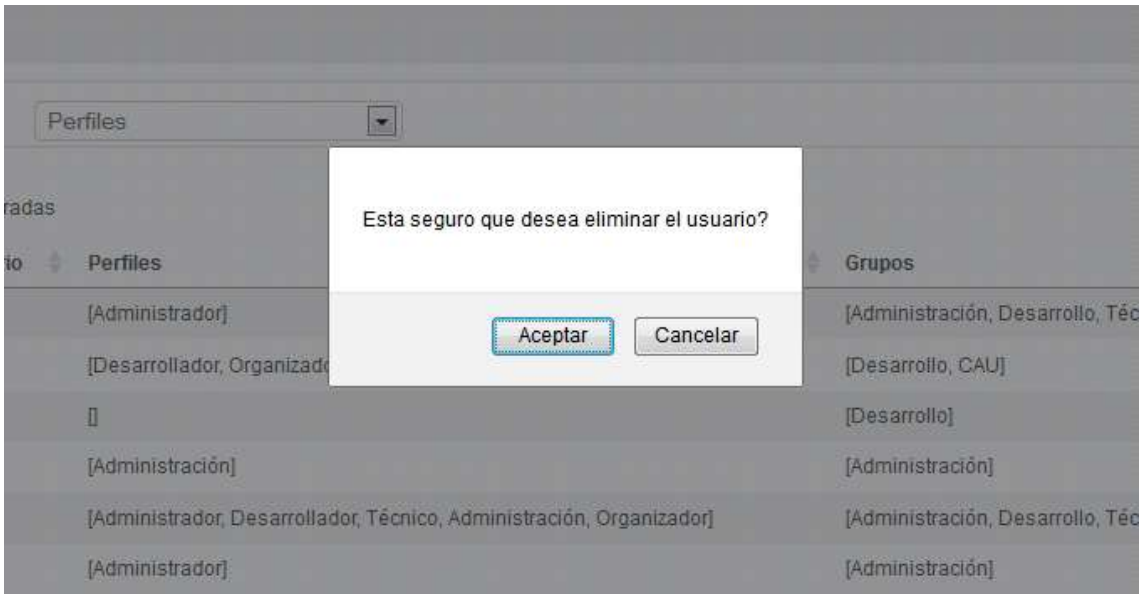
Manual usuario 9.8: Modificar usuario

Sí pulsamos sobre un usuario accederemos a una pantalla donde se permitirá modificarlo, esto carga los datos ya existentes para hacer más fácil el rellenar el formulario por parte del usuario.

Nuevamente se permite abandonar la operación en cualquier momento pulsando sobre "volver". Uno de los datos que no se permite modificar en este formulario es la contraseña, pues no se considera una práctica segura.

9.3.4.3 *Eliminar usuario*

Presionando sobre el botón de las acciones de eliminar usuario aparece un dialogo javascript pidiendo confirmación, en caso de aceptar el usuario desaparece de la lista y en caso contrario se cancela la operación.



Manual usuario 9.9: Eliminar usuario

9.3.5 Manejo de clientes

Dentro de este proyecto solo se permite el manejo de clientes relacionados con los dominios, lo que quiere decir que no es responsabilidad de esta aplicación el crear un objeto cliente en la base de datos. El usuario puede desde ella elegir si crear un cliente nuevo con todos los datos de usuario y cliente, o utilizando alguno de los objetos que ya existe dentro de la base de datos.

Al elegir unos de esos datos automáticamente se cargan en el formulario los datos asociados.

Manual usuario 9.10: Insertar cliente

9.3.6 Manejo de buzones

9.3.6.1 *Añadir buzones*

En el caso de los buzones no se ofrece una página completa para añadirlos, pues sus datos son muy reducidos, por eso para ello se ofrece un formulario en la barra de de herramientas de la página que permite verlos.

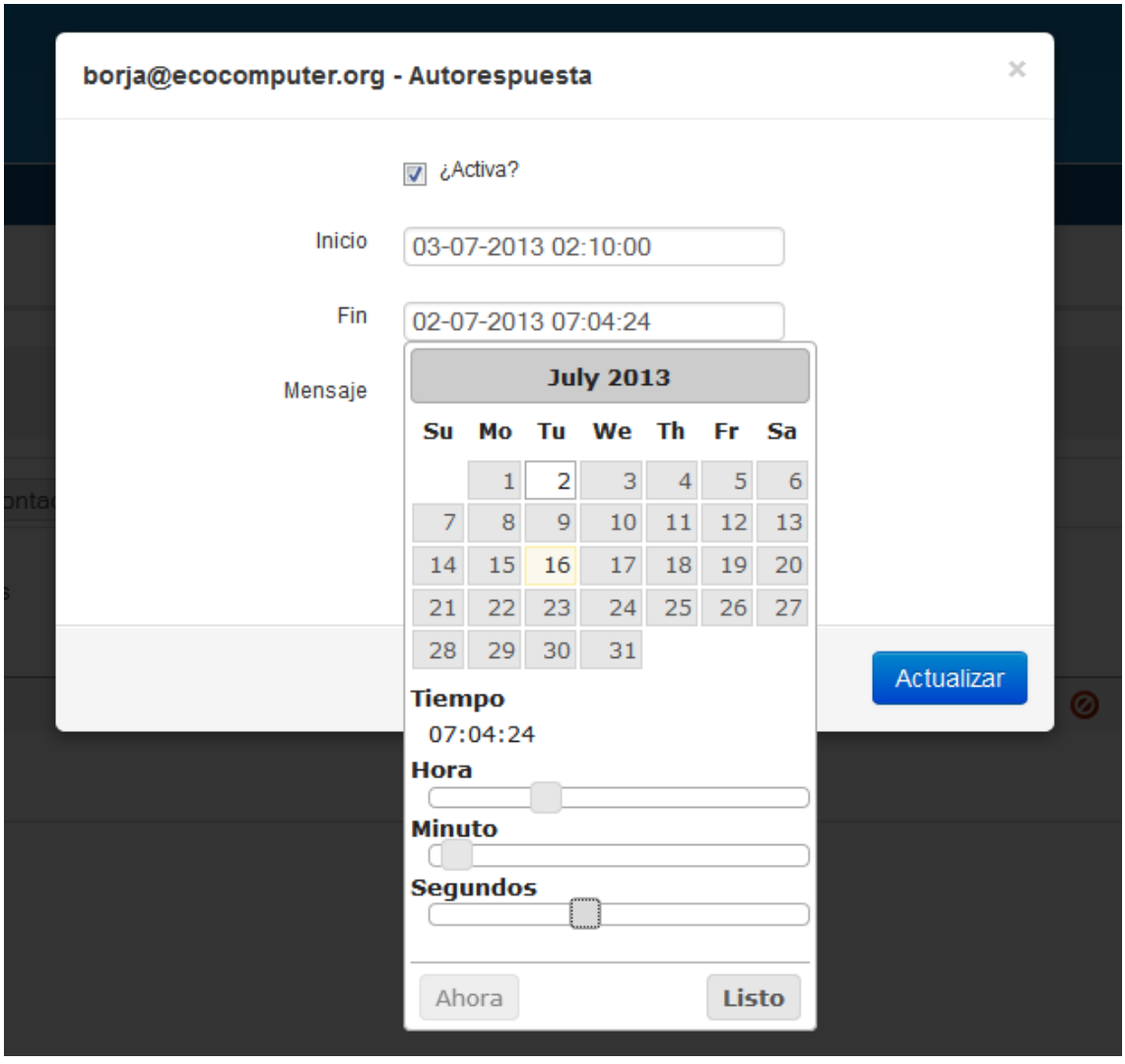
Manual usuario 9.11: Añadir buzón

9.3.6.2 *Añadir cuentas a un buzón*

Si pulsamos sobre el nombre del buzón en la lista de ver buzones que tenemos en la imagen anterior entramos dentro de sus detalles, donde se nos muestran todas las cuentas que existan para ese buzón.

Manual usuario 9.12: Cuentas de correo

Desde esta pestaña podemos insertar una cuenta de correo nueva o configurar la autorespuesta de una ya existente, para ello debemos pulsar sobre el icono de autorespuesta, lo que desplegará un dialogo como el siguiente.



Manual usuario 9.13: Activar autorespuesta

Con esto terminamos con una visión rápida sobre como utilizar la aplicación, dada su naturaleza, está más enfocada a los desarrolladores pues el principal punto es ofrecer una

base para la inclusión de nuevos módulos. Por lo que cualquiera que quiera utilizarla se le recomienda ir al manual del programador.

9.4 Manual del Programador

En este manual se describe cualquier aspecto que pueda ayudar a otros programadores a ampliar, modificar o entender aspectos de la construcción de nuestra aplicación. No se explica la arquitectura general de una aplicación struts2 o el funcionamiento de Spring, si no aquellos aspectos de quizá mayor dificultad de comprensión.

Como ya hemos ido viendo a lo largo de la documentación, este proyecto desarrolla una base común pero independiente: la intranet, una librería común y la base de datos, además de dos módulos que funcionan sobre ella. Esto se transforma en una serie de facilidades para el programador de los módulos, puesto que no tiene que gestionar algunos aspectos importantes. Concretamente:

- La Intranet se encarga de colocar al usuario en sesión; los Action de los módulos pueden recuperar la información del usuario implementando la interfaz SessionAware y utilizando el método `session.get("usuario")`.
- La clase Usuario y asociadas (Perfil, Cliente…) se encuentran en la librería Java ecoclases.jar, incluida dentro del directorio de librerías de Apache Tomcat.
- La librería también contiene lo necesario para gestionar el log o registro de la aplicación. Para ello basta con declarar la variable `Logger log = new Logger(this.getClass().getName(), session)`. Escribir entradas del registro se hace mediante los métodos `log.info()`, `log.error()` y `log.warning()`, recibiendo como parámetro un String con el mensaje a escribir en la base de datos.
- También se encarga de la comunicación con la base de datos, realizada mediante un *pool de conexiones* para gestionar mejor los recursos y reducir tiempos de espera innecesarios. La conexión con la base de datos se establece declarando `Connection con = ConnectionManager.getConnection()`.
- Para crear un nuevo elemento del modelo se recomienda seguir la misma jerarquía de clases e interfaces vista en apartados anteriores (ManagerService, DataService, DAO…). Recuérdese declarar la inyección de dependencias necesarias en el fichero applicationContext.xml.
- Algunos Action tienen tipos de retorno especiales. Tal es el caso, por ejemplo, de ObtenerDatosUsuarioAction, que se invoca tras una llamada AJAX desde añadirCliente.jsp y que retorna un objeto JSON para obtener nueva información sin necesidad de recargar toda la interfaz. Las definiciones de “result-types” pueden encontrarse en struts.xml.
- Las JavaServer Pages se componen mediante el uso de “tiles”, unas librerías Java que se encargan de crear la interfaz como si de un árbol se tratase, compartiendo atributos de padres a hijos para ahorrar código. Veamos esto en un ejemplo:
    - El fichero struts.xml llama a la ejecución de un Action cuyo resultado apunta a un “tiles” (por ejemplo, `<result type="tiles">AccesoUsuario.tiles</result>`).

- La definición de dicho resultado se encuentra en el fichero tiles.xml, que declara sus propios atributos y a su vez los hereda de su padre:

```
    <definition name="BaseLayout" template="/WEB-INF/tiles/Layout.jsp">
    </definition>

    <!-- Esta referencia solo se utiliza para la página de Login -->
    <definition name="Login.tiles" extends="BaseLayout">
        <put-attribute name="title" value="EcoComputer - Inicio de sesión" />
        <put-attribute name="logo" value="media/images/logo_gestor.png"/>
        <put-attribute name="content" value="/public/login.jsp" />
    </definition>

    <definition name="AccesoUsuario" extends="AccesoLayout">
        <put-attribute name="content" value="/WEB-
INF/pages/usersManagement/usuario.jsp"/>
    </definition>
    <definition name="AccesoCliente" extends="AccesoLayout">
        <put-attribute name="content" value="/WEB-
INF/pages/usersManagement/cliente.jsp"/>
    </definition>
    <definition name="CambioPassword" extends="AccesoLayout">
        <put-attribute name="content" value="/WEB-
INF/pages/usersManagement/changePassword.jsp"/>
    </definition>
```

Este ejemplo muestra cómo se pueden generar tres interfaces web distintas con elementos en común, en este caso AccesoUsuario, AccesoCliente y CambioPassword. Esto resulta muy útil a la hora de realizar cambios, pues muchos elementos utilizarán el mismo fichero para construirse.

- Finalmente, se ofrece un sistema completo de gestión de usuarios y permisos de lectura, siempre y cuando se utilicen los componentes de manera adecuada, a continuación se explica cómo se estructuran las aplicaciones desarrolladas hasta el momento y que paquetes se contemplan.
    - Paquete “LoginAccess”: Cualquier action situado dentro de este paquete requerirá que el usuario se encuentre identificado en la aplicación para su ejecución, define también algunos resultados globales para ser utilizados por nuevos desarrolladores.

```
<package name="LoginAccess" extends="struts-default">
                <result-types>
             <result-type name="tiles"
                 class="org.apache.struts2.views.tiles.TilesResult" />
        </result-types>

                <interceptors>
                        <interceptor name="LoginInterceptor"

       class="org.eco.mvc.impl.intranet.security.LoginInterceptor">
                        </interceptor>
                        <interceptor-stack name="securityLoginInterceptorStack">
                                <interceptor-ref name="LoginInterceptor" />
                                <interceptor-ref name="defaultStack" />
                        </interceptor-stack>
                </interceptors>

                <default-interceptor-ref name="securityLoginInterceptorStack" />

                <global-results>
                        <result name="login" type="redirect">../Intranet</result>
                        <result name="errorBaseDatos"
type="tiles">ErrorBaseDatos.tiles</result>
                </global-results>

                <action></action>
```

- Paquete "SecurityAccessControl": Dentro de este paquete se colocarán los actions que requieran un control de acceso especial, por ejemplo aquellos que requieran una carga de datos previa.

```
<package name="LoginAccess" extends="struts-default">
		<result-types>
	    <result-type name="tiles"
	        class="org.apache.struts2.views.tiles.TilesResult" />
	 </result-types>

		<interceptors>
			<interceptor name="LoginInterceptor"

	class="org.eco.mvc.impl.intranet.security.LoginInterceptor">
			</interceptor>
			<interceptor-stack name="securityLoginInterceptorStack">
				<interceptor-ref name="LoginInterceptor" />
				<interceptor-ref name="defaultStack" />
			</interceptor-stack>
		</interceptors>

		<default-interceptor-ref name="securityLoginInterceptorStack" />

		<global-results>
			<result name="login" type="redirect">../Intranet</result>
			<result name="errorBaseDatos"
type="tiles">ErrorBaseDatos.tiles</result>
		</global-results>

		<action></action>
	</package>
```

- Paquete "Object_management": Dentro de este paquete habrán de colocarse los actions que requieran al usuario tener permisos de lectura sobre los actions que en él se definen, dado que uno de los parámetros es id del módulo que se quiere evaluar, deberá crearse una definición de este paquete por cada módulo que quiera integrarse.

```
<package name="user_management" extends="SecurityAccessControl">
		<interceptors>
			<interceptor name="UserManagementAccessControl"
		class="org.eco.mvc.impl.intranet.security.ReadAccessInterceptor">
			<param name="idModulo">1</param>
		</interceptor>

			<interceptor-stack name="userManagementInterceptorStack">
				<interceptor-ref name="LoginInterceptor" />
				<interceptor-ref name="UserManagementAccessControl" />
				<interceptor-ref name="defaultStack" />
			</interceptor-stack>
		</interceptors>

		<default-interceptor-ref name="userManagementInterceptorStack" />

		<action></action>
	</package>
```

- Paquete "Object_write_management": Similar al paquete anterior, solo que en este caso se requerirá el permiso de escritura sobre el módulo al usuario.

```
<package name="user_write_management" extends="SecurityAccessControl">
		<interceptors>
			<interceptor name="UserWriteAccessControl"
		class="org.eco.mvc.impl.intranet.security.WriteAccessInterceptor">
			<param name="idModulo">1</param>
		</interceptor>

			<interceptor-stack name="userWriteInterceptorStack">
				<interceptor-ref name="LoginInterceptor" />
				<interceptor-ref name="UserWriteAccessControl" />
				<interceptor-ref name="defaultStack" />
			</interceptor-stack>
```

```
            </interceptors>

            <default-interceptor-ref name="userWriteInterceptorStack" />

            <action> </action>
        </package>
```

Capítulo 10. Conclusiones y Ampliaciones

10.1 Conclusiones

Aunque el objetivo principal que se planteó para este proyecto cambio durante su desarrollo, se puede concluir que se ha conseguido el nuevo objetivo principal del mismo, asentar una base sostenible y flexible para la Intranet de la empresa que haya podido combinar de manera satisfactoria todos los módulos desarrollados por los compañeros.

El realizar un proyecto para un cliente real fue otro de los aspectos que se persiguieron en la ejecución del mismo, siendo un paso lógico tras terminar la rama profesional del máster, realizar un proyecto para una empresa. Especificación de requisitos, reuniones de seguimiento, cambios imprevistos de funcionalidad… Todo lo relacionado con el trato con un cliente sucedió. Y aunque normalmente resultase más inconveniente que ventaja, se esperaba que esto sucediese para intentar hacer de este proyecto un caso lo más parecido posible a la realidad del mercado laboral actual.

Atarse a los deseos y necesidades de un cliente tiene también sus inconvenientes, principalmente centrados en no poder aplicar lo que se quiere y debe en ciertos aspectos del desarrollo. Por ejemplo, todo lo relacionado con el diseño de la interfaz de usuario y sus aspectos de usabilidad y accesibilidad fueron eliminados de las responsabilidades del programador, puesto que era uno de los requisitos del cliente. De no haber sido así los resultados podrían haber sido muy diferentes, para bien o para mal, pero al menos bajo los deseos del programador. Aunque, como ya se ha mencionado, con este proyecto se buscaba satisfacer las necesidades de un cliente y no las del desarrollador.

Posiblemente uno de los peores amigos del desarrollador sea su capacidad de Long Life Learning (Aprendizaje continuo) pues siempre se encuentra una manera mejor de afrontar los problemas y es decisión del desarrollador y seguir por el camino elegido inicialmente o comenzar de nuevo de una manera más acertada, durante el desarrollo de este proyecto esto llego a ser un gran inconveniente pues de él dependían otros proyectos dentro de la empresa, lo que hacía imposible tomar este tipo de decisiones sin influir en ellos. Por esto quizás el proyecto no cumpla totalmente el grado de satisfacción que se esperaba de él por parte del desarrollador.

En términos generales, es una satisfacción haber realizado un proyecto de esta envergadura, investigando tecnologías nuevas y utilizando otras que ya se consideraban atractivas. El trato con el cliente, la colaboración con otros equipos de trabajo, el interés personal en la tecnología utilizada y la capacidad para ver problemas reales que con certeza se harán visibles durante la vida del desarrollador hacen que de este proyecto se hayan aprendido no solo tecnologías si no aspectos muchos más intrínsecos al trabajo en empresa, los cuales de ninguna otra manera podían haberse obtenido.

Con todo lo anterior, se puede concluir que aunque el proyecto desarrollado no cumpla la calidad que se esperaba de el en un principio, la satisfacción general es amplia, pues se han adquirido muchos tipos de conocimientos durante su desarrollo, como ya se dijo antes, no solo vinculados a las tecnologías intrínsecas del mismo.

10.2 Ampliaciones

10.2.1 Usabilidad y accesibilidad

Una de las ampliaciones más claras sería su adaptación convirtiéndolo en una aplicación que cumpla con los estándares de usabilidad y accesibilidad esperados para un proyecto de este tipo.

10.2.2 Utilización de otra tecnología más acorde

Durante el desarrollo del proyecto, se tuvo la oportunidad de trabajar en otros que nada tenían que ver con esta aplicación. Sin embargo sí que permitieron un acercamiento mayor a tecnologías para el desarrollo web o el acceso a base de datos, que si bien se vieron durante el máster, no se aplicaron debido al común acuerdo al que se llegó con los demás proyectantes para realizar un proyecto lo más homogéneo posible.

10.2.2.1 *Cambios del framework de trabajo*

Se comenzó utilizando Struts pues era un framework que se estudió durante el máster y era una tecnología cuyos conocimientos eran compartidos por todos los compañeros. Sin embargo existe un framework que se está en gran medida sobreponiendo a Struts, estamos hablando de Spring.

Este framework ofrece muchas librerías y da soporte para números tipos de tecnologías, entre ellos JPA con las librerías de SpringData.

10.2.2.2 *Cambio de la tecnología de acceso a datos*

La utilización de alguna implementación de JPA hubiese resultado muy atractiva, pues el trabajo desarrollado se hubiese simplificado en gran medida gracias a las anotaciones que proporciona. Concretamente se pensó en Hibernate para ello, pues SpringData ofrece un gran soporte para ella.

10.2.3 Mayor soporte al servidor ISPConfig

Aunque se intentó durante el desarrollo del proyecto, se terminó ofreciendo un soporte bastante básico para la comunicación con el servidor ISPConfig, quedándose solo con algunos de los muchos elementos que este ofrece.

10.2.4 Aplicación para Android

Una de las ideas iniciales que se descartó fue el diseño de la aplicación bajo la plataforma móvil Android. Pese a que la aplicación fue diseñada para funcionar bajo cualquier tipo de navegador, no todos son capaces de realizar todas las funciones. Android no es una excepción.

Es por esto que podría resultar interesante realizar una versión simplificada de la aplicación para dicha plataforma, de manera que los usuarios pudiesen hacer uso de la misma a través de su dispositivo móvil y no tener que recurrir a un equipo de escritorio.

Capítulo 11. **Presupuesto**

Debido a que la realización del trabajo se ha llevado a cabo para una empresa, y que por lo tanto tienen su propia manera de realizar presupuestos, correspondiente a multiplicar por 39€ las horas de desarrollo del mismo, derivando de ellas beneficios, alquiler de equipos, pago de facturas,… Se plantea un presupuesto tal y como lo haría dicha empresa para poder realizar una comparativa con el presupuesto desarrollado según los conocimientos adquiridos durante la carrera y el máster y comprobar de esta manera si dicho presupuesto se acercaría a la realidad.

11.1 Presupuesto de la empresa

Ítem	Concepto	Cantidad	Precio unitario	TOTAL
0	Horas cliente	17	39,00 €	663,00 €
1	Horas diseñador gráfico	10	39,00 €	390,00 €
2	Horas programador	490	39,00 €	19.110,00 €
			Subtotal	20.163,00 €
			IVA (21%)	4.013,10 €
			TOTAL	**24.176,10 €**

Como vemos en este presupuesto se cuentan tal y como se dijo las horas dedicadas por todas las partes al proyecto, incluyendo horas de reuniones puesto que el cliente dedicará su tiempo al mismo y no a otros quehaceres.

11.2 Presupuesto desarrollado por el alumno

Ítem	Concepto	Cantidad	Precio unitario	TOTAL
1	**Desarrollo de la aplicación**	**Horas**	**Precio/Hora**	
1.1	Análisis	45	15,00 €	675,00 €
1.2	Diseño	25	15,00 €	375,00 €
1.3	Implementación	245	15,00 €	3.675,00 €
1.4	Pruebas	35	15,00 €	525,00 €
2	**Formación**	**Horas**	**Precio/Hora**	
2.1	Manuales	30	15,00 €	450,00 €
2.2	Formación de usuarios	10	15,00 €	150,00 €
2.3	Mantenimiento	50	15,00 €	750,00 €
3	**Adquisiciones**			
3.1	**Hardware**	**Unidad**	**Precio/Unidad**	
3.1.1	Servidor	1	0,00 €	0,00 €
3.2	**Software**	**Unidad**	**Precio/Unidad**	
3.2.1	Microsoft SQLServer 2012	1	3.500,00 €	3.500,00 €
3.2.2	Servidor Apache Tomcat	1	0,00 €	0,00 €
3.2.3	Eclipse	1	0,00 €	0,00 €
4	**Alquileres**	**Meses**	**Precio/Mes**	
4.1	Ordenador portátil	7	20,00 €	140,00 €
4.2	Oficina	7	200,00 €	1.400,00 €
4.3	Luz	7	20,00 €	140,00 €
			Subtotal	11.780,00 €
			IVA (21%)	2.473,80 €
			Beneficio (40%)	4.712,00 €
			Total	**18.965,80 €**

Obviamente esto es un presupuesto interno, pues no se le pueden presentar al cliente los beneficios esperados del proyecto como tales, sino que se le incluirán en las horas de desarrollo.

Como se puede observar, varía en bastante del ofrecido por la empresa, pero esto era algo claro pues en el caso de la empresa además del beneficio han de salir varios sueldos más que no se contemplan aquí, además del mantenimiento de una oficina más grande, facturas eléctricas, etc.

Capítulo 12. Referencias Bibliográficas

12.1 Referencias en Internet

[Oracle] "Java™ Platform, Standard Edition 7". http://docs.oracle.com/javase/7/docs/api/. 2013.

[W3C] "HTML 4.01 Specification". http://www.w3.org/TR/html401/. 1999.

[W3C] "Cascading Style Sheets home page". http://www.w3.org/Style/CSS/. 2013.

[Oracle] "JavaServer Pages API Documentation".

http://docs.oracle.com/cd/E17802_01/products/products/jsp/2.1/docs/jsp-2_1-pfd2/. 2005.

[Microsoft MSDN] "Transact-SQL Reference (Database Engine)".

http://msdn.microsoft.com/en-us/library/bb510741.aspx. 2007.

[The Apache Software Foundation] "Apache Tomcat 6.0 documentation".

http://tomcat.apache.org/tomcat-6.0-doc/index.html. 2013.

[The Apache Software Foundation] "Apache Struts 2 Documentation".

http://struts.apache.org/release/2.2.x/docs/home.html. 2011.

[The Apache Software Foundation] "Tiles API".

http://tiles.apache.org/framework/apidocs/index.html. 2012

[jQuery] "jQuery API Documentation". http://api.jquery.com/. 2013.

[jQueryUI] "jQuery UI 1.9 API Documentation". http://api.jqueryui.com/1.9/. 2013.

[BootStrap] "Bootstrap v2.3.2". http://twitter.github.io/bootstrap/index.html. 2013.

12.2 Tutoriales y guías varias

[RoseIndia] "Struts 2 Tutorial". http://www.roseindia.net/struts/struts2/. 2008.

[Mkyong] "Struts 2 Tutorial". http://www.mkyong.com/tutorials/struts-2-tutorials/. 2010.

[DZone] "Struts 2 Tutorials". http://www.dzone.com/tutorials/java/struts-2. 2012.

[viralpatel] "Struts 2 Tutorials, Tips & Tricks". http://viralpatel.net/blogs/category/j2ee/struts-2/. 2009.

12.3 Foros de ayuda

[stackoverflow] "Stack Overflow". http://stackoverflow.com/. 2013.

[CodeRanch] "Java Forums at the Big Moose Saloon". http://www.coderanch.com/forums. 2013.

Capítulo 13. Apéndices

13.1 Contenido Entregado en el CD-ROM

Directorio	Contenido
./ Directorio raíz del CD	Contiene un fichero leeme.txt explicando toda esta estructura.
./intranet	Contiene dentro los trés módulos desarrollados para esta aplicación.
./intranet/intranet	Contiene todo el directorio de proyecto del módulo intranet.
./intranet/gestionUsuarios	Contiene todo el directorio de proyecto del módulo usuarios.
./intranet/gestionDominios	Contiene todo el directorio de proyecto del módulo dominios.
./documentacion	Contiene toda la documentación asociada al proyecto.
./documentacion/img	Directorio que contiene las imágenes utilizadas en la documentación.
./documentacion/uml	Directorio que contiene todos los archivos de diagramas utilizados en el desarrollo del proyecto.
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto (lógicamente sólo las que sean distribuibles).
./herram/desarrollo	Ficheros de instalación de las herramientas utilizadas en el desarrollo
./herram/explotacion	BD, servidor Web y herramientas en general.