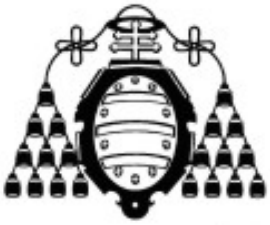
UNIVERSIDAD DE OVIEDO



MÁSTER EN INGENIERÍA WEB

CENTRO INTERNACIONAL DE POSTGRADO

TRABAJO FIN DE MÁSTER

Comparativa de los métodos de desarrollo de ampliaciones en gestores de contenidos web

AUTOR: Adrián Menéndez Monroy
DIRECTORA: B. Cristina Pelayo García-Bustelo

Agradecimientos

A mi tutora, Cristina Pelayo, por la paciencia que ha tenido y por las buenas directrices y ánimos dados durante todo el desarrollo del proyecto.

A Adele Robots por proporcionarme el marco ideal para el desarrollo.

Resumen

En este proyecto se ha desarrollado una comparativa entre los tres Gestores de Contenido Web (CMS) más utilizados en la actualidad, WordPress, Joomla y Drupal y una métrica para dotarla de un carácter más tangible. El objetivo principal es permitir y/o ayudar a desarrolladores externos a llevar a cabo ampliaciones de la funcionalidad de los mismos. En base a la comparativa se desarrollará unas métricas que permitan seleccionar los CMS que se ajusten más a las necesidades del desarrollador y de los administradores del CMS y del contexto, y que permita tanto a usuarios como a desarrolladores basar esta decisión en datos empíricos.

La comparativa no se ha llevado a cabo a nivel de usabilidad del framework, o de publicación de contenido, si no que se han desarrollado plugins, módulos o ampliaciones de la funcionalidad del gestor para tratar de estudiar las características que los diferencian y las herramientas que proporcionan a los desarrolladores para llevar a cabo sus proyectos.

Para que el desarrollo de los módulos se asemeje lo máximo posible a una situación de desarrollo real por parte de una compañía, se ha implementado la integración de los asistentes virtuales de la empresa Adele Robots con estos tres CMS.

Palabras Clave

Gestores de Contenidos Web, Asistentes Virtuales, PHP, Publicación de Contenido en la Web, Integración, Métricas.

Abstract

This project has developed a comparision between the most used Content Managers (Wordpress, Joomla and Drupal) and also a metric to make this comparison more tangible. The main objetive is to allow or help third-party developers to carry out functionality expansions. Based on the comparative, metrics that allows devevelopers and administrators to choice the best option for they needs and their project context will be develop.

The comparision has not been carry out at usability level or content publishing of the framework, instead functionality expansions of the CMS has been developed trying to study the characteristics that differentiate them and the developer tools they provide.

Trying to make the development as real as posible, the expansions were made to interatuate with Adele Robots Virtual assistant.

Keywords

Web Content Managers, Virtual Assistants, PHP, Web Content Publishing, Integration, Metrics.

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

La motivación que tiene el desarrollo de este proyecto, es poder aplicar parte de los conocimientos adquiridos en el Máster de Ingeniería Web de la Universidad de Oviedo a un entorno de trabajo lo más parecido posible a uno real.

Dado que mi formación anterior como Ingeniero Técnico Industrial, especializado en Electrónica Industrial, está muy lejos del ámbito del Máster en Ingeniería Web, he sufrido en mi propia piel el enorme esfuerzo que supone para alguien sin los conocimientos técnicos necesarios, iniciarse en el desarrollo web. Puesto que todos los conocimientos que poseo sobre desarrollo web los he adquirido en este Máster y, desgraciadamente, no todo el mundo tiene acceso a él, me surgió la idea de trabajar en un proyecto que me permitiese hacer de intermediario en algo tan complejo como es el desarrollo web y dos tipos de usuario. Un usuario estándar de la web, sin conocimientos de desarrollo, y desarrolladores que quieran/necesiten comenzar a desarrollar.

Fue gracias a la posibilidad de realizar las prácticas del Máster en Adele Robots cuando encontré el marco ideal para este proyecto.

Adele Robots es una empresa dedicada en gran parte al desarrollo web, y uno de los productos que oferta son los llamados asistentes virtuales.

Uno de los principales problemas que se encuentra un usuario a la hora de utilizar uno de estos asistentes en su página web, es la manera de integrarlo, dado que requiere, al menos, unos pequeños conocimientos de programación.

Es en este punto, donde este proyecto cobra sentido.

¿Cuál es la manera más sencilla para un usuario estándar de Internet de comenzar a publicar contenido?

No hay nada como un Gestor de Contenidos.

¿Cuál es la manera de facilitar a un usuario el acceso a un producto como pueden ser los asistentes virtuales de Adele Robots?

Integrar dichos asistentes con los principales Gestores de Contenidos de manera que resulte prácticamente transparente para el usuario.

Además, aprovechando todo lo aprendido en mi paso por el Máster de Ingeniería Web, se pretende hacer una comparativa entre las herramientas que nos proporcionan los principales Gestores de Contenidos para realizar integraciones como la mencionada y obtener unas

métricas que ayuden a hacer más tangible algo tan abstracto como la preferencia por uno u otro método de desarrollo.

De esta manera, se pretende que las métricas obtenidas al final de este Trabajo Final de Máster, ayuden a un usuario a elegir que Gestor de Contenidos utilizar si se pretende aumentar su funcionalidad, y a un desarrollador y/o compañía interesada en el desarrollo, que CMS permitirá llevar a cabo el desarrollo con los menores inconvenientes y en el menor tiempo posible.

Capítulo 2. Aspectos Teóricos

Para entender la justificación de este trabajo final de máster y poder seguir su desarrollo es necesario tener unos conceptos básicos previos del funcionamiento de Internet. En un breve repaso a los conceptos básicos necesarios, se hará referencia únicamente a las partes que intervienen directamente en la navegación, sin entrar en detalle en las tecnologías utilizas para la comunicación entre ellas, ya que no aportaría nada relevante para el entendimiento del proyecto.

2.1 Conceptos Previos

2.1.1 Navegador Web

El navegador web es el software que permite a un usuario el acceso a internet. Realiza peticiones a un servidor, recibe la respuesta de este, decodifica la información y con ella construye la página web que se muestra al usuario.

Los más utilizados actualmente son Google Chrome, Mozilla Firefox, Internet Explorer, Safari y Opera.

Figura 1. Principales Navegadores Web.

2.1.2 Servidor

Un servidor se trata de una aplicación que procesa las peticiones de un cliente y le devuelve una respuesta. En el caso que nos ocupa, y para simplificar un poco el funcionamiento, un cliente le hace una petición de información y el servidor responde en un lenguaje que el navegador pueda interpretar.

2.1.3 Página Web

Podemos entender una página web como un documento que contiene información, ya sea texto, video, imágenes, etc., adaptado a Internet. Generalmente, y en su forma más sencilla, está formado por código HTML y una hoja de estilo, CSS.

Figura 2. Contenido Web.

2.1.4 HTML

Es un lenguaje estructurado que permite, mediante etiquetas, organizar la información que se envía desde un servidor a un cliente. Gracias a esta estructura, y a las etiquetas utilizadas, el cliente, generalmente un navegador web, es capaz de procesar dicha información y mostrársela al usuario de manera legible.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Adrián Menéndez - Curriculum Vitae</title>
        <link rel="stylesheet" type="text/css" href="css/estilo.css">
        <link href='http://fonts.googleapis.com/css?family=Rokkitt:400,700|Lato:400,300' rel='stylesheet' type='
        text/css'>
    </head>
    <body>
        <header class="vcard">
            <img src="img/foto.jpg" alt="fotografía del autor"/>
            <section>
                <h1><span class="given-name">Adrián</span> <span class="family-name">Menéndez</span></h1>
                <h2 class="title">Desarrollador Web</h2>
            </section>
            <section>
                <ul>
                    <li>e: <a class="email" href="mailto:amenendezm@gmail.com" target="_blank">amenendezm@gmail.com<
                    /a></li>
                    <li>w: <a class="url" href="http://www.adrianmenendez.com">www.adrianmenendez.com</a></li>
                    <li class="tel">m: 647821603</li>
                </ul>
            </section>
        </header>
        <section>
```

Figura 3. Ejemplo de código HTML.

2.1.5 CSS

Habitualmente, asociado al código HTML de una página web, se encuentra una hoja de estilo, actualmente escrita en código CSS. La hoja de estilo, como su propio nombre indica, "informa" al navegador el aspecto con el que debe presentar la información recibida en el HTML al usuario.

```css
html {
    background: black;
    font-family: 'Lato', helvetica, arial, sans-serif;
    font-size: 1em;
    color: #222;
}

body {
    max-width: 800px;
    margin: 25px auto;
    background: #f3f3f3;
    padding: 50px;
}

p {
    font-size: 1em;
    line-height: 1.4em;
    color: #444;
}
```

Figura 4. Ejemplo de código CSS.

2.1.6 XML

El enorme intercambio de información que se produce en internet, no siempre se produce entre un navegador web y un servidor. Si no que, en muchas ocasiones, es necesario que diferentes máquinas de diferentes plataformas intercambien información entre sí. Para intentar que diferentes plataformas puedan "entenderse" entre sí, surge, en los años 70, un lenguaje de marcado para el almacenamiento de información llamado GML (Generalized Markup Language), del que evolucionaría el actual XML.

```xml
<answernumbering>abc</answernumbering>
<correctfeedback format="html">
  <text></text>
</correctfeedback>
<partiallycorrectfeedback format="html">
  <text></text>
</partiallycorrectfeedback>
<incorrectfeedback format="html">
  <text></text>
</incorrectfeedback>
<answer fraction="-25" format="html">
  <text><![CDATA[<p>Que todas las etiquetas que se abren, se cierran</p>]]></text>
  <feedback format="html">
    <text><![CDATA[<p>Ésa es una condición para que esté bien formado.</p>]]></text>
  </feedback>
</answer>
```

Figura 5. Ejemplo de código XML.

2.1.7 JavaScript

JavaScript es un lenguaje utilizado generalmente en el lado del cliente (navegador web) para generar páginas web con una interfaz de usuario más avanzada y dinámica. La función más utilizada para el JavaScript es la modificación o generación de HTML en el cliente bajo demanda. Es decir, permite modificar elementos del documento HTML, crearlos o eliminarlos, sin necesidad de una interacción con el servidor, todo el procesamiento se lleva a cabo en el cliente.

```
function encimaBoton() {
  $("#boton").attr("src","./imagenes/boton-rojo.png");
}
function fueraBoton() {
    $("#boton").attr("src","./imagenes/boton-azul.png");
}

$("#boton").mouseover(encimaBoton).mouseout(fueraBoton);

$(document).ready(main);
```

Figura 6. Ejemplo de código JavaScript.

A medida que la web fue evolucionando, con cada vez más usuarios y más información, se hizo necesario tener una manera fluida de publicar toda esta información. La manera tradicional de publicarla, generando un código HTML y asociándole una hoja de estilo, requiere tiempo, conocimiento de ambos lenguajes y se hace pesado para el cada vez más creciente tráfico en Internet. Es aquí donde surgen los Gestores de Contenidos.

2.2 Gestores de Contenidos

Los Gestores de Contenidos surgieron a finales de los 90 como respuesta a la creciente necesidad de publicación de contenido en la web. Concretamente, el sitio web CNET, enfocado a las noticias de ámbito tecnológico, a través de la creación de la compañía Vignette, se convirtió en pionero de los gestores de contenido comerciales.

Un Gestor de Contenidos, o conocido por sus siglas en Inglés CMS, es un framework para la publicación y administración de contenido, principalmente en la web. Dota al usuario de herramientas para publicar información de manera transparente a la tecnología utilizada. Permite, a un usuario estándar de la web, dejar de ser un mero consumidor de información para convertirse en una fuente de divulgación.

Los CMS en lugar de funcionar con contenido estático, como era habitual, consisten en una interfaz que controla las bases de datos en las que se aloja el contenido. Gracias a que este almacenamiento se produce de manera independiente para el contenido y el diseño, es muy sencillo para el usuario generar contenido de manera independiente al estilo del sitio, y a su vez, cambiar el estilo de toda la web de manera automática.

En el campo concreto de este Trabajo Final de Máster, se abordará la versión más extendida para este tipo de frameworks, los Gestores de Contenidos Web o Web CMS.

2.2.1 Gestores de Contenidos Web

Se trata de gestores de contenidos orientados a la publicación en la web. Permiten a un usuario acceder al contenido de un servidor de manera mucho más fluida que una página web estándar, ya que genera el contenido bajo demanda.

Reduce en gran medida el coste de mantenimiento y administración de un sitio web, ya que no es necesario maquetar cada HTML, si no que a través de un editor integrado en la interfaz web de la aplicación, se puede añadir contenido a la base de datos, y esta se encarga de insertarlo en el conjunto del sitio web con el estilo predefinido.

Según las estadísticas más recientes, la cuota de "mercado" de los gestores de contenidos web está repartida en su mayor parte por WordPress, Joomla y Drupal, por lo que serán los CMS utilizados en este proyecto.

En la gráfica que se muestra a continuación puede verse un gráfico de uso de CMS en 2014.

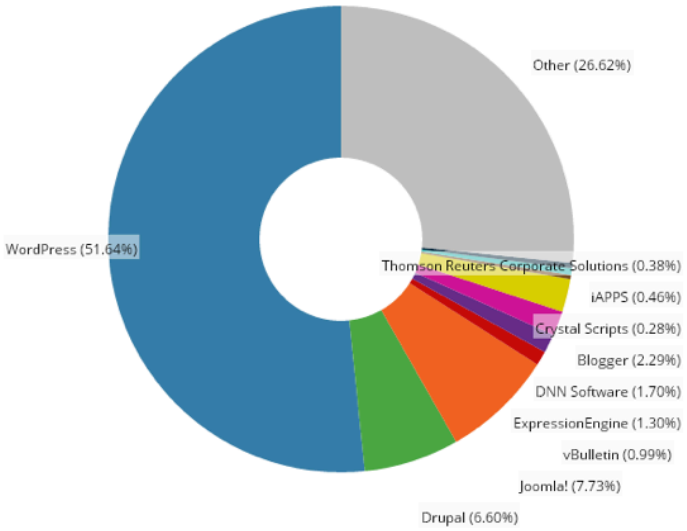


Figura 7. Comparativa de uso de navegadores.

2.2.1.1 Joomla

Figura 8. Joomla.

Joomla es el segundo CMS más utilizado en la actualidad, solo por detrás de WordPress. Su nombre proviene de la palabra swahili “jumla”, que significa "todos juntos" o "como un único todo", que permite hacerse una idea de la manera en que está desarrollado este Gestor de Contenidos.

Está desarrollado en PHP, y como se explicó en la introducción de esta memoria, requiere una base de datos y un servidor para su funcionamiento. Generalmente se utiliza MySQL como base de datos, aunque MariaDB, al ser una base de datos de código libre y completamente compatible con MySQL, también es una de las favoritas para servir de almacenamiento de este Gestor de Contenidos. Del mismo modo, el servidor HTTP de Apache es el más extendido.

Una de las mayores ventajas que tiene Joomla como CMS, es que la funcionalidad base que incluye en su Core, puede ser fácilmente ampliada por medio de extensiones.

2.2.1.1.1 Extensiones

2.2.1.1.1.1 Lenguajes

Gracias a que Joomla es multi-idioma de manera nativa, mediante archivos de traducción resulta muy sencillo cambiar el idioma del sitio web completo.

2.2.1.1.1.2 Plantillas

Las plantillas en Joomla, a diferencia de otros CMS, se configuran en único archivo. Podríamos hacer una similitud entre una plantilla de Joomla y una página HTML única, lo que simplifica en gran medida la labor del diseñador.

2.2.1.1.1.3 Componentes

Es el tipo de extensión más complejo que posee Joomla, pero también el más potente. Están ideados para presentar el contenido principal de una página.

Los componentes gestionan información, funciones y datos, y en general pueden ser utilizados para ejecutar cualquier funcionalidad que no recaiga dentro de las funcionalidades básicas del CMS.

2.2.1.1.1.4 Módulos

Se trata de extensiones más ligeras y flexibles para la presentación de contenidos que un componente. Sus usos más habituales son proporcionar una ventana de salida a los componentes o contener pequeñas aplicaciones de terceros, por ejemplo una aplicación de previsión meteorológica o un canal de noticias externas.

Se les asigna un posición en la plantilla de las llamadas “posiciones de módulo", por ejemplo "left" o "right" en un diseño clásico de tres columnas.

2.2.1.1.1.5 Plugins

Es, quizás, el tipo de extensión más avanzada que posee Joomla. Permite configurar funcionalidades dentro del CMS asociadas a disparos de eventos. Según la documentación oficial del CMS, es la forma recomendada de conectar con fuentes externas.

2.2.1.2 Drupal

Figura 9. Drupal

Drupal es el tercer gestor de contenidos más utilizado en la actualidad, por detrás de WordPress y Joomla. Al igual que los anteriores posee licencia de software libre, está escrito en PHP y es compatible con MySQL o MariaDB.

Tanto en el desarrollo del propio CMS, como de extensiones existe una comunidad de usuarios muy activa, lo que lo convierte en un gestor tremendamente flexible.

Como punto negativo, está más orientado a desarrolladores experimentados que a usuarios finales, por lo que en el momento de su instalación no incorpora demasiada funcionalidad ni contenido de ejemplo, es el desarrollador el encargado de su configuración. Se podría hacer referencia a Drupal como un Gestor de Contenido “*desarrollado por y para desarrolladores*”.

2.2.1.2.1 Extensiones

2.2.1.2.1.1 Módulos

Los módulos son el tipo de extensión utilizado en Drupal para ampliar las funcionalidades del Gestor de Contenidos. Se distinguen en dos grandes grupos, los módulos que Drupal incorpora por defecto en la instalación, llamados "Core Módules", y los módulos desarrollados por terceros, como es el caso del que nos ocupa, llamados "Contributed Módules".

2.2.1.2.1.2 Temas

Al igual que en Joomla las Plantillas, los Temas en Drupal son los encargados de la configuración estética del sitio.

El uso de esta característica hace el diseño de un CMS tremendamente configurable, Drupal permite incluso, establecer un tema diferente para el administrador del sitio al del resto de usuarios.

2.2.1.3 WordPress

Figura 10. WordPress.

WordPress es, sin duda, el CMS más utilizado en la actualidad. Su éxito se basa en su enorme comunidad de desarrolladores, tanto desarrollando su Core como ampliándolo mediante extensiones o temas. Está desarrollado en PHP y enfocado a entornos de MySQL y Apache.

2.2.1.3.1 Extensiones

2.2.1.3.1.1 Temas

Es la manera en la que WordPress gestiona la presentación del contenido. En apartados anteriores de esta memoria, se hacía referencia a la facilidad con la que un usuario puede

cambiar completamente el aspecto y la presentación de su sitio web de manera casi automática, pues bien, en WordPress, esto es gracias a los Temas.

Al hacer referencia a que con un tema se cambia no solo el aspecto, si no también la presentación, es porque un tema no es solo una "piel" para el sitio web, si no que un Tema puede incluir, además de ficheros CSS para dar estilo al sitio, archivos de código PHP que generan páginas personalizadas.

No se ahondará más en esta funcionalidad de WordPress, ya que no es importante para el desarrollo del proyecto, y abordarla en profundidad sería demasiado extenso.

2.2.1.3.1.2 Plugins

Son una de las características por las que los CMS se han hecho tan populares.

El Core de WordPress está diseñado para ser ligero y maximizar la flexibilidad del gestor de contenidos. Debido a esto, es posible que las funcionalidades por defecto sean insuficientes para algunos usuarios finales. Gracias a los plugins, el usuario puede ampliar la funcionalidad de su sitio web de forma personalizada.

Según la documentación de WordPress, un plugin "*es un programa, o un set de una o más funciones, escritas en el lenguaje de scripting PHP, que añade características específicas o servicios a un sitio WordPress, el cual puede ser integrado sin problemas con el sitio web utilizando los métodos y puntos de acceso que proporciona la Interfaz de Programación de Aplicaciones de WordPress (API)".*

Gracias a esta API que proporciona el CMS, resulta sencillo acceder a funcionalidades del Core de WordPress que pueden ser utilizados como interfaz de acceso a las nuevas funcionalidades desarrolladas.

2.3 Asistentes Virtuales

En los apartados que se desarrollan a continuación se explica, de manera genérica, que es un Asistente Virtual y que aspectos lo conforman.

2.3.1 Introducción

Un asistente virtual es un personaje multimedia capaz de interactuar, al menos de forma básica, con un usuario. Esta interacción puede ser por medio de voz o de texto escrito, y por medio de ella, el asistente puede dar información u ofrecer un servicio.

Los asistentes virtuales tienen dos componentes principales y bien diferenciados, pero que trabajan estrechamente, la interfaz de usuario y la base de conocimiento.

2.3.2 Base de conocimiento

Para que un asistente virtual sea capaz de interactuar con un usuario, debe poseer los conocimientos necesarios para ello, es decir, debe saber qué y cómo responder ante una interacción. Por ejemplo, si se considera un asistente virtual con el que se puede interactuar por medio de un cuadro de diálogo, este debe ser capaz de "entender" lo que el usuario está escribiendo y "generar" una respuesta.

En la práctica, esto no es 100% así, si no que en la base de conocimiento del asistente, han de estar reflejadas todas las posibles entradas del usuario y precargadas una serie de respuestas asociadas a ellas. Dado que la interacción con los asistentes intenta asemejarse los más posible a la interacción con una persona real, y debido a la complejidad y ambigüedad del lenguaje natural, el procesamiento de este tipo de lenguaje es un campo extremadamente complejo.

Para solventar este problema surge el lenguaje de marcado AIML, *Artificial Intelligence Mark-up Language*. Se trata de un lenguaje de programación basado en XML que permite, mediante el marcado de la información y la configuración con etiquetas, almacenar la información que el asistente virtual necesita, y lo que es más importante, relacionarla entre sí, de manera que el asistente sea capaz de simular una conversación más o menos espontánea.

2.3.3 Interfaz de usuario

Si se continúa con el ejemplo anterior de un usuario interactuando con un asistente virtual mediante un cuadro de diálogo, en este punto tenemos el software del asistente virtual ligado a una base de conocimiento. Para poder completar el asistente y que un usuario pueda

comunicarse con él, es necesario una interfaz de usuario. Hay asistentes virtuales basados en cuadros de chat, con personajes virtuales...

La interfaz de usuario debe encargarse de proporcionar un método de entrada para que el usuario envíe información al asistente y un método de salida, a través del cual el asistente pueda enviar información al usuario.

2.3.4 Asistente virtual Fiona

En el caso concreto de este proyecto, la integración se realiza con Fiona, un asistente virtual con una interfaz de usuario compuesta por un cuadro de diálogo y un personaje tridimensional.

El método de entrada de información es un cuadro de texto, en el que el usuario introduce la información o la petición que desea enviarle al asistente y posee dos métodos de salida, uno mediante audio y otro mediante texto impreso en la ventana del asistente que funcionan simultáneamente. El personaje tridimensional y la salida de audio trabajan al unísono para hacer la experiencia de usuario lo más real posible. Además, como parte del modelo de negocio de la compañía, el personaje posee determinados parámetros configurables.

Con todas las piezas del puzzle ya definidas, ya se puede conocer a Fiona:

Figura 5.1 Fiona

Fiona es configurable desde un sitio web desarrollado por Adele Robots:

www.theclientrelationsfactory.com

En él se pueden personalizar los aspectos del asistente virtual mencionados en los apartados anteriores.

Capítulo 3. Planificación del Proyecto

3.1 Resumen de la planificación

Como puede observarse más detalladamente en el fichero de Microsoft Project Adjunto, el proyecto se divide en 5 apartados principales:

3.1.1 Tramitación inicial

Su cometido principal es preparar los trámites que aseguren la validez de la idea original para este Trabajo Final del Máster en Ingeniería Web de la Universidad de Oviedo.

3.1.2 Anteproyecto

Redacción y planificación de la documentación a presentar como anteproyecto del Trabajo Final de Máster.

En dicho anteproyecto se incluye la planificación inicial del Trabajo Final de Máster, cuyo listado de tareas puede observarse en el apartado 3.2.

3.1.3 Desarrollo

Es la fase más crítica, engloba las tareas orientadas al estudio y desarrollo del proyecto.

Se encuentra dividida en 2 subtareas principales:

3.1.3.1 Trabajo en la empresa

Como se ha mencionado anteriormente, parte del proyecto se desarrolla durante el tiempo de realización de las prácticas externas del Máster.

3.1.3.2 Trabajo autónomo

Engloba las tareas del proyecto realizadas una vez finalizado el periodo en Adele. Es la fase del proyecto que más modificaciones ha sufrido a lo largo de su desarrollo.

Debido a la incorporación del desarrollador del proyecto a la plantilla de la compañía ASAC Comunicaciones, el número de horas diarias que puede dedicar al proyecto se ha visto mermada. Por ello, se ha tenido que reorganizar la planificación, contemplando esta reducción del número de horas, lo que se ha traducido en un aumento de la duración del proyecto.

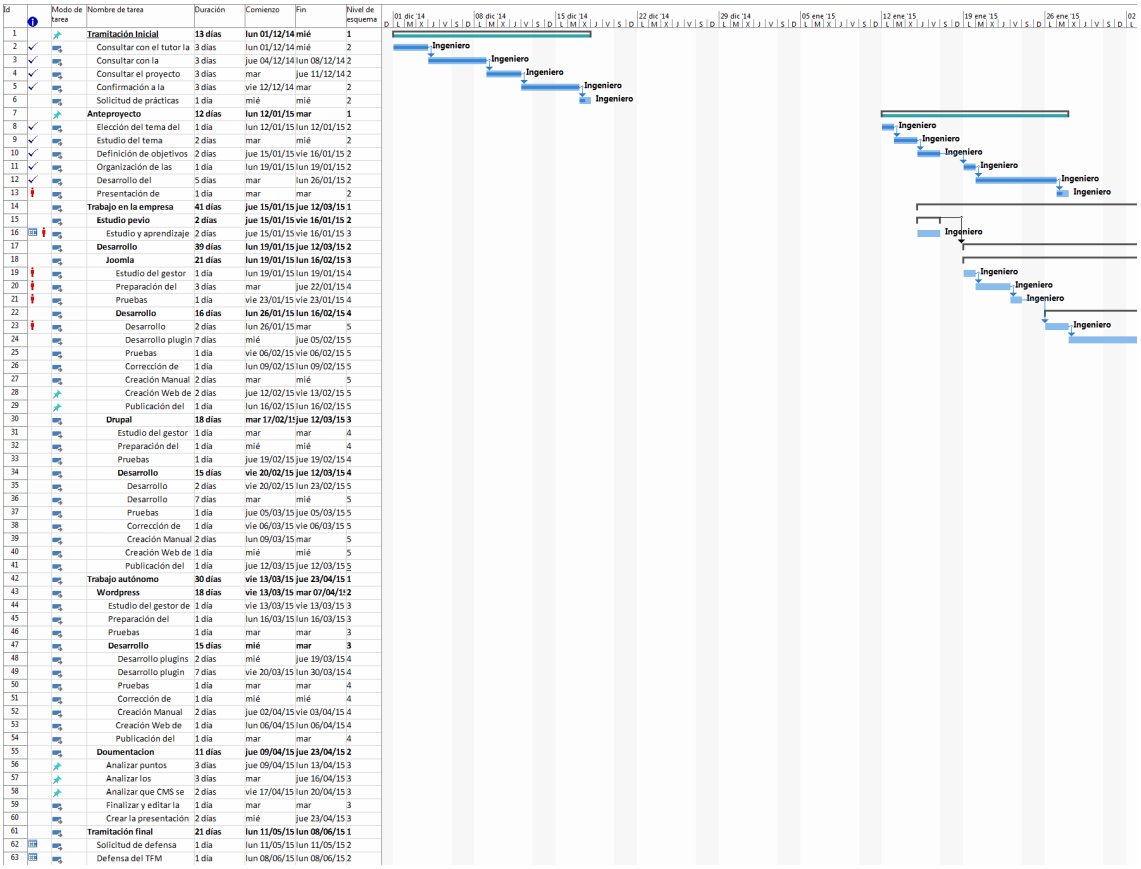
3.1.4 Documentación

Apartado fundamental para arreglar y matizar la documentación hecha durante el desarrollo. A partir de la información recopilada durante el desarrollo, durante las tareas de documentación se han desarrollado tanto la comparativa como la métrica.

3.1.5 Tramitación final

En este apartado se engloban las tareas que marcan la finalización del proyecto, como son la solicitud y la defensa del proyecto.

3.2 Planificación inicial



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Nivel de esquema
1		**Tramitación Inicial**	**13 días**	**lun 01/12/14**	**mié**	**1**
2		Consultar con el tutor la	3 días	lun 01/12/14	mié	2
3		Consultar con la	3 días	jue 04/12/14	lun 08/12/14	2
4		Consultar el proyecto	3 días	mar	jue 11/12/14	2
5		Confirmación a la	3 días	vie 12/12/14	mar	2
6		Solicitud de prácticas	1 día	mié	mié	2
7		**Anteproyecto**	**12 días**	**lun 12/01/15**	**mar**	**1**
8		Elección del tema del	1 día	lun 12/01/15	lun 12/01/15	2
9		Estudio del tema	2 días	mar	mié	2
10		Definición de objetivos	2 días	jue 15/01/15	vie 16/01/15	2
11		Organización de las	1 día	lun 19/01/15	lun 19/01/15	2
12		Desarrollo del	5 días	mar	lun 26/01/15	2
13		Presentación de	1 día	mar	mar	2
14		**Trabajo en la empresa**	**41 días**	**jue 15/01/15**	**jue 12/03/15**	**1**
15		**Estudio pevio**	**2 días**	**jue 15/01/15**	**vie 16/01/15**	**2**
16		Estudio y aprendizaje	2 días	jue 15/01/15	vie 16/01/15	3
17		**Desarrollo**	**39 días**	**lun 19/01/15**	**jue 12/03/15**	**2**
18		**Joomla**	**21 días**	**lun 19/01/15**	**lun 16/02/15**	**3**
19		Estudio del gestor	1 día	lun 19/01/15	lun 19/01/15	4
20		Preparación del	3 días	mar	jue 22/01/15	4
21		Pruebas	1 día	vie 23/01/15	vie 23/01/15	4
22		**Desarrollo**	**16 días**	**lun 26/01/15**	**lun 16/02/15**	**4**
23		Desarrollo	2 días	lun 26/01/15	mar	5
24		Desarrollo plugin	7 días	mié	jue 05/02/15	5
25		Pruebas	1 día	vie 06/02/15	vie 06/02/15	5
26		Corrección de	1 día	lun 09/02/15	lun 09/02/15	5
27		Creación Manual	2 días	mar	mié	5
28		Creación Web de	2 días	jue 12/02/15	vie 13/02/15	5
29		Publicación del	1 día	lun 16/02/15	lun 16/02/15	5
30		**Drupal**	**18 días**	**mar 17/02/15**	**jue 12/03/15**	**3**
31		Estudio del gestor	1 día	mar	mar	4
32		Preparación del	1 día	mié	mié	4
33		Pruebas	1 día	jue 19/02/15	jue 19/02/15	4
34		**Desarrollo**	**15 días**	**vie 20/02/15**	**jue 12/03/15**	**4**
35		Desarrollo	2 días	vie 20/02/15	lun 23/02/15	5
36		Desarrollo	7 días	mar	mié	5
37		Pruebas	1 día	jue 05/03/15	jue 05/03/15	5
38		Corrección de	1 día	vie 06/03/15	vie 06/03/15	5
39		Creación Manual	2 días	lun 09/03/15	mar	5
40		Creación Web de	1 día	mié	mié	5
41		Publicación del	1 día	jue 12/03/15	jue 12/03/15	5
42		**Trabajo autónomo**	**30 días**	**vie 13/03/15**	**jue 23/04/15**	**1**
43		**Wordpress**	**18 días**	**vie 13/03/15**	**mar 07/04/15**	**2**
44		Estudio del gestor de	1 día	vie 13/03/15	vie 13/03/15	3
45		Preparación del	1 día	lun 16/03/15	lun 16/03/15	3
46		Pruebas	1 día	mar	mar	3
47		**Desarrollo**	**15 días**	**mié**	**mar**	**3**
48		Desarrollo plugins	2 días	mié	jue 19/03/15	4
49		Desarrollo plugin	7 días	vie 20/03/15	lun 30/03/15	4
50		Pruebas	1 día	mar	mar	4
51		Corrección de	1 día	mié	mié	4
52		Creación Manual	2 días	jue 02/04/15	vie 03/04/15	4
53		Creación Web de	1 día	lun 06/04/15	lun 06/04/15	4
54		Publicación del	1 día	mar	mar	4
55		**Doumentacion**	**11 días**	**jue 09/04/15**	**jue 23/04/15**	**2**
56		Analizar puntos	3 días	jue 09/04/15	lun 13/04/15	3
57		Analizar los	3 días	mar	jue 16/04/15	3
58		Analizar que CMS se	2 días	vie 17/04/15	lun 20/04/15	3
59		Finalizar y editar la	1 día	mar	mar	3
60		Crear la presentación	2 días	mié	jue 23/04/15	3
61		**Tramitación final**	**21 días**	**lun 11/05/15**	**lun 08/06/15**	**1**
62		Solicitud de defensa	1 día	lun 11/05/15	lun 11/05/15	2
63		Defensa del TFM	1 día	lun 08/06/15	lun 08/06/15	2

Figura 11.1. Gráfico de Gantt de la planificación inicial parte 1.

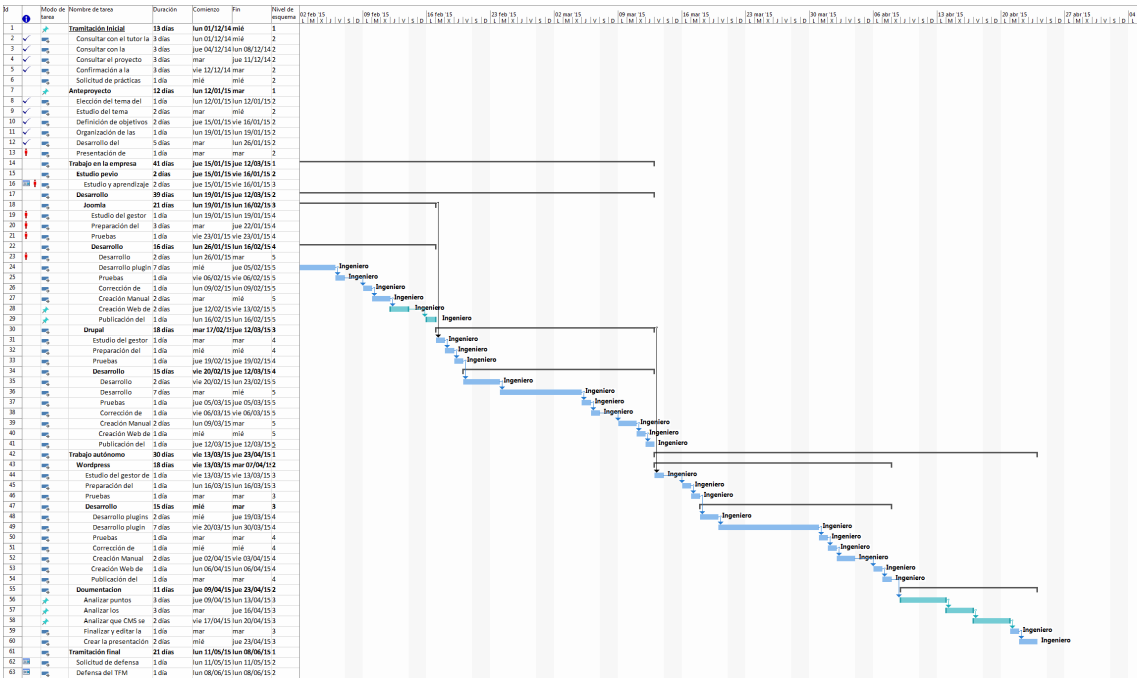
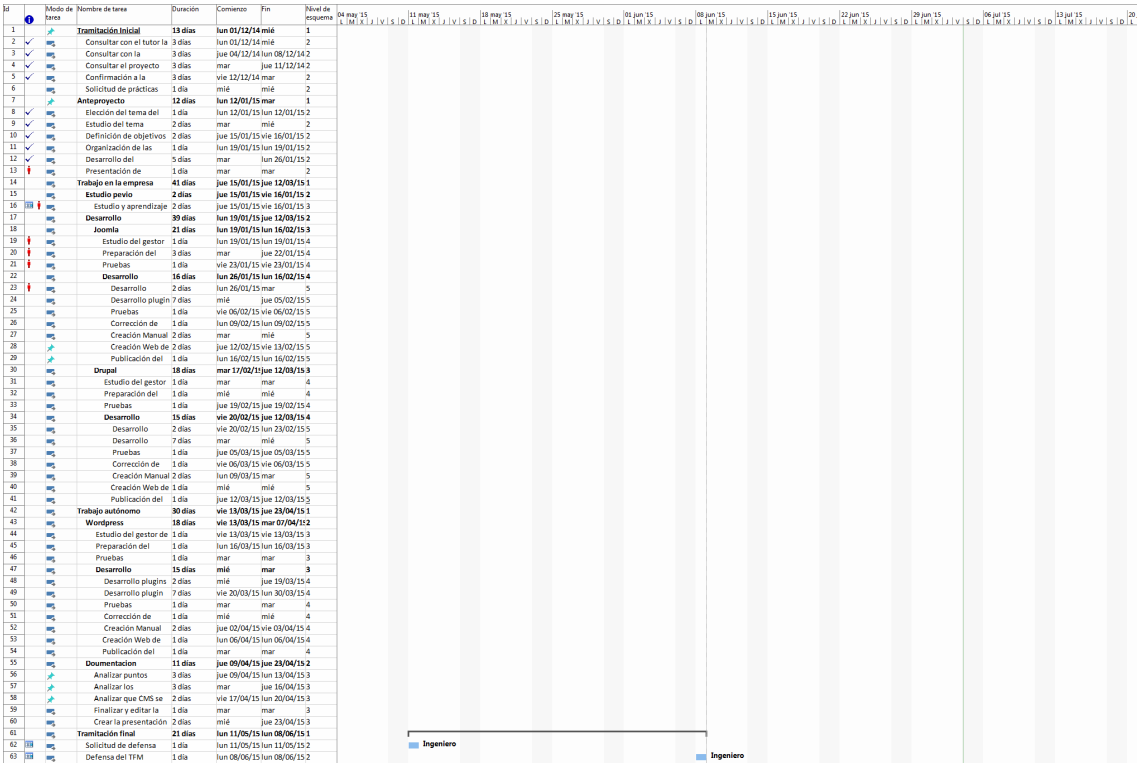


Figura 11.2. Gráfico de Gantt de la planificación inicial parte 2.



Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Nivel de esquema
1		**Tramitación Inicial**	**13 días**	**lun 01/12/14**	**mié**	**1**
2		Consultar con el tutor la	3 días	lun 01/12/14	mié	2
3		Consultar con la	3 días	jue 04/12/14	lun 08/12/14	2
4		Consultar el proyecto	3 días	mar	jue 11/12/14	2
5		Confirmación a la	3 días	vie 12/12/14	mar	2
6		Solicitud de prácticas	1 día	mié	mié	2
7		**Anteproyecto**	**12 días**	**lun 12/01/15**	**mar**	**1**
8		Elección del tema del	1 día	lun 12/01/15	lun 12/01/15	2
9		Estudio del tema	2 días	mar	mié	2
10		Definición de objetivos	2 días	jue 15/01/15	vie 16/01/15	2
11		Organización de las	1 día	lun 19/01/15	lun 19/01/15	2
12		Desarrollo del	5 días	mar	lun 26/01/15	2
13		Presentación de	1 día	mar	mar	2
14		**Trabajo en la empresa**	**41 días**	**jue 15/01/15**	**jue 12/03/15**	**1**
15		**Estudio previo**	**2 días**	**jue 15/01/15**	**vie 16/01/15**	**2**
16		Estudio y aprendizaje	2 días	jue 15/01/15	vie 16/01/15	3
17		**Desarrollo**	**39 días**	**lun 19/01/15**	**jue 12/03/15**	**2**
18		**Joomla**	**21 días**	**lun 19/01/15**	**lun 16/02/15**	**3**
19		Estudio del gestor	1 día	lun 19/01/15	lun 19/01/15	4
20		Preparación del	3 días	mar	jue 22/01/15	4
21		Pruebas	1 día	vie 23/01/15	vie 23/01/15	4
22		**Desarrollo**	**16 días**	**lun 26/01/15**	**lun 16/02/15**	**4**
23		Desarrollo	2 días	lun 26/01/15	mar	5
24		Desarrollo plugin	7 días	mié	jue 05/02/15	5
25		Pruebas	1 día	vie 06/02/15	vie 06/02/15	5
26		Corrección de	1 día	lun 09/02/15	lun 09/02/15	5
27		Creación Manual	2 días	mar	mié	5
28		Creación Web de	2 días	jue 12/02/15	vie 13/02/15	5
29		Publicación del	1 día	lun 16/02/15	lun 16/02/15	5
30		**Drupal**	**18 días**	**mar 17/02/15**	**jue 12/03/15**	**3**
31		Estudio del gestor	1 día	mar	mar	4
32		Preparación del	1 día	mié	mié	4
33		Pruebas	1 día	jue 19/02/15	jue 19/02/15	4
34		**Desarrollo**	**15 días**	**vie 20/02/15**	**jue 12/03/15**	**4**
35		Desarrollo	2 días	vie 20/02/15	lun 23/02/15	5
36		Desarrollo	7 días	mar	mié	5
37		Pruebas	1 día	jue 05/03/15	jue 05/03/15	5
38		Corrección de	1 día	vie 06/03/15	vie 06/03/15	5
39		Creación Manual	2 días	lun 09/03/15	mar	5
40		Creación Web de	1 día	mié	mié	5
41		Publicación del	1 día	jue 12/03/15	jue 12/03/15	5
42		**Trabajo autónomo**	**30 días**	**vie 13/03/15**	**jue 23/04/15**	**1**
43		**Wordpress**	**18 días**	**vie 13/03/15**	**mar 07/04/15**	**2**
44		Estudio del gestor de	1 día	vie 13/03/15	vie 13/03/15	3
45		Preparación del	1 día	lun 16/03/15	lun 16/03/15	3
46		Pruebas	1 día	mar	mar	3
47		**Desarrollo**	**15 días**	**mié**	**mar**	**3**
48		Desarrollo plugins	2 días	mié	jue 19/03/15	4
49		Desarrollo plugin	7 días	vie 20/03/15	lun 30/03/15	4
50		Pruebas	1 día	mar	mar	4
51		Corrección de	1 día	mié	mié	4
52		Creación Manual	2 días	jue 02/04/15	vie 03/04/15	4
53		Creación Web de	1 día	lun 06/04/15	lun 06/04/15	4
54		Publicación del	1 día	mar	mar	4
55		**Doumentacion**	**11 días**	**jue 09/04/15**	**jue 23/04/15**	**2**
56		Analizar puntos	3 días	jue 09/04/15	lun 13/04/15	3
57		Analizar los	3 días	mar	jue 16/04/15	3
58		Analizar que CMS se	2 días	vie 17/04/15	lun 20/04/15	3
59		Finalizar y editar la	1 día	mar	mar	3
60		Crear la presentación	2 días	mié	jue 23/04/15	3
61		**Tramitación final**	**21 días**	**lun 11/05/15**	**lun 08/06/15**	**1**
62		Solicitud de defensa	1 día	lun 11/05/15	lun 11/05/15	2
63		Defensa del TFM	1 día	lun 08/06/15	lun 08/06/15	2

Figura 11.3. Gráfico de Gantt de la planificación inicial parte 3.

3.3 Planificación final

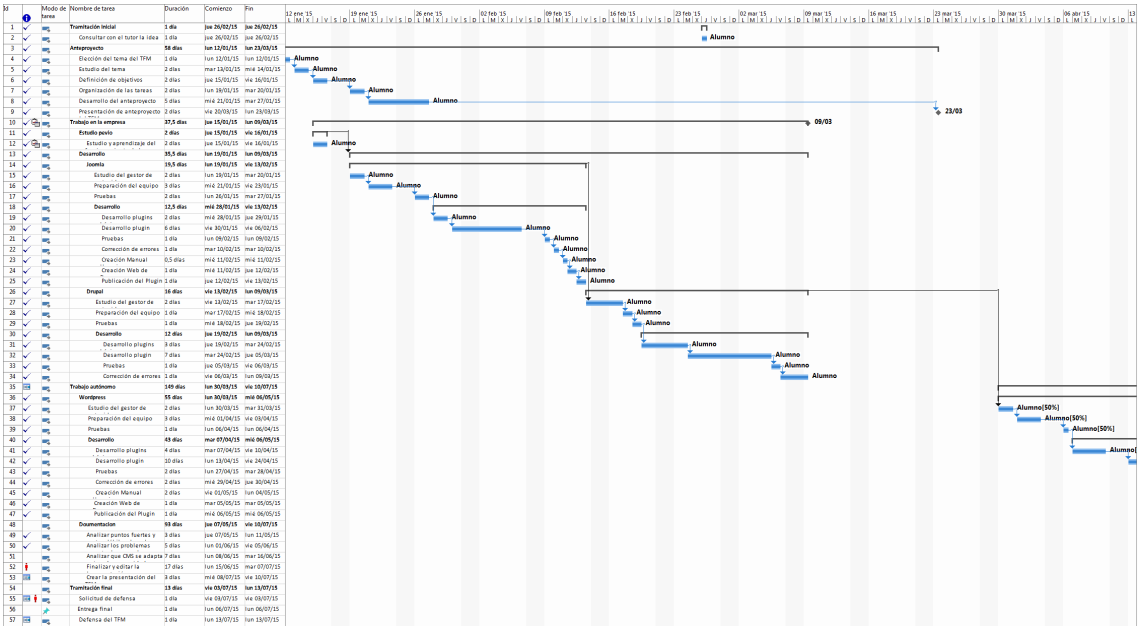


Figura 12.1. Gráfico de Gantt de la planificación final parte 1.

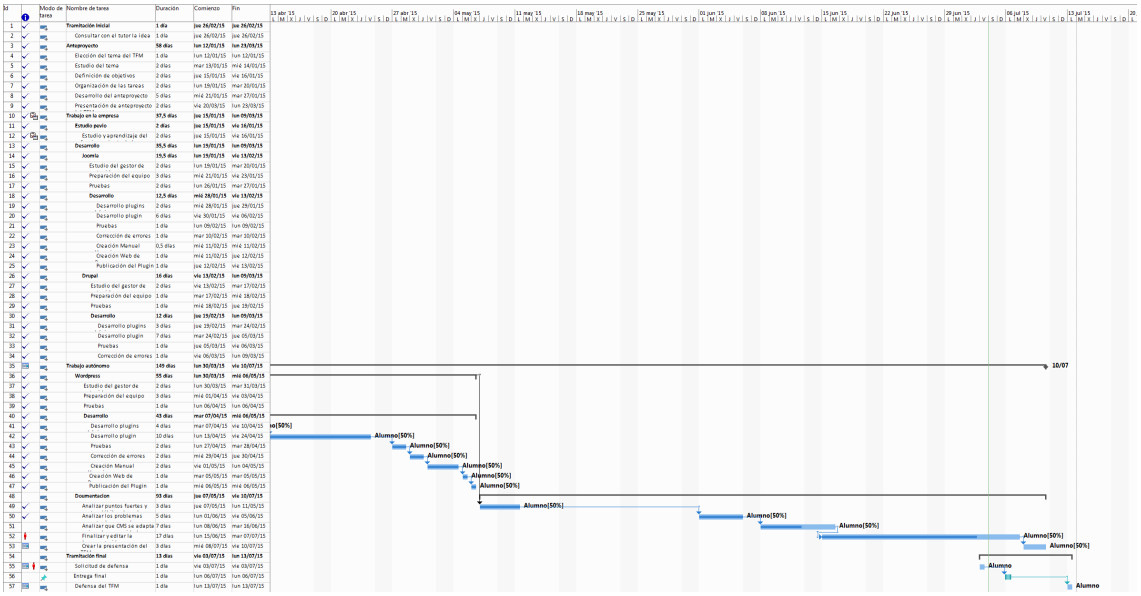


Figura 12.2. Gráfico de Gantt de la planificación final parte 2.

Adjuntas a este proyecto, pueden verse las imágenes de la planificación en el directorio /Planificación.

Capítulo 4. Integración de Asistentes Virtuales en Gestores de Contenido

En el siguiente apartado se detalla paso a paso el camino seguido para llevar a cabo este proyecto. Desde la introducción a los Asistentes Virtuales, y más concretamente a los desarrollados por la compañía Adele Robots, a la integración final de dichos Asistentes con los Gestores de Contenidos Joomla, Drupal y WordPress.

4.1 Asistente Virtual Fiona

Como se ha visto en el apartado 2.3 de este proyecto, los asistentes virtuales tienen dos componentes claves, la interfaz de usuario y la base de conocimiento. A continuación se muestra como configurar dichos componentes en el caso concreto de Fiona.

4.1.1 Personalización

En el apartado de personalización se distinguen las cuentas gratuitas de las cuentas de pago. Las primeras tienen unas opciones de personalización ligeramente inferiores, además de tener limitado el número de instancias del asistente, de manera que solo puede haber un cliente interactuando con el avatar al mismo tiempo. Podríamos compararlo con tener a un único asistente técnico en el servicio de atención al cliente, de manera que sólo es capaz de atender una llamada.

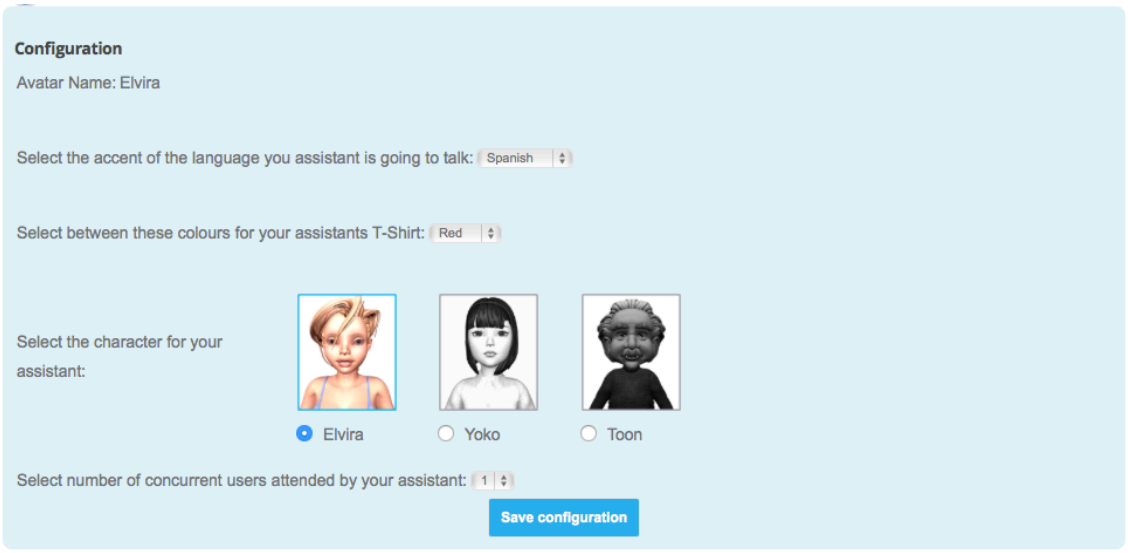


Figura 13. Panel de configuración de Fiona.

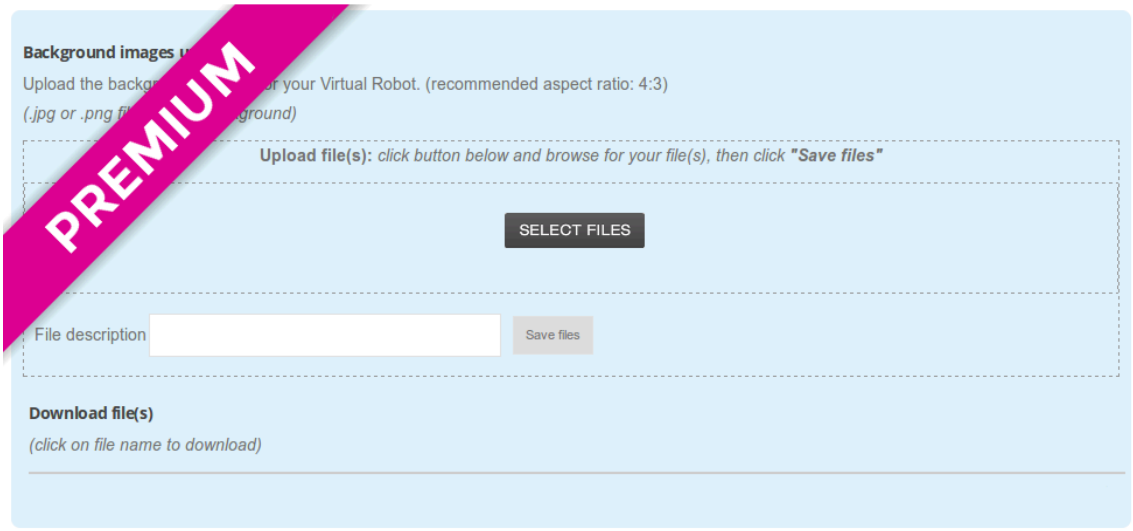


Figura 14. Opciones de personalización únicas de la versión Pro.

4.1.2 Conocimiento

Como se ha mencionado en apartados anteriores, el asistente virtual necesita una base de conocimiento para poder comunicarse con el cliente, por lo que es necesario incluirle el/los fichero/s AIML que representen el conocimiento que se desea aportar a Fiona. Este proceso puede resultar, y se ha demostrado que resulta, demasiado complejo para un usuario estándar de la web como al que van dirigidas las extensiones desarrolladas en este proyecto. Por ello, como trabajo previo al mismo, y en colaboración con desarrolladores de la compañía, se ha implementado un generador de AIML basado en una interfaz gráfica en la que generar

categorías, preguntas y respuestas resulta muy sencillo (www.theclientrelationsfactory.com/editor).

El editor está separado en tres apartados:

- Topics: Tabla en la que se organizan las categorías de preguntas y respuestas del asistente.

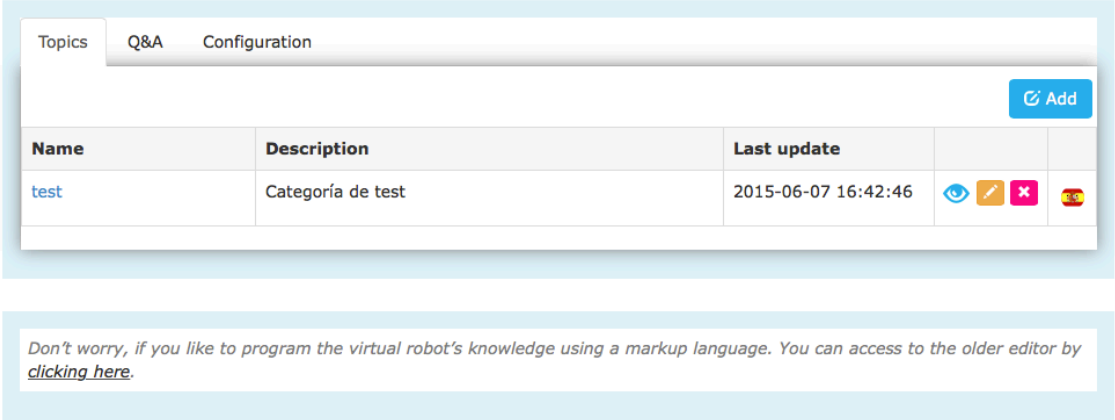


Figura 15. Panel de edición de categorías del editor AIML.

- Q&A: En este apartado se especifican las preguntas y respuestas que componen el conocimiento del avatar.

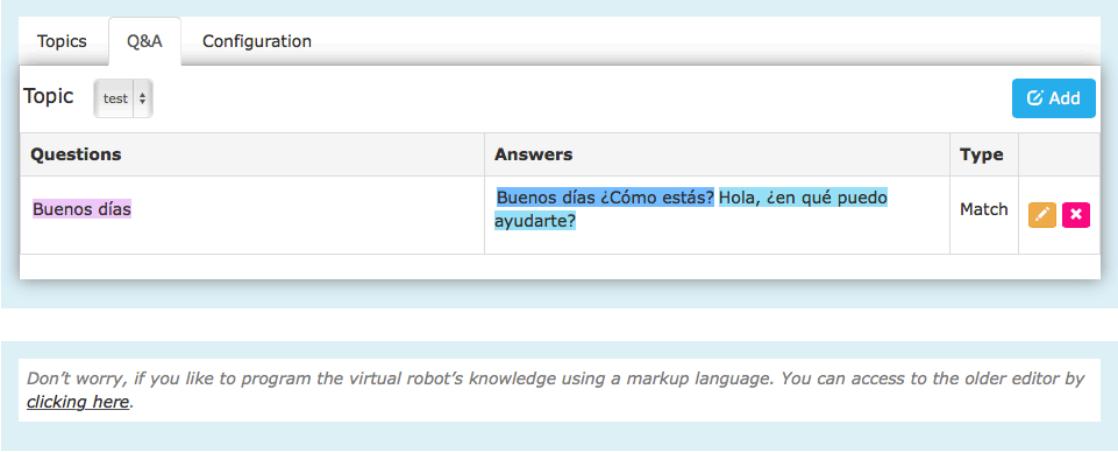


Figura 16. Ventana principal del editor de preguntas y respuestas.

En el momento de añadir las preguntas y respuestas, una de las opciones más interesantes del lenguaje de marcado AIML es poder asociar una lista de preguntas a una lista de respuestas, de manera que sea el sistema el que, ante una pregunta de entrada, escoja de manera aleatoria

una de las respuestas. De este modo, la interacción se asemeja mucha más a la interacción real entre dos personas.

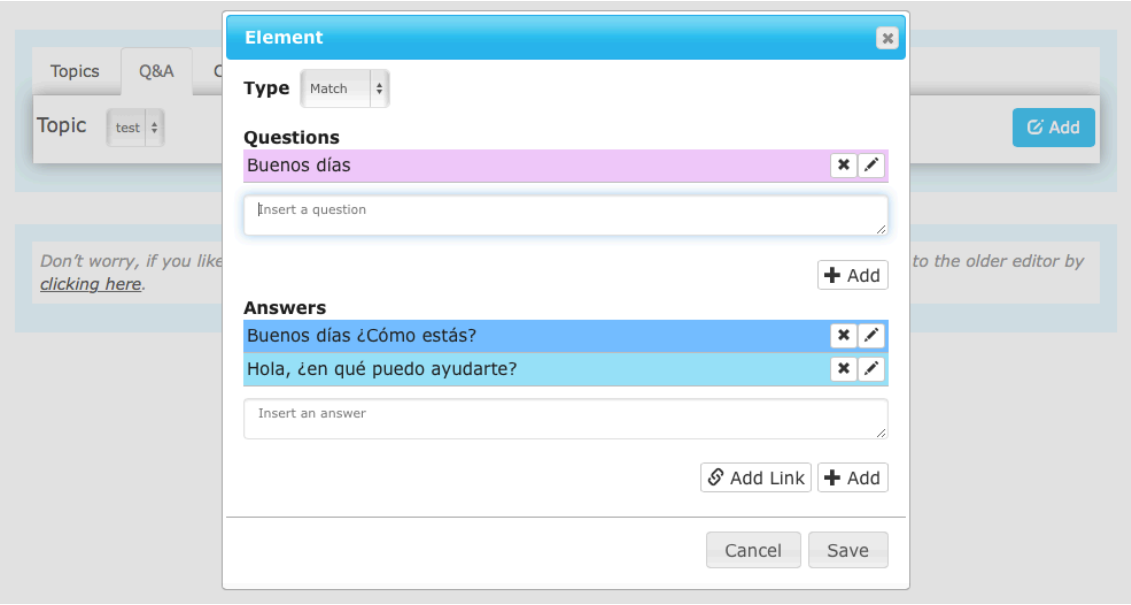


Figura 17. Cuadro de inserción de preguntas y respuestas.

- Configuration: Se establece el saludo inicial y la respuesta por defecto en caso de que el asistente no sea capaz de resolver una pregunta. Es decir, si el usuario introduce una pregunta para la cual el avatar no tiene una respuesta predefinida, responderá con la respuesta introducida en este campo.

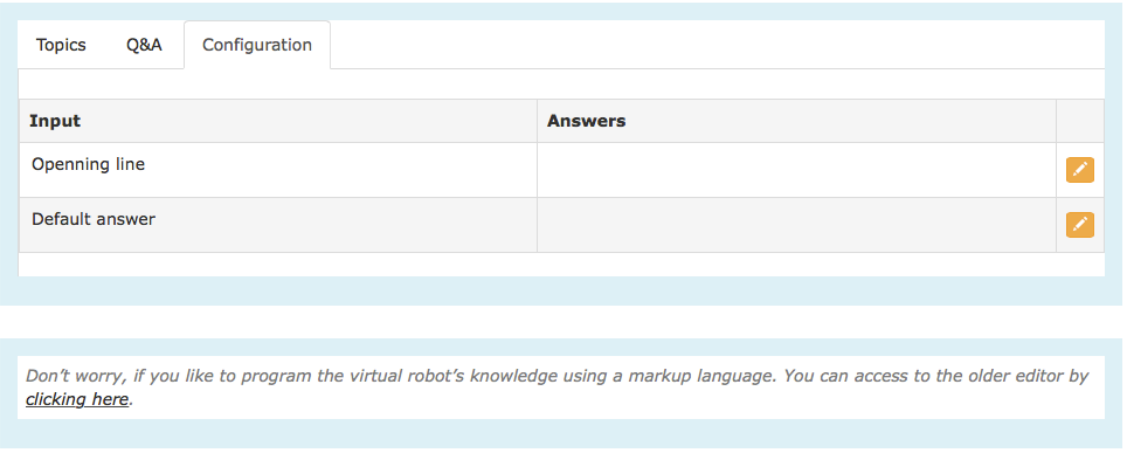


Figura 18. Panel de configuración de saludo inicial y respuesta por defecto.

4.1.3 Instalación

Una vez que el usuario ha configurado a su gusto tanto el aspecto estético del avatar como su nivel de conocimiento, es momento de integrarlo en su sitio web. La compañía Adele Robots, hasta la realización de este proyecto, únicamente proporcionaba a los usuarios un sistema de integración mediante JavaScript.

El sistema consiste en un bloque de JavaScript que el usuario debe insertar en el código fuente de su sitio web. Este script consta de:

- Bloque de variables que contienen las características de configuración del avatar.
- Función para generar el código de inclusión de un fichero JavaScript en el que se establece toda la funcionalidad del avatar.
- Bloque div contenedor para el marco del avatar.

```
<script type='text/javascript'>
usermail = "amenendezm@gmail.com";
usrid2 = "d1780a0c51883bb75908cd5ca5b3436b";
usrid1 = "068b608fcc8984428b444124d152135a";
avatar_size = "big";//big or small
avname = "Elvira";
avatarId = "229";
show_pop_up = true; // true or false
allow_camera = false; // true or false
(function() {
var fi = document.createElement('script');
fi.type = 'text/javascript';
fi.async = true;
fi.src = ('http://sm.adelerobots.com/js/servicemng.js');
var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(fi, s);
})();
</script>
<div id="fiona" class="fiona_main" style="display:none;"></div>
```

Figura 19. Bloque JavaScript.

Una vez incluido el primer bloque de código, se puede asociar a un elemento interactivo, un botón por ejemplo, o una imagen.

```
onclick="initFiona(this,usermail,usrid2,usrid1,avatarId)"
```

Figura 20. Evento "onclick" asociado a la llamada al avatar.

Como se puede observar, son necesarios, al menos, unos conocimientos básicos de programación web para llevar a cabo la integración del asistente virtual con el sitio web del cliente.

4.2 Gestores de Contenidos

Adele Robots, quería conseguir que cualquier usuario que desease crear un sitio web, pudiese, sin necesidad de adquirir conocimientos nuevos y de manera inmediata, integrar sus asistentes virtuales en él, por lo que crear extensiones para los principales gestores de contenidos parece un buen punto de partida.

En base a esto, se ha realizado el análisis y desarrollo de la integración de Fiona en Joomla, Drupal y WordPress. Los procedimientos seguidos para ello se detallan a continuación.

4.2.1 Joomla

4.2.1.1 Desarrollo

Tras leer la documentación disponible acerca de ampliaciones de este gestor de contenidos, se ha decidido optar por desarrollar un plugin para la integración con los asistentes virtuales de Adele.

Como puede observarse en la planificación del proyecto, al tratarse de una herramienta desconocida, se han desarrollado plugins de prueba con funcionalidades mucho más reducidas, en concreto 14, pero cuya discusión no aporta nada a esta Memoria, ya que su carácter es meramente didáctico para el desarrollador.

Como todas las extensiones en Joomla, los plugin se instalan de manera sencilla a través de un fichero .Zip. Sin embargo, los archivos contenidos por este fichero, no son para nada arbitrarios:

4.2.1.1.1 Archivos del plugin

Un plugin para Joomla tiene que estar formado por:

- Archivo PHP que contiene el código del plugin.
- Archivo de instalación. En este archivo, escrito en XML, se deben incluir:
  - Información acerca de la procedencia del plugin como nombre del desarrollador, fecha de creación, licencia, versión…
  - Listado de todos los archivos incluidos dentro el fichero .Zip.
  - Campos de la página de configuración. Se entrará más en detalle en apartados posteriores.
- Archivo index.html en blanco. Se recomienda la inclusión de un archivo HTML en blanco por motivos de seguridad. El directorio que contiene el plugin es accesible si se conoce la url exacta de destino, añadiendo un index.html en blanco los

navegadores, por defecto, mostraran este archivo en vez de listar el contenido del directorio.

- Carpeta assets. El nombre de esta carpeta puede variar, pero se recomienda, para seguir unas pautas de organización, incluir los archivos "accesorios" al plugin (CSS, JavaScript…) en esta carpeta.

4.2.1.1.2 Enfoque

Antes de comenzar el desarrollo, es importante hacer una labor de abstracción para intentar tener un enfoque más global del proyecto y de lo que se va a desarrollar. En el caso de este plugin:

- Intención: Integrar a Fiona con Joomla.
- Método: Mediante un plugin
- Material necesario:
    - Script proporcionado por Adele robots (se modificará ligeramente).
    - Fichero XML de configuración, nombrado fionaPlugin.xml
    - Fichero PHP con la funcionalidad del plugin, nombrado fionaPlugin.php
    - CSS para dar estilo al plugin, nombrado style.css
    - Clave pública. También hará falta un clave pública para conectarnos a los servidores de Adele Robots, se verá en el desarrollo de fionaPlugin.php

4.2.1.1.3 Implementación

Como hemos visto, el archivo fionaPlugin.php debe contener la funcionalidad que deseemos implementar en el plugin, en el caso concreto de este plugin la forma de hacerlo es conectarse a los servidores de la compañía a través de una conexión segura (SSL).

En el script original que se proporciona en el sitio web de Adele Robots tras generar el asistente virtual, vienen precargados una serie de datos personalizados:


```
<script type='text/javascript'>
usermail = "amenendezm@gmail.com";
usrid2 = "d1780a0c51883bb75908cd5ca5b3436b";
usrid1 = "068b608fcc8984428b444124d152135a";
avatar_size = "big";//big or small
avname = "Elvira";
avatarId = "229";
show_pop_up = true; // true or false
allow_camera = false; // true or false
(function() {
var fi = document.createElement('script');
fi.type = 'text/javascript';
fi.async = true;
fi.src = ('http://sm.adelerobots.com/js/servicemng.js');
var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(fi, s);
})();
</script>
<div id="fiona" class="fiona_main" style="display:none;"></div>
```

Figura 21. Bloque JavaScript detallado.

Como el plugin que estamos desarrollando, debe funcionar para cualquier usuario, se deben obtener estos datos de manera dinámica con los datos de acceso de cada usuario. Conectándose por SSL a los servidores de la compañía y proporcionando, por un lado las credenciales del usuario, y por otro, la clave pública mencionada anteriormente, se obtendrán dichos datos.

Dado que se necesitan los datos de acceso del usuario, se debe crear una página de configuración del plugin para que el usuario pueda registrar sus credenciales. Según la especificación de Joomla, dicha página de configuración se establece en el fichero de instalación. Se aprovechará esta página, además de para recoger las credenciales del usuario, para permitir al usuario introducir parámetros de configuración del avatar.

4.2.1.1.3.1 fionaPlugin.xml

```
    <config>
        <fields name="params">
            <fieldset name="basic">
                <field name="user-options" type="spacer" default="User Settings" label="&lt;b&gt;User Settings&lt;/b&gt;" description="Settings of your
                The Client Relations Factory account." />
                    <field name="client-relations" type="spacer" description="The Client Relations Factory" label="Please Sign In with your The Client
                    Relations Factory credentials. If you don't have an account yet... &lt;a href='http://www.theclientrelationsfactory.com/#price'&gt;
                    Register Now!&lt;/a&gt;" />
                    <field name="email"  type="text" default="" required="true" validate="email" label="Username" description="User of The Client
                    Relations Factory." />
                    <field name="password"  type="password" default="" required="true" label="Password" description="Password of The Client Relations
                    Factory." />
                    <field name="user-options-advice" type="spacer" default="User Settings Advice" label="&lt;i&gt;Attention: If the Virtual Robot's start
                    button is not shown, please check your credentials.&lt;/i&gt;"/>
                <field name="avatar-options" type="spacer" default="Virtual Robot Settings" label="&lt;b&gt;Virtual Robot Settings&lt;/b&gt;" description="
                Settings of your Virtual Robot's display." />
                    <field name="avatar_size" type="radio" default="big" description="Sets the size of your Virtual Robot display frame." label="Avatar
                    Size">
                        <option value="big">Big</option>
                        <option value="small">Small</option>
                    </field>
                    <field name="topBarColor" type="list" default="#CCC" description="Sets the color of your Virtual Robot's widget dialog top bar." label=
                    "Dialog Top Bar Color">
                        <option value="#3CE">Blue</option>
                        <option value="#4F5">Green</option>
                        <option value="#CCC">Grey</option>
                        <option value="#F44">Red</option>
                        <option value="#EF3">Yellow</option>
                    </field>
                    <field name="btnText"  type="text" default="" label="Send Button Text" description="Defines the text that will be displayed into the
                    Send button of the Virtual Robot's chat dialog." />
                <field name="button-options" type="spacer" default="Virtual Robot's Start Button Settings" label="&lt;b&gt;Virtual Robot's Start Button
                Settings&lt;/b&gt;" description="Settings of your Virtual Robot's start button" />
                    <field name="chatBtnText"  type="text" default="" label="Button text" description="Defines the text that will be displayed on the
                    Virtual Robot's start button." />
            </fieldset>
        </fields>
    </config>
</extension>
```

Figura 22. fionaPlugin.xml_1.

Como se puede observar en la imagen superior toda la información que se debe "solicitar" al usuario se encuentra dentro de las etiquetas `<fieldset></fieldset>`.

Dentro del fieldset, se organizan los campos con las etiquetas <field></field>, definiendo el tipo de cada uno según el tipo de campo que se quiera crear:

- type="spacer" sirve para mantener agrupados campos con características comunes.
- type="text" crea un input de texto.
- type="password" genera un campo para una contraseña
- type="radio" construye un formulario de radio botones.
- type="list" implementa una lista desplegable de opciones.

Se completa fionaPlugin.xml con los datos del desarrollador, licencia, etc.:

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="3.3" type="plugin" group="content" method="upgrade">
    <name>Content - The Client Relations Factory</name>
    <author>Adele Robots Inc.</author>
    <creationDate>January 2015</creationDate>
    <copyright>Copyright (C) 2015</copyright>
    <license>GPLv3</license>
    <authorUrl>http://www.adelerobots.com/</authorUrl>
    <version>1.0</version>
    <description>This plugin let you have your own Virtual Robot from The Client Relations Factory
    fully integrated within Joomla.</description>
    <files>
        <filename plugin="fionaPlugin">fionaPlugin.php</filename>
        <filename>index.html</filename>
        <filename>assets/key/pub.key</filename>
        <filename>assets/css/style.css</filename>
    </files>
```

Figura 23. fionaPlugin.xml_2.

Con estas construcciones, ya está lista la página de configuración del asistente virtual. Una vez instalado el plugin quedará de la siguiente manera:

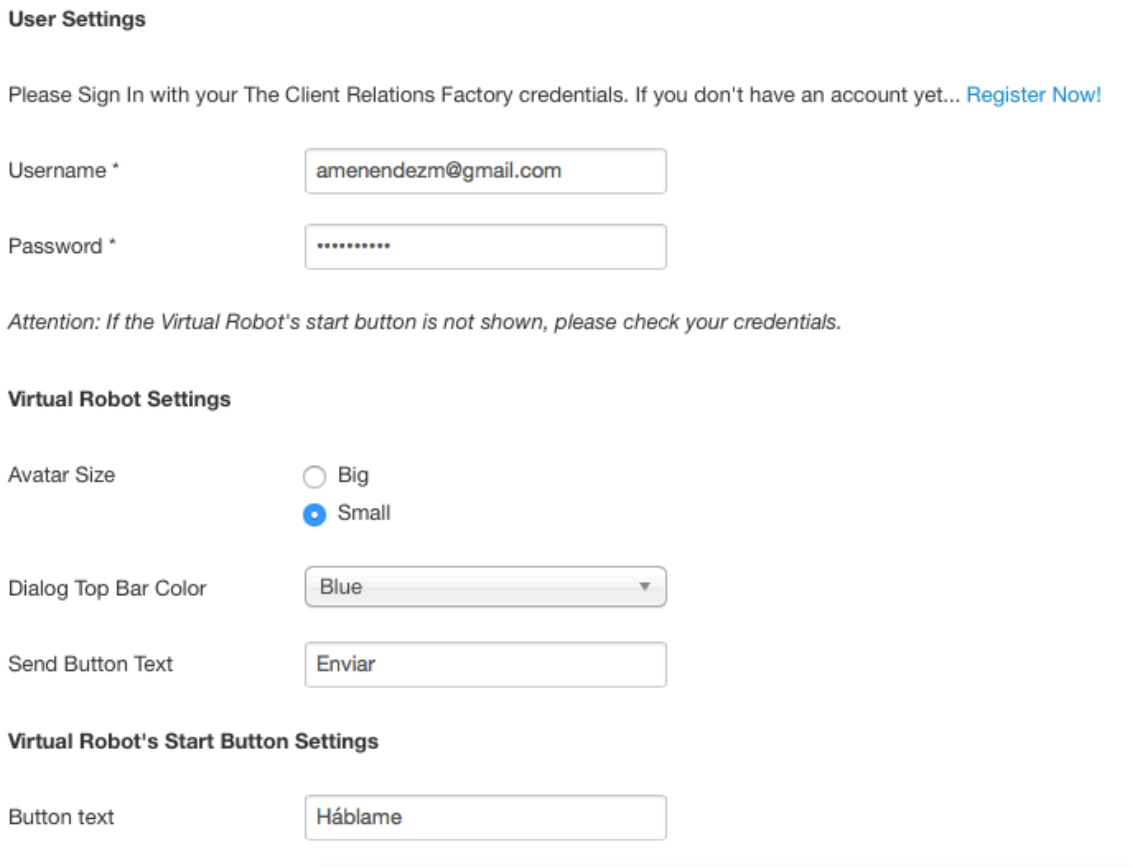


Figura 24. Visualización de la página de configuración.

Una vez recopilados los datos necesarios del usuario, hay que centrarse en el desarrollo de la funcionalidad del plugin, archivo fionaPlugin.php.

4.2.1.1.3.2 fionaPlugin.php

El desarrollo de Joomla, está basado en conseguir un framework lo más fiel posible al paradigma de la orientación a objetos. Por ello, en las últimas versiones se ha desarrollado un sistema nuevo de plugins que sigue el patrón Observer. Los plugin son las clases "Observer", ligadas a un manejador de eventos global del Core del CMS.

Para que el plugin forme parte de dicho patrón, debe heredar de la clase JPlugin.

```
class plgContentFionaPlugin extends JPlugin {

}
```

Figura 25. Herencia de JPlugin.

Ya está la clase del plugin definida, es el momento de decidir cuando el plugin "ha de entrar en acción".

Para ejecutar la funcionalidad de los plugins, Joomla incorpora un sistema de eventos que permiten "disparar" la funcionalidad del plugin cuando uno de estos eventos ocurra. En este caso particular, se necesita un evento que se dispare cada vez que se cargue una página, de manera que el botón de llamada al asistente virtual esté disponible en todo el sitio. Sin embargo, no es necesario que éste aparezca en el administrador del portal, ya que son páginas privadas para los administradores.

Con todas las condiciones anteriores, el evento elegido que parece adecuarse más a ellas es el llamado onContentPrepare.

El evento onContentPrepare es la primera etapa para la generación de contenido. Cuando el contenido está preparado y listo para mostrarse, puede ser modificado invocando a este evento. En este caso, se utilizará para conocer cuando se está preparando contenido para publicar, pues es precisamente en las páginas en las que se publica contenido en las que se ha de mostrar el asistente virtual.

Para "conectar" con el manejador y que nuestro código se ejecute en el momento en que "salte" el evento onContentPrepare, creamos una función homónima.

```
public function onContentPrepare($context, &$row, &$params, $page = 0) {

}
```

Figura 26. Función asociada al evento onContentPrepare.

En este punto ya está configurado el plugin para que responda a los eventos que se han elegido, por lo que el desarrollo se ha de centrar en la funcionalidad.

Lo primero que se necesita para cargar el asistente virtual, es el script que lo contiene. Este script, contiene dos claves que se deben obtener a partir de una contraseña proporcionada por la compañía en tiempo real. Para obtenerla, es necesaria una conexión con el servidor de Adele Robots. Se necesitan, para ello, los datos que el usuario ha introducido en la página de configuración que se ha creado anteriormente y una clave pública facilitada por la compañía.

Para acceder a los parámetros introducidos por el usuario en el formulario de configuración, Joomla facilita: $this->params->get(), que accede a los parámetros introducidos en la página de configuración del plugin.

```
//Get parameters from Joomla Plugin Manager
//User settings
$email = $this->params->get('email');
$password = $this->params->get('password');
//Avatar settings
$avatar_size = $this->params->get('avatar_size');
$btnText = $this->params->get('btnText', 'Send');
$topBarColor = $this->params->get('topBarColor', '#CCC');
//Chat button settings
$chatBtnText = $this->params->get('chatBtnText', 'Chat with me!');
```

Figura 27. Acceso a los parámetros.

Por comodidad, se almacenará la clave pública en el directorio del plugin, en la carpeta "key" del directorio "assets".

Con la clave pública y los datos de usuario, se puede realizar la conexión con Adele. Hay que tener en cuenta que el servicio web que la empresa proporciona para obtener la contraseña del avatar, genera JSON como respuesta, por lo que hay que decodificarlo antes de poder acceder a los datos.

```
//Connect to The Client Relations Factory
$public_key = file_get_contents(JURI::base() . 'plugins/content/fionaPlugin/assets/key/pub.key');

$login_data = json_encode(array('username' => $email, 'password' => $password));

openssl_public_encrypt($login_data, $login_data, $public_key);

$scriplet_service_url = 'http://www.theclientrelationsfactory.com/services/scriptlet-generator/get-data.php';

$options = array(
    'http' => array(
        'method'  => 'POST',
        'content' => $login_data,
        'header'=>  "Content-Type: application/json\r\n" .
                    "Accept: application/json\r\n"
    )
);

$context     = stream_context_create($options);
$result      = file_get_contents($scriplet_service_url, false, $context);
$response    = json_decode($result);
```

Figura 28. Conexión con los servidores.

Para poder "llamar" al asistente virtual, son necesarios un botón que lo active y un contenedor en el que insertarlo en el momento deseado.

Se comprueba que la respuesta del servidor ha sido correcta y se genera el código necesario.

```
//if correct response, load button.js
if (!$response->error && $response->found){
    echo '
    <script type="text/javascript">
        function load() {

            //Chat with me button
            var btn = document.createElement("button");
            var t = document.createTextNode("' . $chatBtnText . '");
            btn.appendChild(t);
            btn.setAttribute("onclick", "initFiona(this,usermail,usrid2,usrid1,avatarId)");
            btn.setAttribute("class", "fiona_button");

            document.body.appendChild(btn);

            //Fiona div
            var dv = document.createElement("div");
            dv.setAttribute("id", "fiona");
            dv.setAttribute("class", "fiona_main");
            dv.setAttribute("style", "display:none;");
            document.body.appendChild(dv);

        }

        window.onload = load;
    </script>
';
}
```

Figura 29. Código necesario para activar Fiona.

Por último, se genera el script personalizado con los datos obtenidos de los servidores de Adele.

La contraseña se utilizará para generar dos cadenas encriptadas con el algoritmo MD5 a modo de claves.

En la respuesta, además de la contraseña, se ha recibo datos del avatar que también se utilizan para generar el script, como son su nombre y su id.

```
//Generate Fiona Script
echo '

        <script type="text/javascript">
            usermail = "' . $email . '";
            usrid2 = "' . md5($response->avatar_password) . '";
            usrid1 = "' . md5($response->avatar_password . $response->avatar_id) . '";
            avatar_size = "' . $avatar_size . '";
            avname ="' . $response->avatar_name . '";
            avatarId ="' . $response->avatar_id . '";
            show_pop_up = true;
            allow_camera = false;
            btnText = "' . $btnText . '";
            dialogTopBarColor = "' . $topBarColor . '";
            (function() {
            var fi = document.createElement("script");
            fi.type = "text/javascript";
            fi.async = true;
            fi.src = ("http://sm.adelerobots.com/js/servicemng.js");
            var s = document.getElementsByTagName("script")[0];
            s.parentNode.insertBefore(fi, s);
            })();
        </script> ';
```

Figura 30. Script personalizado.

Con todo esto, ya está listo el archivo fionaPlugin.php que contiene la funcionalidad del plugin.

Como colofón, simplemente queda añadir un poco de estilo al botón que “llama” a Fiona. Se añade el fichero con la CSS (style.css) al directorio assets/css del plugin.

```css
.fiona_button {
    -webkit-transform: rotate(-90deg);
    -webkit-transform-origin: right center;
    -moz-transform-origin: right center;
    -ms-transform-origin: right center;
    transform-origin: right center;
    -moz-transform: rotate(-90deg);
    -ms-transform: rotate(-90deg);
    -o-transform: rotate(-90deg);
    transform: rotate(-90deg);

    color: #616161;
    font-weight: bold;
    background-color: #dddddd;
    border-left-color:  #dddddd;
    border-left-style: solid;
    border-right-color:  #dddddd;
    border-right-style: solid;
    border-top-color: #dddddd;
    border-top-style: solid;
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
    border-width: 3px 3px 0px;

    cursor: pointer;

    display: block;
    height: 30px;
    margin-top: -50px;
    min-width: 120px;
    position: fixed;
    right: 16px;
    top: 40%;
    z-index: 100000;
}
```

Figura 31. Hoja de estilo.

Fiona ya se encuentra completamente integrada en Joomla.

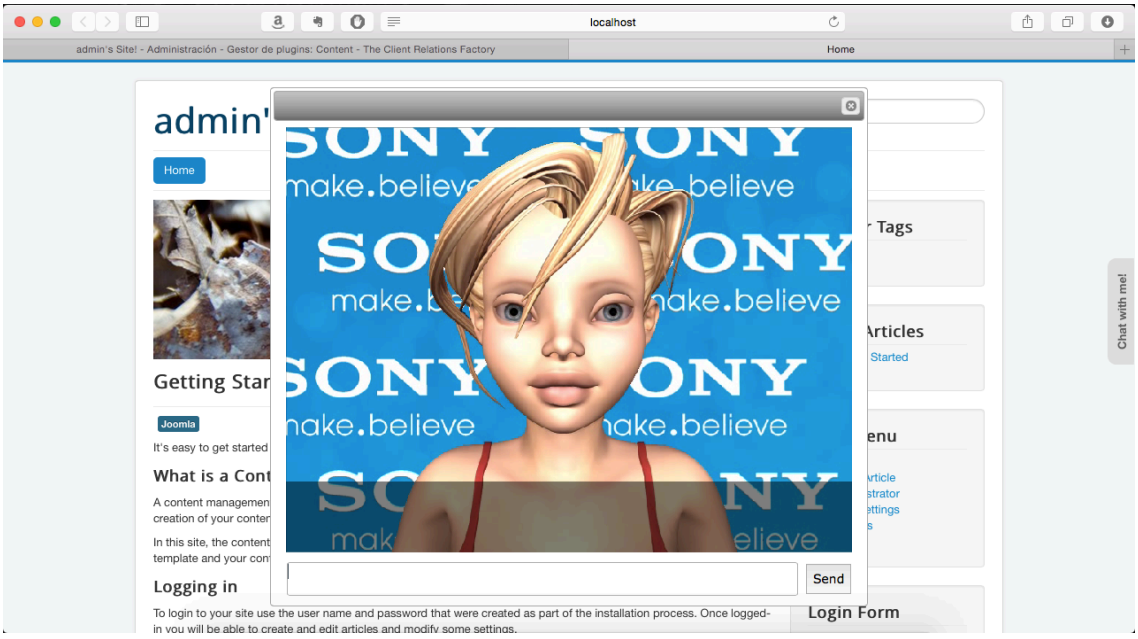


Figura 32. Fiona en Joomla.

4.2.2 Drupal

4.2.2.1 Desarrollo

4.2.2.1.1 Archivos del plugin

El directorio del plugin para Drupal, es muy similar al creado en Joomla:

- Archivo PHP principal del plugin que contendrá la funcionalidad del plugin.
- Archivo con la información del plugin (Nombre, descripción, desarrollador…).
- Directorio que contenga los archivos "accesorios" al plugin (JavaScript, CSS…).

4.2.2.1.2 Enfoque

Al igual que se ha hecho en los gestores anteriores, se debe organizar el enfoque antes de comenzar la implementación:

- tcrf.module: Archivo PHP que contiene el código del plugin con extensión.
- Tcrf.info: Archivo de información del plugin.
- Carpeta assets. El nombre de esta carpeta puede variar, pero se recomienda, para seguir unas pautas de organización, incluir los archivos "accesorios" al plugin (CSS, JavaScript…) en esta carpeta.

4.2.2.1.3 Implementación

Dado que la funcionalidad que se pretende conseguir con el desarrollo del módulo para Drupal, es la misma que para el plugin de Joomla, se procede a explicar directamente el desarrollo de los ficheros del módulo.

4.2.2.1.3.1 tcrf.module

Drupal, pese a no ser un Gestor de Contenido como es Joomla, que está muy orientado al paradigma de la orientación a objetos, sigue es cierto modo el patrón Observer. Mediante convención de nombres, se asocian eventos con funciones del plugin.

El nombrado que se ha de seguir consiste en el nombre del plugin, en este caso tcrf, guión bajo y nombre del "hook", por ejemplo, para crear una página de ayuda como recomienda Drupal:

```
function tcrf_help($path, $arg) {
  switch ($path) {
    case "admin/help#tcrf":
      return '<p>' . t("A block module that let you have your own Virtual Robot from The Client
        Relations Factory fully integrated within Drupal.") . '</p>';
      break;
  }
}
```

Figura 33. Función tcrf_help().

A continuación, hook para la información del plugin:

```
function tcrf_block_info() {
  $blocks['tcrf'] = array(
    // The name that will appear in the block list.
    'info' => t('The Client Relations Factory'),
    // Default setting.
    'cache' => DRUPAL_CACHE_PER_ROLE,
  );
  return $blocks;
}
```

Figura 34. Función tcrf_info().

Se ha de generar también, la información para el Menú y la página de configuración:

```
function tcrf_menu() {
  $items = array();

  $items['admin/config/content/tcrf'] = array(
    'title' => 'The Client Relations Factory',
    'description' => 'Configuration for The Client Relations Factory module.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('tcrf_form'),
    'access arguments' => array('access administration pages'),
    'type' => MENU_NORMAL_ITEM,
  );

  return $items;
}
```

Figura 35. Función tcrf_menu().

Al igual que ocurría en Joomla, es necesario obtener información del usuario a través de la página de configuración. A diferencia del anterior CMS, la página de configuración de los módulos de Drupal no se configura a través de un XML, si no que se realiza en una función dentro del archivo PHP de la funcionalidad del módulo:

```
function tcrf_form($form, &$form_state) {

}
```

Figura 36. Función tcrf_form().

Dentro de esta función, se debe definir, con una estructura preestablecida por el CMS, los campos de adquisición de datos del usuario. Como se puede observar en la figura siguiente, se puede hacer un "árbol" de etiquetas, con el cual crear un formulario y definir los campos que formarán parte de él. En este primer bloque se crea un formulario con el que recoger las credenciales que se utilizarán para conectar con el servidor de Adele Robots.

Al objeto form que ha llegado como parámetro a la función, se le añaden los campos de configuración. En primer lugar se declara un fieldset para englobar los campos de entrada de datos relacionados con la cuenta del usuario, se establece un párrafo informativo de creación de una cuenta por si el usuario no dispusiese de una y se declaran los campos de entrada de nombre de usuario y contraseña.

```
/*********User setting fields*********/
$form['usersettings'] = array(
  '#type' => 'fieldset',
  '#title' => t('User Settings'),
  '#collapsible' => TRUE, // Added
  '#collapsed' => FALSE,  // Added
);

$form['#prefix'] = "<p>" . t("Please Sign In with your The Client Relations Factory credentials.
  If you don't have an account yet... ") . "<a href='http://www.theclientrelationsfactory.
  com/#price' target='_blank'>" . t("Register Now!") . "</a></p>";

$form['usersettings']['username'] = array(
  '#type' => 'textfield',
  '#title' => t('Username'),
  '#description' => t('Your The Client Relations Factory username'),
  '#size' => 27,
);

$form['usersettings']['user_password'] = array(
  '#type' => 'password',
  '#title' => t('Password'),
  '#description' => t('Your The Client Relations Factory password'),
  '#size' => 27,
);
```

Figura 37. Campos de entrada de datos del usuario.

Al estar el código de la página de configuración en el fichero PHP, hay más posibilidades de mejorar la experiencia de usuario, y hacer que la configuración del módulo parezca más interactiva. Por ejemplo, en el siguiente fragmento de código, se establece como condición que exista la variable avatar_name para mostrar los campos de configuración del avatar. De esta manera, dado que la variable avatar_name solo existe (se verá más adelante por qué) cuando se ha establecido una conexión correcta con Adele Robots, los campos de configuración solo serán visibles tras introducir unas credenciales correctas en el formulario anterior.

```php
//Shown if the user is logged on The Client Relations Factory
if (variable_get('avatar_name')){
  //Unlink account button
  drupal_set_message(t('Connected with ') . variable_get('avatar_name'));
  $form['usersettings']['submit_unlink'] = array('#type' => 'submit', '#value' => t('Unlink Account!'), '#submit' => array('tcrf_form_submit_unlink'));


  /********Avatar setting fields*********/
  $form['avatarsettings'] = array(
    '#type' => 'fieldset',
    '#title' => t('Virtual Robot Settings'),
    '#collapsible' => TRUE,
    '#collapsed' => FALSE,
  );

  $window_size_options = array (
    'big' => t('Big'),
    'small' => t('Small'),
    );

  if(strtolower(variable_get("window_size")) == "small")
    $actual_window_size = "small";
  else
    $actual_window_size = "big";

  $form['avatarsettings']['window_size'] = array(
    '#type' => 'radios',
    '#title' => t('Avatar display size'),
    '#description' => t('Sets the size of your Virtual Robot display frame.'),
    '#options' => $window_size_options,
    '#default_value' => $actual_window_size
    );

  $form['avatarsettings']['top_bar_color'] = array(
    '#type' => 'textfield',
    '#title' => t('Dialog top bar color'),
    '#description' => t("Sets the color of your Virtual Robot's widget dialog top bar."),
    '#size' => 27,
    '#attributes' => array('class' => array('color {pickerFaceColor:"transparent", pickerFace:1,
     pickerBorder:0, pickerInsetColor:"black", slider:true, pickerPosition:"right", hash:true}'),
      'value' => array(variable_get("top_bar_color", "#CCCCCC"))),
  );

  $form['avatarsettings']['send_button_text'] = array(
    '#type' => 'textfield',
    '#title' => t('Chat button text'),
    '#description' => t("Defines the text that will be displayed into the send button of the Virtual Robot's chat dialog."),
    '#default_value' => variable_get("send_button_text", "Send") ,
    '#size' => 27,
    '#maxlength' => 6,
  );

  $form['avatarsettings']['start_button_text'] = array(
    '#type' => 'textfield',
    '#title' => t('Start button text'),
    '#description' => t("Defines the text that will be displayed on the Virtual Robot's start button."),
    '#default_value' => variable_get("start_button_text", "Chat with me!") ,
    '#size' => 27,
  );

  $form['avatarsettings']['submit_avatarsettings'] = array('#type' => 'submit', '#value' => t('Save'), '#submit' => array('tcrf_form_submit_avatarsettings'));
```

Figura 38. Campos de entrada de datos del avatar.

En la primera parte de esta sentencia condicional, dado que hemos establecido que existe un conexión con los servidores externos, se muestra un mensaje que verifica dicha condición mostrando el nombre del avatar obtenido y un botón para desvincular la cuenta. La funcionalidad de este botón se implementa en la función tcrf_form_submit_unlink().

```php
//Unlink account button
drupal_set_message(t('Connected with ') . variable_get('avatar_name'));
$form['usersettings']['submit_unlink'] = array('#type' => 'submit', '#value' => t('Unlink Account!'), '#submit' => array('tcrf_form_submit_unlink'));
```

Figura 39.

Siguiendo la misma metodología que para los datos de usuario, se recogen los parámetros de configuración de Fiona.

```php
$form['avatarsettings'] = array(
  '#type' => 'fieldset',
  '#title' => t('Virtual Robot Settings'),
  '#collapsible' => TRUE,
  '#collapsed' => FALSE,
);
```

Figura 40.

```
$window_size_options = array (
  'big' => t('Big'),
  'small' => t('Small'),
  );

if(strtolower(variable_get("window_size")) == "small")
  $actual_window_size = "small";
else
  $actual_window_size = "big";

$form['avatarsettings']['window_size'] = array(
  '#type' => 'radios',
  '#title' => t('Avatar display size'),
  '#description' => t('Sets the size of your Virtual Robot display frame.'),
  '#options' => $window_size_options,
  '#default_value' => $actual_window_size
  );
```

Figura 41.

En el fragmento de código que se muestra a continuación, se puede observar una de las grandes ventajas que aporta Drupal con respecto a Joomla y la página de configuración. A los elementos que la componen, se le pueden añadir atributos, que se verán reflejados en el código HTML generado por el CMS. Dichos atributos pueden ser utilizados, como en este caso, para dotar a un elemento de funcionalidad extra. En este caso concreto, se establecen los atributos *class* y *value* para generar un selector de color con la librería jscolor.

```
$form['avatarsettings']['top_bar_color'] = array(
  '#type' => 'textfield',
  '#title' => t('Dialog top bar color'),
  '#description' => t("Sets the color of your Virtual Robot's widget dialog top bar."),
  '#size' => 27,
  '#attributes' => array('class' => array('color {pickerFaceColor:"transparent", pickerFace:1,
   pickerBorder:0, pickerInsetColor:"black", slider:true, pickerPosition:"right", hash:true}'),
    'value' => array(variable_get("top_bar_color", "#CCCCCC"))),
);
```

Figura 42.

Por supuesto, debemos añadir dicha librería a la página. Drupal también ofrece una manera sencilla de hacerlo:

```
drupal_add_js(drupal_get_path('module', 'tcrf') . '/assets/js/jscolor/jscolor.js', array('scope' => 'header'));
```

Figura 43.

Se continúa con la generación de campos de entrada para los parámetros de configuración:

```
$form['avatarsettings']['send_button_text'] = array(
  '#type' => 'textfield',
  '#title' => t('Chat button text'),
  '#description' => t("Defines the text that will be displayed into the send button of the Virtual Robot's chat dialog."),
  '#default_value' => variable_get("send_button_text", "Send") ,
  '#size' => 27,
  '#maxlength' => 6,
);

$form['avatarsettings']['start_button_text'] = array(
  '#type' => 'textfield',
  '#title' => t('Start button text'),
  '#description' => t("Defines the text that will be displayed on the Virtual Robot's start button."),
  '#default_value' => variable_get("start_button_text", "Chat with me!") ,
  '#size' => 27,
);
```

Figura 44.

En el último fragmento de código de esta función se establece la funcionalidad del botón Link Account y de un mensaje, cuya visibilidad está condicionada al opuesto de la sentencia condicional anterior, es decir, que no exista la variable avatar_name, lo que se traduce en que no haya establecida una conexión con los servidores de Adele.

```
  else {
    $form['usersettings']['submit_link'] = array('#type' => 'submit', '#value' => t('Link Account!'), '#submit' => array('tcrf_form_submit_link'));
    drupal_set_message(t('Not connected with The Client Relations Factory'), $type = 'warning');
  }

  return $form;
}
```

Figura 45.

Como puede observarse en la imagen anterior, la funcionalidad del botón consiste en invocar a la función tcrf_form_submit_link().

Como se ha visto anteriormente, la función tcrf_form_submit_link() se ejecuta cuando se acciona el botón asociado a ella.

En esta función se realiza la conexión con los servidores de Adele y se almacenan las variables obtenidas como respuesta. No se entrará en más detalle ya la acción se realiza exactamente igual que en el plugin de Joomla.

```
/**
 * Tcrf_module settings, Link Account submit actions.
 */
function tcrf_form_submit_link($form, &$form_state) {
  //Clear messages
  drupal_get_messages();

  //Connect to The Client Relations Factory
  $public_key = file_get_contents(drupal_get_path('module', 'tcrf') . '/assets/key/pub.key');

  $login_data = json_encode(array('username' => $form_state['values']['username'], 'password' => $form_state['values']['user_password']));

  openssl_public_encrypt($login_data, $login_data, $public_key);

  $scriplet_service_url = 'http://www.theclientrelationsfactory.com/services/scriptlet-generator/get-data.php';

  $options = array(
    'http' => array(
        'method'  => 'POST',
        'content' => $login_data,
        'header'=>  "Content-Type: application/json\r\n" .
                    "Accept: application/json\r\n"
        )
  );

   $context  = stream_context_create($options);
   $result   = file_get_contents($scriplet_service_url, false, $context);
   $response = json_decode($result);

   //Manage the response
   if (!$response)
      drupal_set_message(t('Connection with The Client Relations Factory failed, please retry in a few moments.'), $type = 'error');
   if (!$response->error && $response->found){
    variable_set('username', $form_state['values']['username']);
    variable_set('avatar_id', $response->avatar_id);
    variable_set('avatar_name', $response->avatar_name);
    variable_set('avatar_password', $response->avatar_password);

   } else {
      drupal_set_message(t($response->error_message), $type = 'error');
   }

}
```

Figura 46. Función tcrf_form_submit_link().

La función tcrf_form_submit_unlink() que se muestra a continuación elimina los datos de la cuenta de usuario y los parámetros del avatar obtenidos de la fuente externa. De esta manera, la cuenta queda desvinculada.

```
/**
 * Tcrf_module settings, Unlink Account submit actions.
 */
function tcrf_form_submit_unlink($form, &$form_state) {
  //Clear messages
  drupal_get_messages();

  //Delete account data
  variable_del('username');
  variable_del('avatar_id');
  variable_del('avatar_name');
  variable_del('avatar_password');
```

Figura 47. Función tcrf_form_submit_unlink().

Para terminar con la página de configuración, se establece la función asociada al formulario de configuración del avatar, en el que se crean las variables correspondientes con los valores obtenidos en la respuesta de los servidores de Adele.

```php
/**
 * Tcrf_module settings, Avatar settings submit actions.
 */
function tcrf_form_submit_avatarsettings($form, &$form_state) {
  //Clear messages
  drupal_get_messages();

  //Set Avatar Settings
  variable_set('window_size', $form_state['values']['window_size']);
  variable_set('top_bar_color', $form_state['values']['top_bar_color']);
  variable_set('start_button_text', $form_state['values']['start_button_text']);
  variable_set('send_button_text', $form_state['values']['send_button_text']);
}
```

Figura 48. Función tcrf_form_submit_avatarsettings().

Una vez terminada la página de configuración solo queda, en cada página, generar tanto el script de Fiona, como el bloque div que la contendrá y el botón para activarla, así como la hoja de estilo.

Esto se consigue con el hook:

```php
function tcrf_block_view($delta = '') {
  switch ($delta) {

    case 'tcrf':
      if (variable_get('avatar_name')){
        drupal_add_css(drupal_get_path('module', 'tcrf') . '/assets/css/buttonstyle.css', array('group' => CSS_DEFAULT, 'every_page' => TRUE));
        drupal_add_js('
          function load() {

            //Chat with me button
            var btn = document.createElement("button");
              var t = document.createTextNode("' . variable_get("start_button_text", "Chat with me!") . '");
              btn.appendChild(t);
              btn.setAttribute("onclick", "initFiona(this,usermail,usrid2,usrid1,avatarId)");
              btn.setAttribute("class", "fiona_button");

              document.body.appendChild(btn);

              //Fiona div
              var dv = document.createElement("div");
              dv.setAttribute("id", "fiona");
              dv.setAttribute("class", "fiona_main");
              dv.setAttribute("style", "display:none;");
              document.body.appendChild(dv);

          }

          window.onload = load;
          ', array('type' => 'inline', 'scope' => 'header', 'every_page' => TRUE ));

        drupal_add_js('


          usermail = "' . variable_get("username") . '";
          usrid2 = "' . md5(variable_get("avatar_password")) . '";
          usrid1 = "' . md5(variable_get("avatar_password") . variable_get("avatar_id")) . '";
          avatar_size = "' . strtolower(variable_get("window_size")) . '";
          avname ="' . variable_get("avatar_name") . '";
          avatarId ="' . variable_get("avatar_id") . '";
          show_pop_up = true;
          allow_camera = false;
          btnText = "' . variable_get("send_button_text", "Send") . '";
          dialogTopBarColor = "' . variable_get("top_bar_color") . '";
          (function() {
            var fi = document.createElement("script");
            fi.type = "text/javascript";
            fi.async = true;
            fi.src = ("http://sm.adelerobots.com/js/servicemng.js");
            var s = document.getElementsByTagName("script")[0];
            s.parentNode.insertBefore(fi, s);
          })();

          ', array('type' => 'inline', 'scope' => 'header'));
      }
  }
}
```

Figura 49. Función tcrf_form_view().

4.2.3 WordPress

4.2.3.1 Desarrollo

Al igual que ocurre en Drupal, en WordPress no es difícil tomar la decisión de qué tipo de extensión desarrollar, se implementará un Plugin.

4.2.3.1.1 Archivos del plugin

El directorio del plugin para WordPress, es algo más complejo que los vistos anteriormente:

- Archivo PHP principal del plugin que contendrá la funcionalidad del plugin (fionaPlugin.php).
- Archivo PHP que contiene la configuración del script de Fiona.
- Directorio *views* en el que se almacenarán los PHP de la vista de configuración (config-form.php) y de login (login-form.php).
- Directorio *assets,* que contiene los archivos "accesorios" al plugin (JavaScript, CSS…)
- Directorio *images*, que como su nombre indica, contiene las imágenes utilizadas en el plugin.

4.2.3.1.2 Enfoque

Al igual que se ha hecho en los gestores anteriores, se debe organizar el enfoque antes de comenzar la implementación:

- Intención: Integrar a Fiona con WordPress.
- Método: Mediante un plugin.
- Material necesario:
    - Script proporcionado por Adele robots (se modificará ligeramente).
    - Fichero PHP con la funcionalidad del plugin, nombrado fionaPlugin.php
    - Fichero PHP que modificará el script.
    - Ficheros PHP para la vista de los formularios de login y configuración del avatar
    - CSS para dar estilo al plugin, nombrado style.css
    - Clave pública. También hará falta un clave pública para conectarnos a los servidores de Adele Robots, se verá en el desarrollo de fionaPlugin.php

4.2.3.1.3 Implementación

Al igual que en el apartado correspondiente a Drupal, dado que la funcionalidad que se pretende conseguir con el desarrollo del plugin, es la misma que para el plugin de Joomla, se procede a explicar directamente el desarrollo de los ficheros del módulo.

4.2.3.1.3.1 fionaPlugin.php

El caso de WordPress vuelve a ser algo diferente al resto de gestores de contenido, en este caso en como el CMS ejecuta la funcionalidad de los plugin.

WordPress ejecuta el archivo del plugin en cada carga de página siempre y cuando dicho plugin esté activo. Debido a esto, "se debe" crear una función que englobe las funcionalidades del plugin y hacer que dicha función se ejecute.

```
tcrf_init();

/**
 * Enables admin menu for the plugin and if the plugin is enabled
 * adds the CSS and the JavaScript.
 */
function tcrf_init(){

    add_action('admin_menu', 'tcrf_admin_menu');

    $tcrf_user_params = get_option('tcrf_user_params');

    if ($tcrf_user_params['fiona_enabled']){

        add_action( 'wp_enqueue_scripts', 'tcrf_add_css' );
        add_action('wp_footer', 'tcrf_add_fiona_script');
    }
}
```

Figura 50. Función tcrf_init().

En esta primera función se utilizan los métodos proporcionados por WordPress para asociar eventos a funciones específicas. En este caso concreto, se asocian las funciones *tcrf_admin_menu(), tcrf_add_css()* y *tcrf_add_fiona_script()* a los eventos *admin_menu, wp_enqueue_scripts* y *wp_footer* respectivamente. Los dos últimos de estos eventos, están condicionados a que el parámetro fiona_enabled sea verdadero, lo cual indicaría que el avatar está activado. Esta sentencia condicional es utilizada para decidir si se muestra o no el botón de inicio del avatar y si se carga el script que contiene a Fiona.

La función *admin_menu()* sirve para configurar el título y el icono de la página de configuración. Dicho título e icono, gracias a la función *add_menu_page()* aparecerán en la barra lateral del panel de administración del CMS.

```php
function tcrf_admin_menu(){

    add_menu_page('Fiona Configuration',
        'TCRF Settings',
        'manage_options',
        __FILE__,
        'tcrf_config_view',
        plugins_url( '/images/favicon.png', __FILE__)
    );
}
```

Figura 51. Función tcrf_admin_view().

En dicha función se establece como 5º parámetro, la función que generará la vista de la página de configuración, es este caso *tcrf_config_view():*

```php
function tcrf_config_view(){

    //If user is doing somethig
    if (isset($_POST['tcrf_action'])){

        //If user is logged
        if (get_option('tcrf_user_params')){

            switch($_POST['tcrf_action']){

                case 'enable-fiona': {
                    tcrf_enable_fiona();
                    break;
                }

                case 'disable-fiona': {
                    tcrf_disable_fiona();
                    break;
                }
                case 'logout': {
                    tcrf_logout();
                    break;
                }

                default: {
                    show_config_view();
                    break;
                }
            }
        }
        //If the user is not logged
        else if ($_POST['tcrf_action'] == 'login') {
            tcrf_login();
        } else {
            tcrf_show_login_view();
        }
    } else {
        //If user is logged
        if (get_option('tcrf_user_params')){
            tcrf_show_config_view();
        }
        //If the user is not logged
        else{
            tcrf_show_login_view();
        }
    }
}
```

Figura 52. Función tcrf_config_view().

Como puede observarse en la imagen anterior, la vista mostrada dependerá de la acción que haya realizado el usuario (se verá más adelante como se establece el parámetro que indica dicha acción). Dicha vista se configurará según la ejecución de las funciones establecidas:

- Si el usuario no ha realizado ninguna acción:
    - Si el usuario está logeado:
        - *tcrf_show_config_view()*
    - Si el usuario no está logeado
        - *tcrf_show_login_view()*
- Si el usuario ha realizado una acción
    - Si el usuario está logeado:

        - Si ha habilitado el avatar: *tcrf_enable_fiona()*
        - Si ha deshabilitado el avatar: *tcrf_disable_fiona()*
        - Si ha hecho logout: *tcrf_logout()*
        - Por defecto: *show_config_view()*
    - Si el usuario no está logeado
        - Si se ha hecho login: *tcrf_login()*
        - Por defecto: *tcrf_show_login_view()*

La función *tcrf_login(),* establece la conexión con los servidores de Adele de la misma manera que en los otros dos Gestores de Contenido y recibe los parámetros de configuración del avatar.

```php
function tcrf_login(){

    if (!($username = filter_var($_POST['tcrf_username'], FILTER_VALIDATE_EMAIL))) {
        tcrf_show_login_view('E-mail is invalid!');
        exit();
    }

    $password = $_POST["tcrf_password"];

     //Connect to The Client Relations Factory
    $public_key = file_get_contents(plugin_dir_path(__FILE__) . 'assets/key/pub.key');

    $login_data = json_encode(array('username' => $username, 'password' => $password));

    openssl_public_encrypt($login_data, $login_data, $public_key);

    $scriplet_service_url = 'http://www.theclientrelationsfactory.com/services/scriptlet-generator/get-data.php';

    $result = wp_remote_post($scriplet_service_url,
        array(
            'method' => 'POST',
            'timeout' => 45,
            'redirection' => 5,
            'httpversion' => '1.0',
            'content-type' => 'application/octet-stream',
            'blocking' => true,
            'headers' => array(),
            'body' => $login_data,
            'cookies' => array()
        )
    );

    $response = json_decode($result['body']);

    //Manage the response
    if (!$response) {
        $response = (object) array(
            'error' => true,
            'error_message' => "Connection with The Client Relations Factory failed, please retry in a few moments."
        );
    }
    if (!$response->error && $response->found) {
        update_option('tcrf_user_params',
            array(
                'username' => $username,
                'avatar_id' => $response->avatar_id,
                'avatar_name' => $response->avatar_name,
                'avatar_password' => $response->avatar_password,
                'fiona_enabled' => false,
                'window_size' => 'Big',
                'allow_camera' => false,
                'button_text' => 'Chat with me!'
            )
        );
        tcrf_show_config_view();
    } else {
        tcrf_show_login_view($response->error_message);
    }
}
```

Figura 53. Función tcrf_login().

Si todo ha ido correcto se ejecuta *tcrf_show_config_view(),* en caso contrario, *tcrf_show_login_view()* enviando como parámetro el mensaje de error obtenido.

La función *tcrf_logout(),* borra los parámetros del usuario y a continuación ejecuta *tcrf_show_login_view().*

```
function tcrf_logout(){

    delete_option('tcrf_user_params');
    tcrf_show_login_view();
}
```

Figura 54. Función tcrf_logout().

La función *enable_fiona()*, recoge los parámetros introducidos por el usuario y los almacena en la memoria del CMS.

```
function tcrf_enable_fiona(){

    $tcrf_user_params = get_option('tcrf_user_params');
    $tcrf_user_params['fiona_enabled'] = true;

    $tcrf_user_params['start_button_text'] = $_POST['tcrf_start_button_text'];
    $tcrf_user_params['chat_button_text'] = $_POST['tcrf_chat_button_text'];
    $tcrf_user_params['dialog_top_bar_color'] = $_POST['tcrf_top_bar_color'];

    $tcrf_user_params['window_size'] = $_POST['tcrf_window_size'];
    $tcrf_user_params['allow_camera'] = isset($_POST['tcrf_allow_camera']) ? true : false;

    update_option('tcrf_user_params', $tcrf_user_params);

    tcrf_show_config_view();
}
```

Figura 55. Función tcrf_enable_fiona().

La función *disable_fiona()* tiene el comportamiento contrario a la función anterior, borra los parámetros del usuario de la memoria del CMS.

```
function tcrf_disable_fiona(){

    $tcrf_user_params = get_option('tcrf_user_params');
    $tcrf_user_params['fiona_enabled'] = false;

    update_option('tcrf_user_params', $tcrf_user_params);

    tcrf_show_config_view();
}
```

Figura 56. Función tcrf_disable_fiona().

Las funciones *show_login_view()* y *show_config_view()*, se encargan de mostrar las vistas de login y configuración respectivamente.

```
function tcrf_show_login_view($error_message = false){

    include(plugin_dir_path(__FILE__) . '/views/login-form.php');
}

function tcrf_show_config_view($error_message = false){

    $tcrf_user_params = get_option('tcrf_user_params');

    include(plugin_dir_path(__FILE__) . '/views/config-form.php');

}
```

Figura 57. Funciones tcrf_show_login_view() y tcrf_show_config_view().

Para limpiar un poco el código, y separar las diferentes funcionalidades del plugin, estas dos vistas se han implementado en ficheros aparte, incluidos gracias a la directiva *include()*.

- Vista de login: archivo login-form.php
- Vista de configuración config-form.php

Por último, las funciones tcrf_add_css() y tcrf_add_fiona_script() incluyen la CSS y el script que contiene a Fiona (fionaScript.php) respectivamente.

```
function tcrf_add_css(){

    wp_enqueue_style('tcrf_stylesheet', plugins_url('/assets/css/fiona-avatar.css', __FILE__));
}

function tcrf_add_fiona_script(){

    include(plugin_dir_path(__FILE__) . '/fionaScript.php');
}
```

Figura 58. Funciones tcrf_add_css() y tcrf_add_fiona_script().

4.2.3.1.3.2 login-form.php

Mediante este fichero se genera el código HTML que creará y dotará de funcionalidad la vista que el usuario debe utilizar para introducir sus credenciales de The Client Relations Factory y conectarse con los servidores de Adele.

En la primera parte del código, se añade la CSS que dará estilo a esta vista.

```
<?php
    wp_enqueue_style('tcrf_login_form_stylesheet', plugins_url('../assets/css/login-form.css', __FILE__));
?>
```

Figura 59. Inclusión de CSS.

A continuación, se genera el div que contendrá la vista y se establece un encabezado.

```
<div class="wrap">
<h2><a href="http://www.theclientrelationsfactory.com" target="_blank"> The Client Relations Factory </a></h2>
```

Figura 60. que contendrá la vista.

Justo después, combinando PHP y HTML, se establece un mensaje de error si a la función que incluye el archivo, y por tanto al archivo PHP, se le pasó como parámetro una variable que contenga dicho mensaje.

```php
<?php
    if ($error_message)
    {
?>
    <div id="login-error">
        <div class="alert-box error-box"><span>error: </span>
        <?php
            if ($error_message)
            {
                echo $error_message;
            }
            else
            {
                echo "Cannot connect with The Client Relations Factory, please try again in a few minutes. If problem persists, please contact us in support@theclientrelationsfactory.com";
            }
        ?></div>
    </div>
<?php
    }
?>
```

Figura 61. Mensaje de error.

Por último, se genera el formulario de entrada de datos del usuario.

```
<form method="post">
    <input type="hidden" name="tcrf_action" value="login">
    <table class="form-table tcrf-form-table">
        <tr valign="top">
        <th scope="row">Username</th>
        <td><input required type="email" name="tcrf_username" value="" /></td>
        </tr>

        <tr valign="top">
        <th scope="row">Password</th>
        <td><input required type="password" name="tcrf_password" value="" /></td>
        </tr>
        <tr>
            <th scope="row">

            </td>
            <td>
                <input class="button button-primary" type="submit" value="Link Account!" title="Link your account and embed your Virtual Robot!"/>
            </td>
        </tr>
    </table>

<div class="tcrf-link">Please Sign In with your The Client Relations Factory credentials. If you don't have an account yet... <br />
    <a href="http://www.theclientrelationsfactory.com/#price" target="_blank" title="Get your account now!"><p class="tcrf-register-now"><strong>Register now!</strong></p></a>
</div>

</form>
</div>
```

Figura 62. Formulario datos usuario.

En este formulario, se establece como parámetro oculto la acción que está realizando el usuario a la hora de enviarlo. Esta es la manera en la que se genera el parámetro que utiliza la función *tcrf_config_view()* para "saber" que acción está realizando el usuario.

4.2.3.1.3.3 config-form.php

En la primera parte del código de este archivo, se añade la librería, utilizada también en el módulo de Drupal, que permite generar un selector de color. También se recogen de la memoria del CMS los parámetros del usuario.

```php
<?php

    wp_enqueue_script('tcrf_jscolor', plugins_url('../assets/js/jscolor/jscolor.js', __FILE__));

    $tcrf_user_params = get_option('tcrf_user_params');

?>
```

Figura 63. Inclusión de jscolor.

A continuación, al igual que en la vista login-form, se crea el div que contendrá la vista, se establece un encabezado y se muestra un mensaje indicando la cuenta con la que el usuario está conectado a los servidores de Adele.

```
<div class="wrap">
<h2><a href="http://www.theclientrelationsfactory.com" target="_blank"> The Client Relations Factory </a></h2>

<p>Your linked account is <?php echo '<strong>' . $tcrf_user_params['username'] . '</strong>'; ?></p>
```

Figura 64. que contendrá la vista con su encabezado.

Posteriormente, se genera el formulario que permitirá al usuario configurar el Asistente Virtual. Cabe destacar que por velocidad en el desarrollo y dado que el desarrollo del HTML y las CSS que forman estas ampliaciones no es el objetivo principal de este proyecto, y aunque es tremendamente desaconsejable, se ha utilizado una tabla para maquetar dicho formulario.

```
<table class="form-table">
    <form method="post" id="fiona_settings">
        <input type="hidden" name="tcrf_action" value="enable-fiona">
        <tr valign="top">
            <th scope="row">Start Button Text</th>
            <td>
                <input type="text" name="tcrf_start_button_text" minlength="1" maxlength="15"
                    value="<?php echo isset($tcrf_user_params['start_button_text']) ? $tcrf_user_params['start_button_text'] : "Chat with me!" ?>"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " readonly ";
                ?>
                title="Defines the text that will be displayed inside Virtual Robot's start button"
                >
            </td>
        </tr>
        <tr valign="top">
            <th scope="row">Chat Button Text</th>
            <td>
                <input type="text" name="tcrf_chat_button_text" minlength="1" maxlength="10"
                    value="<?php echo isset($tcrf_user_params['chat_button_text']) ? $tcrf_user_params['chat_button_text'] : "Send" ?>"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " readonly ";
                ?>
                title="Defines the text that will be displayed into the Send button of Virtual Robot chat dialog"
                >
            </td>
        </tr>
        <tr valign="top">
            <th scope="row">Avatar Size</th>
            <td>
                <select name="tcrf_window_size"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " disabled ";
                ?>
                title="Sets the size of your Virtual Robot display frame"
                >
                    <option
                    <?php
                        if (strcasecmp($tcrf_user_params['window_size'], "Small") == 0) echo " selected ";
                    ?>
                    value="Small">Small
                    </option>
                    <option
                    <?php
                        if (strcasecmp($tcrf_user_params['window_size'], "Big") == 0 || !isset($tcrf_user_params['window_size']))
                            echo " selected ";
                    ?>
                    value="Big">Big
                    </option>
                </select>
            </td>
        </tr>
        <tr valign="top">
            <th scope="row">Top bar widget color</th>
            <td>
                <input type="text" name="tcrf_top_bar_color"
                    <?php if (isset($tcrf_user_params['dialog_top_bar_color']))
                        echo 'value="' . $tcrf_user_params['dialog_top_bar_color'] . '"';
                    ?>
                    class="color {pickerFaceColor:'transparent', pickerFace:3, pickerBorder:0, pickerInsetColor:'black', slider:false, pickerPosition:'right', hash:true}"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " disabled ";
                ?>
                title="Sets the color of your Virtual Robot's widget dialog top bar"
                >
            </td>
        </tr>
    </form>
```

Figura 65. Formulario datos Avatar.

Puede observarse en la 3ª línea de este fragmento de código el campo oculto que establece la acción que está realizando el usuario. El resto de bloques, se encargan de generar los campos que configuran los parámetros del Avatar. A diferencia de Joomla y Drupal, se realiza toda la generación de elementos puramente con HTML. Se han utilizado pequeños bloques de PHP para generar atributos de las etiquetas HTML condicionados a valores de los parámetros del usuario recogidos de la memoria del CMS.

Como último bloque de esta vista se establecen los botones utilizados para habilitar/deshabilitar el Avatar y para desvincular la cuenta de usuario.

```
        <tr valign="top">
            <th>
                <form method="post">
                    <input type="hidden" name="tcrf_action" value="logout">
                    <input class="button button-primary button-no-bold" type="submit" value="Unlink Account" title="Detach your account from the plugin"/>
                </form>
            </th>
            <?php
                if (!$tcrf_user_params['fiona_enabled'])
                {
            ?>
            <td>
                <button class="button button-primary" classe="" title="Enables your Virtual Robot integration for this site" onclick='document.getElementById("fiona_settings").submit();'>Enable Virtual Robot!</button>
            </td>
            <?php
                }
                else
                {
            ?>
            <td>
                <form method="post">
                    <input type="hidden" name="tcrf_action" value="disable-fiona">
                    <input class="button button-primary" type="submit" value="Disable Virtual Robot" title="Disables your Virtual Robot's integration for this site"/>
                </form>
            </td>
            <?php
            }
            ?>
        </tr>
    </table>
</div>
```

Figura 66. Botones.

4.2.3.1.3.4 fionaScript.php

Mediante este archivo se genera el script que contiene los parámetros de configuración de Fiona y contiene la invocación al script, ubicado en los servidores de la compañía Adele Robots, que genera a Fiona.

```php
<?php

$tcrf_user_params = get_option('tcrf_user_params');

echo '
    <div id="fiona" class="fiona_main" style="display:none;">
        </div>
        <div id="fiona-button">
        <!-- Position bottom-right -->
        <span class="span_wrap">
            <span class="span_wrapped" id="starttalk" onclick="initFiona(this, usermail, usrid2, usrid1, avatarId)">
         ' . $tcrf_user_params['start_button_text'] . '
        </span>
        </span>
    </div>

    <script type="text/javascript"><!--//--><![CDATA[//><!--
            usermail ="' . $tcrf_user_params['username'] . '";
            usrid2 = "' . md5($tcrf_user_params['avatar_password']) . '";
            usrid1 = "' . md5($tcrf_user_params['username'] . $ar_tcrf_account_data['avatar_id']) . '";
            avatar_size = "' . strtolower($tcrf_user_params['window_size']) . '";
            avname = "' . $tcrf_user_params['avatar_name'] . '";
            avatarId = "' . $tcrf_user_params['avatar_id'] . '";
            dialogTopBarColor = "' . $tcrf_user_params['dialog_top_bar_color'] . '";
            btnText = "' . $tcrf_user_params['chat_button_text'] . '";

            show_pop_up = true;

            allow_camera = ' . ($tcrf_user_params['allow_camera'] ? 'true' : 'false') . ';

            (function() {
             var fi = document.createElement("script");
             fi.type = "text/javascript";
             fi.async = true;
             fi.src = "http://sm.adelerobots.com/js/servicemng.js";
             var s = document.getElementsByTagName("script")[0];
             s.parentNode.insertBefore(fi, s);
            })();
        //--><!]]></script>';
?>
```

Figura 67. JavaScript necesario para Fiona.

4.2.3.1.3.5 uninstall.php

A diferencia de Joomla y Drupal, WordPress recomienda la inclusión de un archivo llamado uninstall.php, que es invocado durante el proceso de desinstalación del plugin.

En el caso concreto de este plugin, en la llamada a este archivo, se eliminan los parámetros de usuario.

```php
<?php

if ( !defined( 'WP_UNINSTALL_PLUGIN' ) )
{
    exit();
}

delete_option('tcrf_user_params');

?>
```

Figura 68. Unistall.php.

Capítulo 5. Pruebas

A continuación se exponen las pruebas de funcionalidad de las ampliaciones realizadas. Dichas pruebas se han realizado en los navegadores Google Chrome y Mozilla Firefox con idéntico resultado. Inicialmente se contemplaban, además, pruebas en Safari, pero debido a que los asistentes virtuales de Adele Robots no son 100% compatibles con este navegador, se ha prescindido de las mismas. Además, como se puede observar en la gráfica que se expone a continuación, Google Chrome y Mozilla Firefox representan cerca del 70% de la cuota de mercado de los navegadores en los últimos 6 meses.

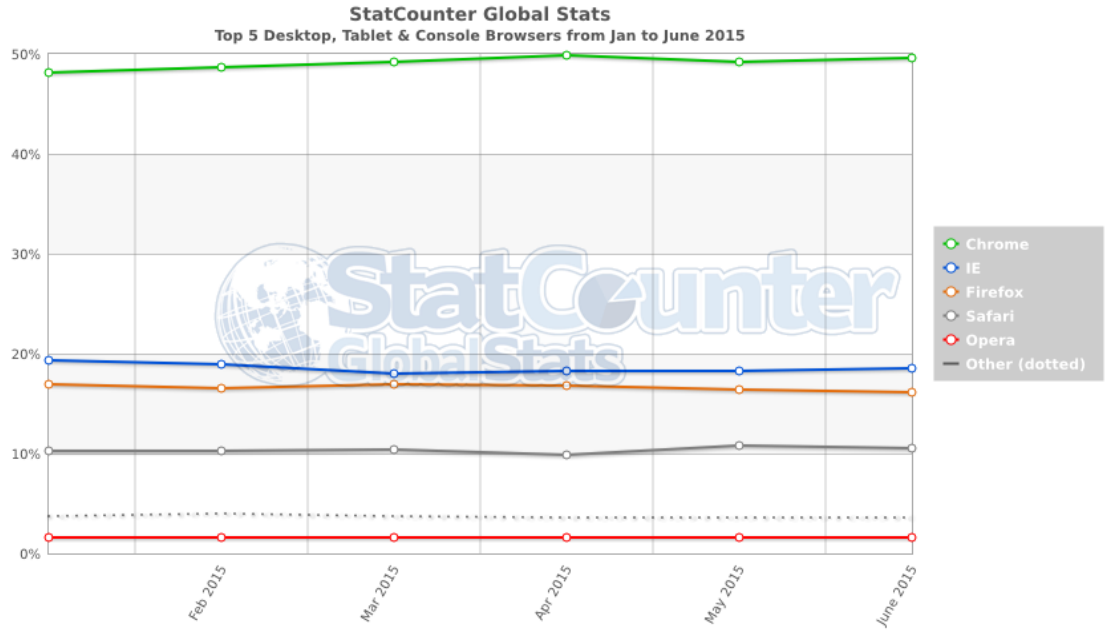


Figura 69. Comparativa navegadores.

5.1 Pruebas en Joomla

Aplicación: Plugin de Fiona para Joomla

Versión: 0.1

Entorno de pruebas: Instalación local de Joomla 3.3.6 en Mac OSX 10.10.3

5.1.1 Pruebas generales

Prueba: Instalación

Descripción: Instalación del plugin utilizando el Joomla Extension Manager. La instalación se realiza mediante un archivo .Zip que contiene el código.

Resultado: Instalación correcta.

Valoración: Positiva.

Prueba: Activación

Descripción: Activación del plugin.

Resultado: Activación correcta.

Valoración: Positiva.

5.1.2 Configuración

Prueba: Usuario y contraseña válidos.

Descripción: Prueba de funcionamiento con configuración básica (valores por defecto) y usuario y contraseña válidos.

Resultado: Fiona carga correctamente.

Valoración: Positiva.

Prueba: Usuario y contraseña no válidos.

Descripción: Prueba de funcionamiento con configuración básica (valores por defecto) y usuario y contraseña no válidos.

Resultado: El botón para iniciar Fiona no se muestra en pantalla.

Valoración: Positiva.

Observaciones: Al intentar establecer una conexión con The Client Relations Factory con una credenciales erróneas, el servicio devuelve un mensaje de error. Dicho mensaje se procesa en el código, el cual no genera el botón para iniciar Fiona.

Prueba: Tamaño Fiona.

Descripción: Se comprueba el campo que permite cambiar el tamaño de la venta en la que se muestra Fiona.

Resultado: Fiona carga correctamente con el tamaño seleccionado.

Valoración: Positiva.

Observaciones: Durante la carga del componente, el botón para enviar mensajes a Fiona y el campo de texto aparece desajustado, pero se corrige en el momento en el que carga el avatar.

Prueba: Color de la barra superior de la ventana de Fiona.

Descripción: Prueba de funcionamiento del selector de color para la barra superior de la ventana que contiene a Fiona.

Resultado: El color cambia correctamente según el seleccionado.

Valoración: Positiva.

Observaciones: Quizás un menú desplegable encajaría mejor en el diseño de la pantalla de configuración que radio buttons. Se ha hecho la modificación.

Prueba: Cambio del texto del botón enviar.

Descripción: Prueba de funcionamiento del campo que permite personalizar el texto que se muestra en el botón utilizado para enviar mensajes a Fiona.

Resultado: El texto mostrado es el introducido en la página de configuración.

Valoración: Positiva.

Prueba: Cambio del texto del botón start robot.

Descripción: Prueba de funcionamiento del campo que permite personalizar el texto que se muestra en el botón utilizado para iniciar la comunicación con el Robot Virtual.

Resultado: El texto mostrado es el introducido en la página de configuración.

Valoración: Positiva.

5.1.3 Conexión

Prueba: Cambios en The Client

Descripción: Prueba que la conexión con The Client funcione correctamente cambiando el nombre del Robot Virtual y el color de su camisa.

Resultado: Los cambios hechos en The Client se reflejan correctamente en el plugin.

Valoración: Positiva.

Observaciones: La interacción con The Client no resultó muy intuitiva, ya que tras guardar los cambios las primeras veces no se reflejaban en el resultado. El fallo era no publicar los cambios mediante el botón Publish del sitio.

5.2 Pruebas en Drupal

Aplicación: Módulo de Fiona para Drupal

Versión: 0.1

Entorno de pruebas: Instalación local de Drupal 7.37. en Mac OSX 10.10.3

5.2.1 Pruebas generales

Prueba: Instalación

Descripción: Instalación del plugin utilizando la herramienta de instalación automática. La instalación se realiza mediante un archivo .Zip que contiene el código.

Resultado: Instalación fallida en un primer intento y exitosa en el segundo.

Valoración: Negativa.

Prueba: Instalación manual

Descripción: Instalación del plugin manualmente. La instalación se realiza copiando el código del plugin al directorio correspondiente.

Resultado: Instalación correcta.

Valoración: Positiva.

Prueba: Activación

Descripción: Activación del plugin.

Resultado: Activación correcta.

Valoración: Positiva.

Prueba: Desactivación

Descripción: Desactivación del plugin.

Resultado: Desactivación correcta.

Valoración: Positiva.

5.2.2 Configuración

Prueba: Usuario y contraseña válidos.

Descripción: Prueba de funcionamiento con usuario y contraseña válidos.

Resultado: Se muestra el aviso "Connected with *NombreDelAvatar*", se muestran las opciones de configuración y el botón para iniciar Fiona se muestra en pantalla.

Valoración: Positiva.

Prueba: Usuario y contraseña no válidos.

Descripción: Prueba de funcionamiento con configuración usuario y contraseña no válidos.

Resultado: El botón para iniciar Fiona no se muestra en pantalla y se muestra el mensaje "*Not Connected with The Client Relations Factory*".

Valoración: Positiva.

Observaciones: Al intentar establecer una conexión con The Client Relations Factory con una credenciales erróneas, el servicio devuelve un mensaje de error. Dicho mensaje se procesa en el código, el cual no genera el botón para iniciar Fiona.

Prueba: Tamaño Fiona.

Descripción: Se comprueba el campo que permite cambiar el tamaño de la venta en la que se muestra Fiona.

Resultado: Fiona carga correctamente con el tamaño seleccionado.

Valoración: Positiva.

Observaciones: Durante la carga del componente, el botón para enviar mensajes a Fiona y el campo de texto aparece desajustado, pero se corrige en el momento en el que carga el avatar.

Prueba: Color de la barra superior de la ventana de Fiona.

Descripción: Prueba de funcionamiento del selector de color para la barra superior de la ventana que contiene a Fiona.

Resultado: La librería jscolor muestra correctamente el selector de color y el color cambia correctamente según el seleccionado.

Valoración: Positiva.

Prueba: Cambio del texto del botón enviar.

Descripción: Prueba de funcionamiento del campo que permite personalizar el texto que se muestra en el botón utilizado para enviar mensajes a Fiona.

Resultado: El texto mostrado es el introducido en la página de configuración.

Valoración: Positiva.

Prueba: Cambio del texto del botón start robot.

Descripción: Prueba de funcionamiento del campo que permite personalizar el texto que se muestra en el botón utilizado para iniciar la comunicación con el Robot Virtual.

Resultado: El texto mostrado es el introducido en la página de configuración.

Valoración: Positiva.

5.2.3 Conexión

Prueba: Cambios en The Client

Descripción: Prueba que la conexión con The Client funcione correctamente cambiando el nombre del Robot Virtual y el color de su camisa.

Resultado: Los cambios hechos en The Client se reflejan correctamente en el plugin.

Valoración: Positiva.

Observaciones: La interacción con The Client no resultó muy intuitiva, ya que tras guardar los cambios las primeras veces no se reflejaban en el resultado. El fallo era no publicar los cambios mediante el botón Publish del sitio.

5.3 Pruebas en WordPress

Aplicación: Plugin de Fiona para WordPress

Versión: 1.0

Entorno de pruebas: Instalación local de WordPress 4.2.2 en Mac OSX 10.10.3

5.3.1 Pruebas generales

Prueba: Instalación

Descripción: Instalación del plugin utilizando la herramienta de instalación de plugins. La instalación se realiza mediante un archivo .Zip que contiene el código.

Resultado: Instalación correcta.

Valoración: Positiva.

Prueba: Activación

Descripción: Activación del plugin.

Resultado: Activación correcta.

Valoración: Positiva.

5.3.2 Configuración

Prueba: Usuario y contraseña válidos.

Descripción: Prueba de validación en el servidor externo con credenciales correctas.

Resultado: La validación se realiza correctamente y aparece la posibilidad de habilitar el avatar.

Valoración: Positiva.

Prueba: Usuario y contraseña no válidos.

Descripción: Prueba de validación en el servidor externo con credenciales no válidas.

Resultado: Se muestra un mensaje de error y aparece de nuevo la pantalla de login.

Valoración: Positiva.

Prueba: Habilitar Fiona.

Descripción: Se comprueba la funcionalidad del botón *Enable Virtual Robot!*.

Resultado: Se desactiva la edición de los parámetros de configuración del robot y se muestra el botón Chat with me! en el blog.

Valoración: Positiva.

Prueba: Deshabilitar Fiona.

Descripción: Se comprueba la funcionalidad del botón *Disable Virtual Robot!*.

Resultado: Se activa la edición de los parámetros de configuración del robot y se elimina el botón Chat with me! del blog.

Valoración: Positiva.

Prueba: Tamaño Fiona.

Descripción: Se comprueba el campo que permite cambiar el tamaño de la venta en la que se muestra Fiona.

Resultado: Fiona carga correctamente con el tamaño seleccionado.

Valoración: Positiva.

Observaciones: Durante la carga del componente, el botón para enviar mensajes a Fiona y el campo de texto aparecen desajustados, pero se corrige en el momento en el que carga el avatar.

Prueba: Color de la barra superior de la ventana de Fiona.

Descripción: Prueba de funcionamiento del selector de color para la barra superior de la ventana que contiene a Fiona.

Resultado: La librería jscolor muestra un cuadro de selección de color y, una vez habilitado de nuevo el avatar, el color cambia correctamente según el seleccionado.

Valoración: Positiva.

Prueba: Cambio del texto del botón enviar.

Descripción: Prueba de funcionamiento del campo que permite personalizar el texto que se muestra en el botón utilizado para enviar mensajes a Fiona.

Resultado: El texto mostrado es el introducido en la página de configuración.

Valoración: Positiva.

Prueba: Cambio del texto del botón start robot.

Descripción: Prueba de funcionamiento del campo que permite personalizar el texto que se muestra en el botón utilizado para iniciar la comunicación con el Robot Virtual.

Resultado: El texto mostrado es el introducido en la página de configuración.

Valoración: Positiva.

5.3.3 Conexión

Prueba: Personalización del aspecto del avatar en The Client

Descripción: Prueba que la conexión con The Client funcione correctamente cambiando el color de la camisa del Asistente Virtual.

Resultado: Los cambios hechos en The Client se reflejan correctamente en el plugin.

Valoración: Positiva.

Prueba: Configuración del conocimiento en The Client

Descripción: Prueba que la conexión con The Client funcione correctamente aportando conocimiento al avatar.

Resultado: El avatar responde con las respuestas predefinidas a las preguntas registrados en su base de conocimiento a través del nuevo editor de AIML

Valoración: Positiva.

Capítulo 6. Comparativa y métrica

Se ha hablado de ampliaciones, refiriéndose a módulos, plugins, extensiones… pero para esta comparativa se va a establecer de una manera un poco más concisa que es, a vista del desarrollador, una ampliación.

Se ha establecido, basado en la experiencia adquirida durante el Máster en Ingeniería Web y sobre todo, durante la realización de este Trabajo Final de Máster, que se puede denominar ampliación, desde el punto de vista de este proyecto, y más concretamente refiriéndose a la funcionalidad de los Gestores de Contenidos Web, a *"todo artefacto que permite aumentar la funcionalidad de un Gestor de Contenidos, por medio de inclusión de código de terceros en los mismos"*.

Por tanto, quedan exentos de esta definición comentarios a artículos, artículos en sí mismos…, es decir, ampliaciones de la información contenida en un gestor de contenidos. También quedan fuera de la definición actualizaciones del propio CMS, así como actualizaciones de seguridad.

Una vez que se ha establecido la definición de ampliación, se establecerán dos marcos diferentes para la comparativa, el primero desde el punto de vista del administrador del CMS, y el segundo, desde el punto de vista del desarrollador.

6.1 Comparativa y métrica para administradores del CMS

Antes de comenzar con la comparativa, y por supuesto, antes de intentar definir una métrica, hay que establecer con claridad dentro de que contexto se considera a una persona como administrado de un Gestor de Contenidos.

Como se ha visto en apartados anteriores, se han desarrollado extensiones para Joomla, Drupal y WordPress. Se considera administrador al encargado de gestionar el contenido de un CMS, y por tanto, destinatario final de estas extensiones. La funcionalidad del Gestor de Contenidos, relacionada con las extensiones, que afecta a este administrador es la instalación y configuración de las mismas, por lo que será lo que valorará esta métrica, siendo 0 la puntuación mínima y 5 la máxima.

Criterio	Puntuación			Observaciones
	Joomla	Drupal	WordPress	
¿Existe documentación acerca de la instalación de extensiones?				
¿Está disponible en varios				

idiomas?				
¿Es fácilmente accesible?				
¿Describe con claridad la instalación?				
¿Posee algún sistema de instalación automática o se debe realizar de forma manual? 0 manual, 5 automática				
Una vez instalada la extensión ¿En fácil de activar y configurar?				

6.2 Comparativa y métrica para desarrolladores

Resulta bastante frustrante como desarrollador, es decir, como aquella persona que "fabrica" las ampliaciones estudiadas, elegir un enfoque para un proyecto y unas herramientas para llevarlo a cabo, y en mitad del proceso de desarrollo darse cuenta de que la elección no ha sido correcta. Esta comparativa y estas métricas pretenden ayudar al desarrollador a tomar esta decisión de una manera mejor cimentada, y que se traduzca en una mejor gestión de su proyecto.

Una primera aproximación a la creación de una métrica para el correcto desarrollo y análisis de un sitio web, en este caso concreto, para una funcionalidad específica de un sitio web, es basarse en la ya conocida guía de usabilidad de Yusef Hassan Montero, que establece una serie de criterios que analizar a la hora de evaluar la usabilidad de un sitio web.

Parece pues, bastante interesante tratar de adaptar dicha guía, no a la usabilidad de un sitio web, si no a la forma de utilizar *o usar* los métodos de desarrollo y las herramientas proporcionadas por los Gestores de Contenidos para el desarrollo de ampliaciones.

6.2.1 Evaluación de la documentación

Es importantísimo para un desarrollador externo, que no haya estado involucrado en el desarrollo de una herramienta o que nunca haya trabajado con ella, como es el caso de este proyecto, encontrar documentación específica y fácilmente accesible de aquello que pretende desarrollar. Es inútil para un desarrollador que, poniendo de ejemplo la materia de este trabajo, un Gestor de Contenidos disponga de unas herramientas extraordinarias para la gestión de eventos y la publicación de contenido si no existe documentación sobre su uso o esta documentación no es fácilmente accesible.

De la evaluación de diferentes apartados concernientes a la documentación surge la métrica que se muestra a continuación, siendo 0 la puntuación mínima y 5 la máxima.

Además, resulta interesante, ya que la mayoría de puntos a analizar no son tan sencillos como una simple puntuación numérica, que en caso que querer resaltar algo, se incluya información adicional en el apartado de observaciones.

Criterio	Puntuación			Observaciones
	Joomla	Drupal	WordPress	
¿Existe documentación acerca del CMS?				
¿Existe documentación acerca del sistema de ampliaciones del CMS?				
¿Es fácilmente accesible?				
¿Está disponible en varios				

idiomas?				
¿Es concisa? ¿Describe con detenimiento cada método de ampliación, para qué sirve y cómo utilizarlo?				
Una vez elegido el método a desarrollar ¿Existe información suficiente para el desarrollo sin recurrir a fuentes externas?				
¿Está dicha información bien organizada? ¿Dispone de un índice claro o un mapa web desde el que tener una visión global de toda la disponible?				
¿Dispone de tutoriales para la creación de ampliaciones básicas?				
¿Están los tutoriales actualizados a las últimas versiones?				
¿Dispone de tutoriales para la creación de ampliaciones avanzadas?				
¿Existe documentación de otras fuentes?				

6.2.2 Evaluación de la personalización

En el desarrollo de un plugin como este, es necesario obtener datos del usuario tanto para poder identificarse en el sistema externo (The Client Relations Factory de Adele Robots) como para personalizar ciertos aspectos del Robot Virtual.

Se ha visto que los tres Gestores de Contenidos permiten crear de manera más o menos sencilla páginas de configuración para el plugin, las cuales pueden ser utilizadas para recoger los datos del usuario necesarios.

Un aspecto importante a valorar en este caso es la relación entre la complejidad del desarrollo de dichas páginas, y la funcionalidad que ofrecen.

Para poder realizar una puntuación realista de la complejidad, se deben establecer primero los conocimientos del desarrollador al que va dirigido, no resulta igual de complejo desarrollar un código HTML para un desarrollador de Frontend que para un desarrollador especializado en Bases de Datos.

El desarrollo de plugins para Gestores de Contenidos suele, o debería, estar en las competencias de un desarrollador web, o tal vez un diseñador. Suponemos por tanto, que posee conocimientos de PHP, HTML, CSS y JavaScript.

Criterio	Puntuación			Observaciones
	Joomla	Drupal	WordPress	
¿Resulta sencillo crear una página básica de configuración?				
¿Resulta sencillo ampliar y/o modificar dicha página?				
¿Es sencillo de configurar sin conocimientos técnicos adicionales?				
¿Es posible proporcionar diferentes campos de entrada al usuario?				
¿Es configurable la disposición de los elementos en la página?				
¿Permite la inclusión de código HTML externo?				
¿Permite la inclusión de JavaScript?				

6.2.3 Evaluación de los métodos de desarrollo

Los tres CMS utilizados en este proyecto tienen muchas características en común, una de las cuales es casi un requisito fundamental en la Ingeniería Web de hoy día y que en el desarrollo hemos pasado un poco por alto, están basados en patrones de diseño.

Más concretamente, estos tres Gestores de Contenidos utilizan patrones de diseño Event-Driven, es decir, impulsados por eventos. En otras palabras, los tres Gestores utilizan eventos para manejar la funcionalidad de los plugin.

Por ejemplo, se ha visto que en Drupal:

```
/**
 * Implements hook_block_info().
 */
function tcrf_block_info() {
  $blocks['tcrf'] = array(
    // The name that will appear in the block list.
    'info' => t('The Client Relations Factory'),
    // Default setting.
    'cache' => DRUPAL_CACHE_PER_ROLE,
  );
  return $blocks;
}
```

Figura 70. Función tcrf_block_info().

La función tcrf_block_info(), por medio de una convención de nombres, implementa hook_block_info(), de manera que cuando se "activa" el evento block_info, se ejecuta la funcionalidad establecida dentro de la función.

La comparativa de esta funcionalidad resulta un poco más abstracta ya que, al nivel estudiado, la única ventaja/inconveniente que presenta el escoger un CMS u otro desde el punto de vista de la gestión de eventos que proporciona es la preferencia personal del desarrollador. La complejidad y la funcionalidad que aportan es muy similar en todos ellos.

Aportaremos pues, una métrica, evaluando esta característica intentando, no tanto dilucidar si un sistema es mejor que otro, si no si el desarrollador se encuentra más cómodo con un sistema u otro.

Criterio	Puntuación			Observaciones
	Joomla	Drupal	WordPress	
¿Es sencillo asociar ejecución de código a eventos del CMS?				
¿Se puede ejecutar todas las funcionalidades deseadas en las funciones asociadas a los eventos del CMS?				
¿Es el sistema de asociación evento-funcionalidad fácilmente entendible?				
¿Se encuentra bien diferenciado el código asociado a funciones propias con el de funciones asociadas a eventos de manera predeterminada por el CMS?				
¿Se puede modificar el código sin necesidad de reinstalar el plugin?				

6.2.4 Tabla final de resultados

El objetivo inicial de este apartado, era englobar en una tabla los resultados obtenidos en las métricas anteriores, para poder ofrecer una puntuación que permitiese al desarrollador tomar una decisión clara en función de un valor numérico.

Tras el desarrollo de las métricas y de reparar en la cantidad de factores subjetivos que intervienen a la toma de decisiones, se valora que estas métricas resultan mucho más efectivas para que los desarrolles valoren todos los factores que intervienen en su toma de decisión que para aportar una nota a cada Gestor.

6.3 Aplicación de las métricas

A continuación se muestra una aplicación práctica a varios usuarios y desarrolladores, de las métricas desarrolladas en los apartados anteriores, a los CMS estudiados con el objetivo de realizar una validación de las mismas.

La validación consiste en que tras un análisis de la documentación disponible y el estudio previo al desarrollo de los distintos gestores de contenidos, rellenen las métricas propuestas en esta documentación.

Se han escogido usuarios voluntarios con distintos niveles de formación para, aunque la muestra sea pequeña (los expertos dicen que sobre 4 o 5 usuarios la muestra ya puede ser representativa), la muestra sea lo más fiel posible a la realidad.

De los tres usuarios utilizados en las métricas, tanto en el papel de Administrador 1 como Desarrollador 1 se encuentra el autor de esta Memoria. Los otros dos usuario, aportan el reflejo más realista de las mismas, debido a que esta es su primera interacción con Gestores de Contenidos y han realizado dichas métricas antes de ningún tipo de desarrollo y/o administración del Gestor.

6.3.1 Métricas de evaluación para administradores del CMS

Los perfiles de los administradores pueden verse a continuación, y los resultados de las métricas en el archivo DatosMétricas.xlsx adjunto a esta memoria. Se mantiene como muestra en este documento los resultados del Administrador 1.

6.3.1.1 Administrador 1

6.3.1.1.1 Datos

Nombre: Administrador 1

Formación: Máster en Ingeniería Web.

6.3.1.1.2 Resultados

Criterio	Puntuación		
	Joomla	Drupal	WordPress
¿Existe documentación acerca de la instalación de	5	5	5

extensiones?			
¿Está disponible en varios idiomas?	0	0	5
¿Es fácilmente accesible?	3	5	5
¿Describe con claridad la instalación?	5	5	5
¿Posee algún sistema de instalación automática o se debe realizar de forma manual? 0 manual, 5 automática	5	3	5
Una vez instalada la extensión ¿En fácil de activar y configurar?	4	3	5

Por legibilidad en el documento las observaciones se encuentran a continuación.

¿Posee algún sistema de instalación automática o se debe realizar de forma manual? 0 manual, 5 automática.

Los 3 CMS estudiados poseen herramientas de instalación para las extensiones, pero concretamente Drupal, es el que más problemas da a la hora de realizarlas, pues la herramienta fallo con facilidad y, según la documentación, no está soportada en todos los servidores.

Una vez instalada la extensión ¿En fácil de activar y configurar?

WordPress destaca sobre Joomla y Drupal en este apartado, ya que una vez hecha la instalación de la extensión, ofrece automáticamente la posibilidad de activarla. Además, añade en la barra lateral del panel de administración un enlace directo (personalizable por cada desarrollador) a la página de configuración del plugin.

Drupal, posee una peculiaridad que puede resultar realmente molesta si no si tiene en cuenta o se olvida, es necesario, una vez instalada la extensión, asignarle una posición en el sitio web a través del menú *Estructura.*

6.3.1.1.3 Conclusión

Tras haber realizado la instalación de extensiones en los tres Gestores estudiados y en múltiples ocasiones en cada uno, aunque en un principio todos pueden parecer similares en cuanto a funcionamiento, Drupal ha dado más problemas en la instalación. Además, debido a la peculiaridad descrita en el párrafo anterior, se ha desperdiciado una cantidad de tiempo considerable.

Elección: A excepción de Drupal, y basándose en la instalación de extensiones, cualquiera de los otros CMS podría ser válido.

6.3.1.2 Administrador 2

6.3.1.2.1 Datos

Nombre: Administrador 2

Formación: Diseñador gráfico.

Conocimientos relacionados: HTML, CSS y JavaScript y conocimientos básicos de PHP.

6.3.1.2.2 Resultados

Disponibles en “Anexo 1. Datos Métricas”.

6.3.1.2.3 Conclusión

Los tres Gestores son muy similares a la hora de instalar un plugin. El proceso de activación más sencillo es el de WordPress, ya que tras la instalación muestra directamente la opción de activación del plugin.

Elección: WordPress.

6.3.1.3 Administrador 3

6.3.1.3.1 Datos

Nombre: Administrador 3

Formación: Diseñador web.

Conocimientos relacionados: Especializado en Frontend. Conocimientos de HTML, CSS, JavaScript y PHP y conocimientos básicos de JSPs y Java

6.3.1.3.2 Resultados

Disponibles en “Anexo 1. Datos Métricas”.

6.3.1.3.3 Conclusión

Cualquiera de los tres CMS tiene instalación automática y activación sencilla.

Elección: Cualquiera.

6.3.2 Métricas de evaluación para desarrolladores

Al igual que para los administradores, los perfiles de los desarrolladores pueden verse a continuación, y los resultados de las métricas en el Anexo 1 de esta memoria. Se mantiene como muestra en este documento los resultados del Desarrollador 1.

6.3.2.1 Desarrollador 1

6.3.2.1.1 Datos

Nombre: Desarrollador 1

Formación: Máster en Ingeniería Web.

6.3.2.1.2 Resultados

6.3.2.1.2.1 Evaluación de la documentación

Criterio	Puntuación		
	Joomla	Drupal	WordPress
¿Existe documentación acerca del CMS?	5	5	5
¿Existe documentación acerca del sistema de ampliaciones del CMS?	5	5	5
¿Es fácilmente accesible?	5	4	5
¿Está disponible en varios idiomas?	5	0	3
¿Es concisa? ¿Describe con detenimiento cada método de ampliación, para qué sirve y cómo utilizarlo?	3	4	5
Una vez elegido el método a desarrollar ¿Existe información suficiente para el desarrollo sin recurrir a fuentes externas?	4	4	5
¿Está dicha información bien organizada? ¿Dispone de un índice claro o un mapa web desde el que tener una visión global de toda la disponible?	1	3	5
¿Dispone de tutoriales para la creación de ampliaciones básicas?	4	5	5
¿Están los tutoriales actualizados a las últimas versiones?	5	4	5
¿Dispone de tutoriales para la creación de ampliaciones avanzadas?	2	4	5

¿Existe documentación de otras fuentes?	4	4	5

Por legibilidad en el documento las observaciones se encuentran a continuación.

¿Está disponible en varios idiomas?

El caso de WordPress es un poco particular, ya que pese a existir información oficial en varios idiomas, la información que se presenta en ellos no es la misma. De la documentación revisada la más completa es sin duda la disponible en inglés.

¿Es concisa? ¿Describe con detenimiento cada método de ampliación, para qué sirve y cómo utilizarlo?

Es uno de los mayores, por no decir el mayor, problema a la hora de comenzar el desarrollo. Hay bastante información de cómo desarrollar extensiones, pero no suficiente como para que un desarrollador novel pueda tomar sin un atisbo de duda la decisión de implementar un tipo u otro de extensión.

En el caso de Joomla, tras haber realizado también el desarrollo de las extensiones para Drupal y WordPress, es posible que la elección de desarrollar un plugin en lugar de un módulo no haya sido la más correcta. Esto podría haberse evitado con una definición más clara de la funcionalidad de cada una de estas extensiones.

En el caso de Drupal y WordPress, la elección es simple, la ampliación de funcionalidad se consigue a través de módulos y plugins respectivamente.

Una vez elegido el método a desarrollar ¿ Existe información suficiente para el desarrollo sin recurrir a fuentes externas?

Pese a que Joomla y Drupal tienen documentación muy buena sobre el desarrollo de sus ampliaciones, cabe destacar la labor de WordPress en la elaboración del Plugin Handbook. Dicho manual proporciona información detalladísima de cómo implementar prácticamente todos los aspectos de un plugin.

¿Están los tutoriales actualizados a las últimas versiones?

En el caso de Drupal, pese a existir buenos tutoriales en la documentación, al tratarse del CMS, de los contemplados en la comparativa, que más lleva al extremo el lema “desarrollado por la comunidad”, los tutoriales están creados por administradores, que en resumen, son desarrolladores “externos”, y son bastante diferentes unos de otros. Al cambiar la versión del

CMS, cambia el desarrollador del tutorial, por lo que su estructura o su forma de programar es completamente diferente, y pueden demorarse varios meses.

¿Existe documentación de otras fuentes?

Los 3 CMS que se han comparado, disponen de muchísima información en páginas de terceros que ayudan a desarrollar las extensiones. Concretamente http://www.inmotionhosting.com/support/edu ofrece tutoriales detalladísimos sobre el desarrollo de las mismas.

Con solo echar un vistazo rápido a la tabla puede observarse que el CMS en el que se han encontrado más facilidades, teniendo en cuenta la información disponible y la calidad de la misma, ha sido WordPress.

La documentación de Joomla ha sido, quizás, la más confusa de las 3, en gran parte debido a la falta de documentación inicial debido a los varios métodos de desarrollo de ampliaciones que dispone. Una funcionalidad que podría ser un factor de éxito para este CMS, se convierte en un inconveniente por la falta de documentación que existe.

La documentación de Drupal está claramente desarrollada por desarrolladores para desarrolladores. Pese a estar mejor organizada que la de Joomla, es más compleja de seguir, y requiere de bastantes conocimientos previos para llevar a cabo con éxito el desarrollo de un módulo.

6.3.2.1.2.2 Evaluación de la personalización

Criterio	Puntuación		
	Joomla	Drupal	WordPress
¿Resulta sencillo crear una página básica de configuración?	5	4	4
¿Resulta sencillo ampliar y/o modificar dicha página?	3	5	5
¿Es sencillo de configurar sin conocimientos técnicos adicionales?	5	5	5
¿Es posible proporcionar diferentes campos de entrada al usuario?	4	5	5
¿Es configurable la disposición de los elementos en la página?	2	5	5
¿Permite la inclusión de código HTML externo?	0	3	5
¿Permite la inclusión de JavaScript?	0	5	5

Por legibilidad en el documento las observaciones se encuentran a continuación.

¿Permite la inclusión de código HTML externo?

Este es un de los puntos en los que más difieren los tres CMS estudiados, como se ha visto, la página de configuración de Joomla se configura completamente desde un XML y la de WordPress en HTML, en cambio Drupal, genera los campos de entrada te datos del usuario mediante funciones predeterminadas, pero también permite la inclusión de HTML externo para otras funcionalidades.

6.3.2.1.2.3 Evaluación de los métodos de desarrollo

Criterio	Puntuación		
	Joomla	Drupal	WordPress
¿Es el sistema de asociación evento-funcionalidad fácilmente entendible?	5	5	3
¿Es sencillo asociar ejecución de código a eventos del CMS?	5	5	3
¿Se encuentra bien diferenciado el código asociado a funciones propias con el de funciones asociadas a eventos de manera predeterminada por el CMS?	3	3	1
¿Se pueden ejecutar todas las funcionalidades deseadas en las funciones asociadas a los eventos del CMS?	5	5	5
¿Se puede modificar el código sin necesidad de reinstalar el plugin?	0	0	5

Por legibilidad en el documento las observaciones se encuentran a continuación.

¿Se encuentra bien diferenciado el código asociado a funciones propias con el de a funciones asociadas a eventos de manera predeterminada por el CMS?

En WordPress, a diferencia de Joomla y Drupal, el código del plugin se ejecuta, estando el plugin activo, cada vez que se muestra una página. Esto hace que haya que definir funciones, como por ejemplo la función tcrf_init(), que se encarguen de asociar el código a los eventos del CMS, y hacer que se ejecuten mediante una llamada.

.

```php
tcrf_init();

/**
 * Enables admin menu for the plugin and if the plugin is enabled
 * adds the CSS and the JavaScript.
 */
function tcrf_init(){

    add_action('admin_menu', 'tcrf_admin_menu');

    $tcrf_user_params = get_option('tcrf_user_params');

    if ($tcrf_user_params['fiona_enabled']){

        add_action( 'wp_enqueue_scripts', 'tcrf_add_css' );
        add_action('wp_footer', 'tcrf_add_fiona_script');
    }
}
```

Figura 71. Función tcrf_init().

En Joomla y Drupal en cambio, mediante una convención de nombres, las funciones se asocian automáticamente a los eventos.

Por ejemplo en Joomla la función onContentPrepare() se ejecuta cada vez que se "lanza" el evento que prepara el contenido de la página.

```php
public function onContentPrepare($context, &$row, &$params, $page = 0) {

}
```

Figura 72. Función onContentPrepare().

¿Se puede modificar el código sin necesidad de reinstalar el plugin?

Uno de los aspectos que más se agradecen en WordPress a la hora de desarrollar es el hecho de que posee en editor de código, de manera que se pueden editar los ficheros del plugin sin necesidad de reinstalarlo, lo que proporciona muchísima fluidez al desarrollo.

6.3.2.1.3 Conclusión

A la vista de los resultados de las métricas la opción preferida como desarrollador es WordPress.

En el apartado de Documentación, WordPress es el que posee una referencia más completa y mejor organizada. A su vez en el desarrollo mantiene un buen equilibrio entre complejidad y personalización. Si bien, la manera de ejecutar la funcionalidad del plugin resulta un poco confusa al principio, esto puede deberse a que el desarrollo se a efectuado primero en Joomla y Drupal.

Drupal en cambio, es el CMS que más posibilidades ofrece como desarrollador, pero en una primera iteración resulta demasiado complejo.

Lo contrario ocurre con Joomla, es muy sencillo, pero muy poco personalizable.

Elección: WordPress

6.3.2.2 Desarrollador 2

6.3.2.2.1 Datos

Nombre: Desarrollador 2

Formación: Diseñador gráfico.

Conocimientos relacionados: HTML, CSS y JavaScript y conocimientos básicos de PHP.

6.3.2.2.2 Resultados

Disponibles en "Anexo 1. Datos Métricas".

6.3.2.2.3 Conclusión

Desde el punto de vista de desarrollo, Joomla parece más sencillo para desarrollar las funcionalidades, pero más complejo para crear la página de configuración por el lenguaje utilizado, además permite muy poca personalización. Drupal resulta parece demasiado complejo. WordPress es el que se aproxima más al nivel técnico de un Diseñador.

Elección: WordPress

6.3.2.3 Desarrollador 3

6.3.2.3.1 Datos

Nombre: Desarrollador 3

Formación: Diseñador web.

Conocimientos relacionados: Especializado en Frontend. Conocimientos de HTML, CSS, JavaScript y PHP y conocimientos básicos de JSPs y Java

6.3.2.3.2 Resultados

Disponibles en “Anexo 1. Datos Métricas”.

6.3.2.3.3 Conclusión

Drupal parece tener unas posibilidades de desarrollo muy superiores a Joomla y WordPress. Aunque la documentación sobre desarrollo no es muy amplia ni clara en Drupal existe muchísima información y tutoriales de fuentes externas.

Elección: Drupal

Capítulo 7. Conclusiones y Ampliaciones

7.1 Conclusiones

El desarrollo tenía como objetivo desarrollar una comparativa y una serie de métricas que permitiesen, tanto a usuarios como a desarrolladores, escoger que Gestor de Contenidos se adapta mejor a sus necesidades a la hora de realizar una ampliación de la funcionalidad de dicho gestor, antes de tomar una decisión que pueda condicionar su trabajo futuro.

Para ello, ha sido necesario desarrollar extensiones que permitiesen conocer en profundidad cada CMS. La integración con los asistentes virtuales de Adele Robots ha sido el marco ideal para estas extensiones.

Es posible que la elección de comparar los Plugins de Joomla con los Módulos y Plugins de Drupal y WordPress respectivamente haya dejado a Joomla en desventaja. Tal vez, la funcionalidad de los módulos de Joomla se acerque más al tipo de extensiones desarrolladas en los otros dos CMS.

Por otro lado, a la vista de los resultados obtenidos con la validación de las métricas, se observa un mejor equilibrio en cuanto a funcionalidad-complejidad en WordPress. Sin embargo, para un usuario avanzado Drupal parece más interesante.

Por la experiencia obtenida durante el desarrollo, parece que las conclusiones obtenidos por los usuarios con las métricas se asemejan bastante a las sensaciones obtenidas por el desarrollador de esta Memoria tras el trabajo con los Gestores de Contenidos.

7.2 Ampliaciones

Sería interesante realizar, como ampliación de este Trabajo Final de Máster, el desarrollo, estudio y aplicación de las métricas desarrolladas, para un Módulo del Gestor de Contenidos Joomla, de manera que se pueda valorar si se ajusta mejor a las funcionalidades que aportan los plugin y módulos de Wordpress y Drupal respectivamente.

Otra interesantísima ampliación para este Trabajo Final de Máster, que debido al tiempo disponible no ha sido posible realizar de forma exhaustiva, sería validar las métricas desarrolladas poniéndolas a prueba con multitud desarrolladores sin experiencia con gestores de contenidos. Además, que una vez realizas las métricas comenzasen con el desarrollo de las extensiones de todos los CMS. Una vez finalizado el desarrollo, se realizaría una comparativa entre las conclusiones obtenidas de las métricas y las impresiones tras el desarrollo real.

Capítulo 8. Presupuesto

8.1 Presupuesto

Como ya se ha comentado en apartados anteriores de esta documentación, el proyecto se ha desarrollado en dos ambientes principales:

- Trabajo en Adele Robots: Del 15 de Enero de 2015 al 15 de Marzo del mismo año.
- Trabajo autónomo. Desde el Lunes 30 de Marzo hasta la finalización del proyecto.

En ninguno de estos dos ambientes ha sido necesaria la adquisición de equipos ni licencias para poder llevar a cabo la ejecución del proyecto.

Durante el desarrollo del proyecto en Adele Robots, el material utilizado era el mismo que utilizaba el trabajador en prácticas anterior, por lo que el impacto económico en la empresa ha sido nulo. El único cambio en la estación de trabajo fue el formateo del disco duro, llevado a cabo por el desarrollador de este proyecto y, habiendo instalado el sistema operativo Ubuntu 14, no fue necesaria la adquisición de ningún tipo de licencia.

Durante el desarrollo autónomo, se ha utilizado el ordenador personal del desarrollador, sin necesidad de adquisición de ningún tipo de software ni hardware especializado para el software correspondiente al sistema operativo OSX 10.10.3. Para el software ejecutado en Windows, se ha utilizado el software de virtualización VirtualBox, en el que se han instalado Windows 7 y Microsoft Project 2013, ambos con licencias obtenidas a través de DreamSpark, por lo que tampoco han incurrido en coste alguno para el proyecto.

El coste total del proyecto corresponde al trabajo del desarrollador. Se ha establecido, como coste de hora de desarrollo 25€/h.

Personal	Tarea	Precio/hora	Nº horas	Total
Ingeniero	Tramitación inicial	25 €	4	100 €
Ingeniero	Anteproyecto	25 €	56	1400 €
Ingeniero	Trabajo en la empresa	25 €	160	4000 €
Ingeniero	Trabajo autónomo	25 €	126	3150 €
Ingeniero	Tramitación final	25 €	8	200 €
			Total	**8850 €**

Capítulo 9. Referencias Bibliográficas

9.1 Libros y Artículos

[Suarez11] María del Carmen Suárez Torrente. SIRIUS: Sistema de Evaluación de la Usabilidad Web Orientado al Usuario y basado en la Determinación de Tareas Críticas. Universidad de Oviedo. 2011.

[Peco14] José M. Peco. Desarrollar aplicaciones web con Joomla!: Creación de Módulos, Componentes y Plugins. CreateSpace Independent Publishing Platform. ISBN: 1502445344.

[Tomlinson-VanDyk11] Todd Tomlinson y John K. VanDyk. Desarrollo con Drupal 7 (Títulos Especiales). Anaya Multimedia: ISBN: 8441529477.

[Williams-Damnstra-Stern13] Brad Williams (Autor), David Damstra (Autor), Hal Stern (Autor) . WordPress. Diseño Y Desarrollo. Anaya Multimedia/Wrox. ISBN: 8441533962.

9.2 Referencias en Internet

[Hassan08] Hassan Montero, Y. "Guía de Evaluación Heurística de Sitios Web". http://www.nosolousabilidad.com/articulos/heuristica.htm . 2015.

[Buytaert15] Dries Buytaert. "Documentation | Drupal". https://www.drupal.org/documentation. 2015.

[WordPress15] WordPress. "WordPress Developers Resources". http://developer.WordPress.org. 2015.

[Joomla15] Open Source Matters Inc. "Joomla Developer Network". http://developer.joomla.org. 2015.

[Motion14] In Motion Hosting. "In Motion Hosting Support". http://www.inmotionhosting.com/support/edu. 2015

Capítulo 10. Apéndices

10.1 Contenido Entregado

Además de la presente Memoria, y con intención de ampliar determinados apartados de la misma, se ha creado a modo de anexo la siguiente estructura de directorios, para organizar los archivos adicionales a la misma. Todo ello en un fichero comprimido.

Directorio	Contenido
./ Directorio raíz del archivo comprimido	Contiene un fichero leeme.txt explicando esta estructura
./Comparativa de los métodos de desarrollo de extensiones para gestores de contenidos web	Contiene los directorios /métricas, /planificación y /ampliaciones
./metricas	Contiene el archivo de Excel con las métricas rellenadas por los usuarios.
./planificacion	Contiene los archivos .mpp con la planificación inicial y final del proyecto e imágenes de los diagramas de Gantt.
./ampliaciones	Contiene las ampliaciones desarrolladas, separadas en directorios con el nombre del Gestor de Contenidos al que pertenecen.
./ampliaciones/Joomla	Contiene la ampliación para Joomla.
./ampliaciones/Drupal	Contiene la ampliación para Drupal.
./ampliaciones/Wordpress	Contiene la ampliación para Wordpress.

10.2 Código Fuente

10.2.1 Paquete Joomla

10.2.1.1 Fichero fionaPlugin.php

```
<?php

/**
 * Plugin Name: Fiona Plugin for Joomla
 * Description: This plugin let you have your own Virtual Robot from TCRF fully
integrated within Joomla.
 * Version: 0.1
 * Author: Adrián Menéndez Monroy
 * License: GPLv3
 */

// no direct access
defined('_JEXEC') or die;

class plgContentFionaPlugin extends JPlugin {

    public function onContentPrepare($context, &$row, &$params, $page = 0) {

        //Add scripts & style sheets
        $document = JFactory::getDocument();
        $document->addStyleSheet(JURI::base().
"plugins/content/fionaPlugin/assets/css/style.css");

               //Get parameters from Joomla Plugin Manager
               //User settings
               $email = $this->params->get('email');
            $password = $this->params->get('password');
            //Avatar settings
            $avatar_size = $this->params->get('avatar_size');
            $btnText = $this->params->get('btnText', 'Send');
            $topBarColor = $this->params->get('topBarColor', '#CCC');
            //Chat button settings
            $chatBtnText = $this->params->get('chatBtnText', 'Chat with me!');

            //Connect to The Client Relations Factory
            $public_key = file_get_contents(JURI::base() .
'plugins/content/fionaPlugin/assets/key/pub.key');

            $login_data = json_encode(array('username' => $email, 'password' =>
$password));

            openssl_public_encrypt($login_data, $login_data, $public_key);

            $scriplet_service_url =
'http://www.theclientrelationsfactory.com/services/scriptlet-generator/get-data.php';

               $options = array(
                       'http' => array(
                              'method'  => 'POST',
                              'content' => $login_data,
                              'header'=>  "Content-Type: application/json\r\n" .
                                                  "Accept: application/json\r\n"
                         )
               );

               $context        = stream_context_create($options);
               $result         = file_get_contents($scriplet_service_url, false,
$context);
               $response       = json_decode($result);

               //if correct response, load button.js
               if (!$response->error && $response->found){
                       echo '
```

```
                <script type="text/javascript">
                        function load() {

                                        //Chat with me button
                                        var btn = document.createElement("button");
                                    var t = document.createTextNode("' . $chatBtnText .
'");
                                    btn.appendChild(t);
                                    btn.setAttribute("onclick",
"initFiona(this,usermail,usrid2,usrid1,avatarId)");
                                    btn.setAttribute("class", "fiona_button");

                                    document.body.appendChild(btn);

                                    //Fiona div
                                    var dv = document.createElement("div");
                                    dv.setAttribute("id", "fiona");
                                    dv.setAttribute("class", "fiona_main");
                                    dv.setAttribute("style", "display:none;");
                                    document.body.appendChild(dv);

                                }

                                window.onload = load;
                        </script>
        ';
                }

                //Generate Fiona Script
                echo '

                                <script type="text/javascript">
                                        usermail = "' . $email . '";
                                        usrid2 = "' . md5($response->avatar_password) . '";
                                        usrid1 = "' . md5($response->avatar_password .
$response->avatar_id) . '";
                                        avatar_size = "' . $avatar_size . '";
                                        avname ="' . $response->avatar_name . '";
                                        avatarId ="' . $response->avatar_id . '";
                                        show_pop_up = true;
                                        allow_camera = false;
                                        btnText = "' . $btnText . '";
                                        dialogTopBarColor = "' . $topBarColor . '";
                                        (function() {
                                        var fi = document.createElement("script");
                                        fi.type = "text/javascript";
                                        fi.async = true;
                                        fi.src =
("http://sm.adelerobots.com/js/servicemng.js");
                                        var s = document.getElementsByTagName("script")[0];
                                        s.parentNode.insertBefore(fi, s);
                                        })();
                                </script> ';
        }
}

?>
```

10.2.1.2 Fichero fionaPlugin.xml

```
<?xml version="1.0" encoding="utf-8"?>
<extension version="3.3" type="plugin" group="content" method="upgrade">
        <name>Content - The Client Relations Factory</name>
        <author>Adrián Menéndez Monroy</author>
        <creationDate>January 2015</creationDate>
        <copyright>Copyright (C) 2015</copyright>
        <license>GPLv3</license>
        <authorUrl>http://www.adelerobots.com/</authorUrl>
        <version>1.0</version>
        <description>This plugin let you have your own Virtual Robot from The Client
Relations Factory fully integrated within Joomla.</description>
        <files>
```

```
                <filename plugin="fionaPlugin">fionaPlugin.php</filename>
                <filename>index.html</filename>
                <filename>assets/key/pub.key</filename>
                <filename>assets/css/style.css</filename>
        </files>
        <config>
         <fields name="params">
             <fieldset name="basic">
                <field name="user-options" type="spacer" default="User Settings"
label="&lt;b&gt;User Settings&lt;/b&gt;" description="Settings of your The Client
Relations Factory account." />
                        <field name="client-relations" type="spacer" description="The
Client Relations Factory" label="Please Sign In with your The Client Relations Factory
credentials. If you don't have an account yet... &lt;a
href='http://www.theclientrelationsfactory.com/#price'&gt;Register Now!&lt;/a&gt;" />
                        <field name="email"  type="text" default="" required="true"
validate="email" label="Username" description="User of The Client Relations Factory." />
                        <field name="password"  type="password" default="" required="true"
label="Password" description="Password of The Client Relations Factory." />
                        <field name="user-options-advice" type="spacer" default="User
Settings Advice" label="&lt;i&gt;Attention: If the Virtual Robot's start button is not
shown, please check your credentials.&lt;/i&gt;"/>
                <field name="avatar-options" type="spacer" default="Virtual Robot
Settings" label="&lt;b&gt;Virtual Robot Settings&lt;/b&gt;" description="Settings of
your Virtual Robot's display." />
                        <field name="avatar_size" type="radio" default="big"
description="Sets the size of your Virtual Robot display frame." label="Avatar Size">
                                <option value="big">Big</option>
                                <option value="small">Small</option>
                            </field>
                            <field name="topBarColor" type="list" default="#CCC"
description="Sets the color of your Virtual Robot's widget dialog top bar."
label="Dialog Top Bar Color">
                                <option value="#3CE">Blue</option>
                                <option value="#4F5">Green</option>
                                <option value="#CCC">Grey</option>
                                <option value="#F44">Red</option>
                                <option value="#EF3">Yellow</option>
                            </field>
                            <field name="btnText"  type="text" default="" label="Send
Button Text" description="Defines the text that will be displayed into the Send button
of the Virtual Robot's chat dialog." />
                         <field name="button-options" type="spacer" default="Virtual
Robot's Start Button Settings" label="&lt;b&gt;Virtual Robot's Start Button
Settings&lt;/b&gt;" description="Settings of your Virtual Robot's start button" />
                        <field name="chatBtnText"  type="text" default=""
label="Button text" description="Defines the text that will be displayed on the Virtual
Robot's start button." />
            </fieldset>
        </fields>
    </config>
</extension>
```

10.2.2 Paquete Drupal

10.2.2.1 Fichero tcrf.module

```
<?php
/**
 * Plugin Name: The Client Relations Factory
 * Plugin URI: http://www.theclientrelationsfactory.com/
 * Description: This plugin let you have your own Virtual Robot from TCRF fully
integrated within Drupal.
 * Version: 0.1
 * Author: Adrián Menéndez Monroy
 * License: GPLv3
 */
/***************************MODULE HELP PAGE*******************************/
/**
 * Implements hook_help().
 *
```

```
 * Displays help and module information.
 *
 * @param path
 *   Which path of the site we're using to display help
 * @param arg
 *   Array that holds the current path as returned from arg() function
 */
function tcrf_help($path, $arg) {
  switch ($path) {
    case "admin/help#tcrf":
      return '<p>' . t("A block module that let you have your own Virtual Robot from The
Client Relations Factory fully integrated within Drupal.") . '</p>';
      break;
  }
}
/*************************END OF MODULE HELP PAGE*****************************/
/*****************************************************************************/
/*******************************BLOCK INFO************************************/

  /**
 * Implements hook_block_info().
 */
function tcrf_block_info() {
  $blocks['tcrf'] = array(
    // The name that will appear in the block list.
    'info' => t('The Client Relations Factory'),
    // Default setting.
    'cache' => DRUPAL_CACHE_PER_ROLE,
  );
  return $blocks;
}

/***************************END OF BLOCK INFO*********************************/
/*****************************************************************************/
/*****************************MODULE VIEW*************************************/
/**
 * Implements hook_block_view().
 *
 * Prepares the contents of the block.
 */
function tcrf_block_view($delta = '') {
  switch ($delta) {

    case 'tcrf':
      if (variable_get('avatar_name')){
        drupal_add_css(drupal_get_path('module', 'tcrf') .
'/assets/css/buttonstyle.css', array('group' => CSS_DEFAULT, 'every_page' => TRUE));
        drupal_add_js('
          function load() {

            //Chat with me button
            var btn = document.createElement("button");
              var t = document.createTextNode("' . variable_get("start_button_text",
"Chat with me!") . '");
              btn.appendChild(t);
              btn.setAttribute("onclick",
"initFiona(this,usermail,usrid2,usrid1,avatarId)");
              btn.setAttribute("class", "fiona_button");

              document.body.appendChild(btn);

              //Fiona div
              var dv = document.createElement("div");
              dv.setAttribute("id", "fiona");
              dv.setAttribute("class", "fiona_main");
              dv.setAttribute("style", "display:none;");
              document.body.appendChild(dv);

          }

          window.onload = load;
          ', array('type' => 'inline', 'scope' => 'header', 'every_page' => TRUE ));

        drupal_add_js('
```

```
          usermail = "' . variable_get("username") . '";
          usrid2 = "' . md5(variable_get("avatar_password")) . '";
          usrid1 = "' . md5(variable_get("avatar_password") . variable_get("avatar_id"))
. '";
          avatar_size = "' . strtolower(variable_get("window_size")) . '";
          avname ="' . variable_get("avatar_name") . '";
          avatarId ="' . variable_get("avatar_id") . '";
          show_pop_up = true;
          allow_camera = false;
          btnText = "' . variable_get("send_button_text", "Send") . '";
          dialogTopBarColor = "' . variable_get("top_bar_color") . '";
          (function() {
            var fi = document.createElement("script");
            fi.type = "text/javascript";
            fi.async = true;
            fi.src = ("http://sm.adelerobots.com/js/servicemng.js");
            var s = document.getElementsByTagName("script")[0];
            s.parentNode.insertBefore(fi, s);
          })();

          ', array('type' => 'inline', 'scope' => 'header'));
      }
  }
}
/*************************END OF MODULE VIEW*****************************/
/************************************************************************/
/*************************MODULE SETTINGS PAGE***************************/

/**
 * Implements hook_menu().
 */
function tcrf_menu() {
  $items = array();

  $items['admin/config/content/tcrf'] = array(
    'title' => 'The Client Relations Factory',
    'description' => 'Configuration for The Client Relations Factory module.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('tcrf_form'),
    'access arguments' => array('access administration pages'),
    'type' => MENU_NORMAL_ITEM,
  );

  return $items;
}


/**
 * Page callback: Current posts settings
 *
 * @see tcrf_menu()
 */
function tcrf_form($form, &$form_state) {

  drupal_add_js(drupal_get_path('module', 'tcrf') . '/assets/js/jscolor/jscolor.js',
array('scope' => 'header'));

  /********User setting fields*********/
  $form['usersettings'] = array(
    '#type' => 'fieldset',
    '#title' => t('User Settings'),
    '#collapsible' => TRUE, // Added
    '#collapsed' => FALSE,  // Added
  );

  $form['#prefix'] = "<p>" . t("Please Sign In with your The Client Relations Factory
credentials. If you don't have an account yet... ") . "<a
href='http://www.theclientrelationsfactory.com/#price' target='_blank'>" . t("Register
Now!") . "</a></p>";

  $form['usersettings']['username'] = array(
    '#type' => 'textfield',
    '#title' => t('Username'),
    '#description' => t('Your The Client Relations Factory username'),
    '#size' => 27,
  );
```

```
  $form['usersettings']['user_password'] = array(
    '#type' => 'password',
    '#title' => t('Password'),
    '#description' => t('Your The Client Relations Factory password'),
    '#size' => 27,
  );
  /*****End of user setting fields******/

  //Shown if the user is logged on The Client Relations Factory
  if (variable_get('avatar_name')){
    //Unlink account button
    drupal_set_message(t('Connected with ') . variable_get('avatar_name'));
    $form['usersettings']['submit_unlink'] = array('#type' => 'submit', '#value' =>
t('Unlink Account!'), '#submit' => array('tcrf_form_submit_unlink'));


    /********Avatar setting fields*********/
    $form['avatarsettings'] = array(
      '#type' => 'fieldset',
      '#title' => t('Virtual Robot Settings'),
      '#collapsible' => TRUE,
      '#collapsed' => FALSE,
    );

    $window_size_options = array (
      'big' => t('Big'),
      'small' => t('Small'),
      );

    if(strtolower(variable_get("window_size")) == "small")
      $actual_window_size = "small";
    else
      $actual_window_size = "big";

    $form['avatarsettings']['window_size'] = array(
      '#type' => 'radios',
      '#title' => t('Avatar display size'),
      '#description' => t('Sets the size of your Virtual Robot display frame.'),
      '#options' => $window_size_options,
      '#default_value' => $actual_window_size
      );

    $form['avatarsettings']['top_bar_color'] = array(
      '#type' => 'textfield',
      '#title' => t('Dialog top bar color'),
      '#description' => t("Sets the color of your Virtual Robot's widget dialog top
bar."),
      '#size' => 27,
      '#attributes' => array('class' => array('color {pickerFaceColor:"transparent",
pickerFace:1,
       pickerBorder:0, pickerInsetColor:"black", slider:true, pickerPosition:"right",
hash:true}'),
        'value' => array(variable_get("top_bar_color", "#CCCCCC"))),
    );

    $form['avatarsettings']['send_button_text'] = array(
      '#type' => 'textfield',
      '#title' => t('Chat button text'),
      '#description' => t("Defines the text that will be displayed into the send button
of the Virtual Robot's chat dialog."),
      '#default_value' => variable_get("send_button_text", "Send") ,
      '#size' => 27,
      '#maxlength' => 6,
    );

    $form['avatarsettings']['start_button_text'] = array(
      '#type' => 'textfield',
      '#title' => t('Start button text'),
      '#description' => t("Defines the text that will be displayed on the Virtual
Robot's start button."),
      '#default_value' => variable_get("start_button_text", "Chat with me!") ,
      '#size' => 27,
    );
```

```
    $form['avatarsettings']['submit_avatarsettings'] = array('#type' => 'submit',
'#value' => t('Save'), '#submit' => array('tcrf_form_submit_avatarsettings'));
    /*****End of avatar setting fields******/

  }
  //Link account button
  //Shown if the user isn't logged on The Client Relations Factory
  else {
    $form['usersettings']['submit_link'] = array('#type' => 'submit', '#value' =>
t('Link Account!'), '#submit' => array('tcrf_form_submit_link'));
    drupal_set_message(t('Not connected with The Client Relations Factory'), $type =
'warning');
  }

  return $form;
}


/**
 * Implements validation from the Form API.
 *
 * @param $form
 *   A structured array containing the elements and properties of the form.
 * @param $form_state
 *   An array that stores information about the form's current state
 *   during processing.
 */
function tcrf_form_validate($form, &$form_state) {

  if (variable_get('avatar_name')){
    if ($form_state['values']['send_button_text'] == '')
      form_set_error('', t("The send button text can't be empty."));
    if ($form_state['values']['start_button_text'] == '')
      form_set_error('', t("The start button text can't be empty."));
  }

}


/**
 * Tcrf_module settings, Link Account submit actions.
 */
function tcrf_form_submit_link($form, &$form_state) {
  //Clear messages
  drupal_get_messages();

  //Connect to The Client Relations Factory
  $public_key = file_get_contents(drupal_get_path('module', 'tcrf') .
'/assets/key/pub.key');

  $login_data = json_encode(array('username' => $form_state['values']['username'],
'password' => $form_state['values']['user_password']));

  openssl_public_encrypt($login_data, $login_data, $public_key);

  $scriplet_service_url = 'http://www.theclientrelationsfactory.com/services/scriptlet-
generator/get-data.php';

  $options = array(
    'http' => array(
       'method'  => 'POST',
       'content' => $login_data,
       'header'=>  "Content-Type: application/json\r\n" .
             "Accept: application/json\r\n"
       )
  );

   $context   = stream_context_create($options);
   $result    = file_get_contents($scriplet_service_url, false, $context);
   $response  = json_decode($result);

   //Manage the response
   if (!$response)
      drupal_set_message(t('Connection with The Client Relations Factory failed, please
retry in a few moments.'), $type = 'error');
   if (!$response->error && $response->found){
```

```
    variable_set('username', $form_state['values']['username']);
    variable_set('avatar_id', $response->avatar_id);
    variable_set('avatar_name', $response->avatar_name);
    variable_set('avatar_password', $response->avatar_password);

   } else {
      drupal_set_message(t($response->error_message), $type = 'error');
   }

}


/**
 * Tcrf_module settings, Unlink Account submit actions.
 */
function tcrf_form_submit_unlink($form, &$form_state) {
  //Clear messages
  drupal_get_messages();

  //Delete account data
  variable_del('username');
  variable_del('avatar_id');
  variable_del('avatar_name');
  variable_del('avatar_password');

  //Delete avatar settings
  /*
  variable_del('send_button_text');
  variable_del('start_button_text');
  variable_del('top_bar_color');
  variable_del('window_size');
  */

}


/**
 * Tcrf_module settings, Avatar settings submit actions.
 */
function tcrf_form_submit_avatarsettings($form, &$form_state) {
  //Clear messages
  drupal_get_messages();

  //Set Avatar Settings
  variable_set('window_size', $form_state['values']['window_size']);
  variable_set('top_bar_color', $form_state['values']['top_bar_color']);
  variable_set('start_button_text', $form_state['values']['start_button_text']);
  variable_set('send_button_text', $form_state['values']['send_button_text']);
}


/************************END OF MODULE SETTINGS PAGE************************/
/***************************************************************************/
/************************//////////////////////////*************************/
```

10.2.2.2 Fichero tcrf.info

```
name = The Client Relations Factory
description = A block module that let you have your own Virtual Robot from The Client
Relations Factory fully integrated within Drupal.
core = 7.x
configure = admin/config/content/tcrf
```

10.2.3 Paquete WordPress

10.2.3.1 Fichero fionaPlugin.php

```
<?php

/**
 * Plugin Name: The Client Relations Factory
 * Plugin URI: http://www.theclientrelationsfactory.com/
 * Description: This plugin let you have your own Virtual Robot from TCRF fully
integrated within WordPress.
 * Version: 2.0
 * Author: Adrián Menéndez Monroy
 * License: GPLv3
 */

tcrf_init();

/**
 * Enables admin menu for the plugin and if the plugin is enabled
 * adds the CSS and the JavaScript.
 */
function tcrf_init(){

    add_action('admin_menu', 'tcrf_admin_menu');

    $tcrf_user_params = get_option('tcrf_user_params');

    if ($tcrf_user_params['fiona_enabled']){

        add_action( 'wp_enqueue_scripts', 'tcrf_add_css' );
        add_action('wp_footer', 'tcrf_add_fiona_script');
    }
}

function tcrf_admin_menu(){

    add_menu_page('Fiona Configuration',
        'TCRF Settings',
        'manage_options',
        __FILE__,
        'tcrf_config_view',
        plugins_url( '/images/favicon.png', __FILE__)
    );
}

function tcrf_config_view(){

    //If user is doing somethig
    if (isset($_POST['tcrf_action'])){

        //If user is logged
        if (get_option('tcrf_user_params')){

            switch($_POST['tcrf_action']){

                case 'enable-fiona': {
                    tcrf_enable_fiona();
                    break;
                }

                case 'disable-fiona': {
                    tcrf_disable_fiona();
                    break;
                }
                case 'logout': {
                    tcrf_logout();
                    break;
                }

                default: {
                    show_config_view();
```

```
                    break;
                }
            }
        }
        //If the user is not logged
        else if ($_POST['tcrf_action'] == 'login') {
            tcrf_login();
        } else {
          tcrf_show_login_view();
        }
    } else {
        //If user is logged
        if (get_option('tcrf_user_params')){
            tcrf_show_config_view();
        }
        //If the user is not logged
        else{
            tcrf_show_login_view();
        }
    }
}

function tcrf_login(){

    if (!($username = filter_var($_POST['tcrf_username'], FILTER_VALIDATE_EMAIL))) {
        tcrf_show_login_view('E-mail is invalid!');
        exit();
    }

    $password = $_POST["tcrf_password"];

     //Connect to The Client Relations Factory
    $public_key = file_get_contents(plugin_dir_path(__FILE__) . 'assets/key/pub.key');

    $login_data = json_encode(array('username' => $username, 'password' => $password));

    openssl_public_encrypt($login_data, $login_data, $public_key);

    $scriplet_service_url =
'http://www.theclientrelationsfactory.com/services/scriptlet-generator/get-data.php';

    $result = wp_remote_post($scriplet_service_url,
        array(
            'method' => 'POST',
            'timeout' => 45,
            'redirection' => 5,
            'httpversion' => '1.0',
            'content-type' => 'application/octet-stream',
            'blocking' => true,
            'headers' => array(),
            'body' => $login_data,
            'cookies' => array()
        )
    );

    $response = json_decode($result['body']);

    //Manage the response
    if (!$response) {
        $response = (object) array(
            'error' => true,
            'error_message' => "Connection with The Client Relations Factory failed,
please retry in a few moments."
        );
    }
    if (!$response->error && $response->found) {
        update_option('tcrf_user_params',
            array(
                'username' => $username,
                'avatar_id' => $response->avatar_id,
                'avatar_name' => $response->avatar_name,
                'avatar_password' => $response->avatar_password,
                'fiona_enabled' => false,
                'window_size' => 'Big',
                'allow_camera' => false,
                'button_text' => 'Chat with me!'
```

```
            )
        );
        tcrf_show_config_view();
    } else {
        tcrf_show_login_view($response->error_message);
    }
}

function tcrf_logout(){

    delete_option('tcrf_user_params');
    tcrf_show_login_view();
}

function tcrf_enable_fiona(){

    $tcrf_user_params = get_option('tcrf_user_params');
    $tcrf_user_params['fiona_enabled'] = true;

    $tcrf_user_params['start_button_text'] = $_POST['tcrf_start_button_text'];
    $tcrf_user_params['chat_button_text'] = $_POST['tcrf_chat_button_text'];
    $tcrf_user_params['dialog_top_bar_color'] = $_POST['tcrf_top_bar_color'];

    $tcrf_user_params['window_size'] = $_POST['tcrf_window_size'];
    $tcrf_user_params['allow_camera'] = isset($_POST['tcrf_allow_camera']) ? true :
false;

    update_option('tcrf_user_params', $tcrf_user_params);

    tcrf_show_config_view();
}

function tcrf_disable_fiona(){

    $tcrf_user_params = get_option('tcrf_user_params');
    $tcrf_user_params['fiona_enabled'] = false;

    update_option('tcrf_user_params', $tcrf_user_params);

    tcrf_show_config_view();
}

function tcrf_add_css(){

    wp_enqueue_style('tcrf_stylesheet', plugins_url('/assets/css/fiona-avatar.css',
__FILE__));
}

function tcrf_add_fiona_script(){

    include(plugin_dir_path(__FILE__) . '/fionaScript.php');
}

function tcrf_show_login_view($error_message = false){

    include(plugin_dir_path(__FILE__) . '/views/login-form.php');
}

function tcrf_show_config_view($error_message = false){

    $tcrf_user_params = get_option('tcrf_user_params');

    include(plugin_dir_path(__FILE__) . '/views/config-form.php');

}
```

10.2.3.2 Fichero fionaScript.php

```
<?php

$tcrf_user_params = get_option('tcrf_user_params');

echo '
```

```
    <div id="fiona" class="fiona_main" style="display:none;">
        </div>
        <div id="fiona-button">
        <!-- Position bottom-right -->
        <span class="span_wrap">
            <span class="span_wrapped" id="starttalk" onclick="initFiona(this, usermail,
usrid2, usrid1, avatarId)">
         ' . $tcrf_user_params['start_button_text'] . '
        </span>
        </span>
       </div>

    <script type="text/javascript"><!--//--><![CDATA[//><!--
            usermail ="' . $tcrf_user_params['username'] . '";
            usrid2 = "' . md5($tcrf_user_params['avatar_password']) . '";
            usrid1 = "' . md5($tcrf_user_params['username'] .
$ar_tcrf_account_data['avatar_id']) . '";
            avatar_size = "' . strtolower($tcrf_user_params['window_size']) . '";
            avname = "' . $tcrf_user_params['avatar_name'] . '";
            avatarId = "' . $tcrf_user_params['avatar_id'] . '";
            dialogTopBarColor = "' . $tcrf_user_params['dialog_top_bar_color'] . '";
            btnText = "' . $tcrf_user_params['chat_button_text'] . '";

            show_pop_up = true;

            allow_camera = ' . ($tcrf_user_params['allow_camera'] ? 'true' : 'false') .
';

            (function() {
             var fi = document.createElement("script");
             fi.type = "text/javascript";
             fi.async = true;
             fi.src = "http://sm.adelerobots.com/js/servicemng.js";
             var s = document.getElementsByTagName("script")[0];
             s.parentNode.insertBefore(fi, s);
            })();
        //--><!]]></script>';
?>
```

10.2.3.3 Fichero config-form.php

```
<?php

    wp_enqueue_script('tcrf_jscolor', plugins_url('../assets/js/jscolor/jscolor.js',
__FILE__));

    $tcrf_user_params = get_option('tcrf_user_params');

?>

<style>
  a { text-decoration: none; }
  .button-no-bold { font-weight: normal; }
</style>

<div class="wrap">
<h2><a href="http://www.theclientrelationsfactory.com" target="_blank"> The Client
Relations Factory </a></h2>

<p>Your linked account is <?php echo '<strong>' . $tcrf_user_params['username'] .
'</strong>'; ?></p>

<table class="form-table">
    <form method="post" id="fiona_settings">
        <input type="hidden" name="tcrf_action" value="enable-fiona">
        <tr valign="top">
            <th scope="row">Start Button Text</th>
            <td>
                <input type="text" name="tcrf_start_button_text" minlength="1"
maxlength="15"
                    value="<?php echo isset($tcrf_user_params['start_button_text']) ?
$tcrf_user_params['start_button_text'] : "Chat with me!" ?>"
```

```
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " readonly ";
                ?>
                title="Defines the text that will be displayed inside Virtual Robot's
start button"
                >
            </td>
        </tr>
        <tr valign="top">
            <th scope="row">Chat Button Text</th>
            <td>
                <input type="text" name="tcrf_chat_button_text" minlength="1"
maxlength="10"
                    value="<?php echo isset($tcrf_user_params['chat_button_text']) ?
$tcrf_user_params['chat_button_text'] : "Send" ?>"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " readonly ";
                ?>
                title="Defines the text that will be displayed into the Send button of
Virtual Robot chat dialog"
                >
            </td>
        </tr>
        <tr valign="top">
            <th scope="row">Avatar Size</th>
            <td>
                <select name="tcrf_window_size"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " disabled ";
                ?>
                title="Sets the size of your Virtual Robot display frame"
                >
                    <option
                    <?php
                        if (strcasecmp($tcrf_user_params['window_size'], "Small") == 0)
echo " selected ";
                    ?>
                    value="Small">Small
                    </option>
                    <option
                    <?php
                        if (strcasecmp($tcrf_user_params['window_size'], "Big") == 0 ||
!isset($tcrf_user_params['window_size']))
                            echo " selected ";
                    ?>
                    value="Big">Big
                    </option>
                </select>
            </td>
        </tr>
        <tr valign="top">
            <th scope="row">Top bar widget color</th>
            <td>
                <input type="text" name="tcrf_top_bar_color"
                    <?php if (isset($tcrf_user_params['dialog_top_bar_color']))
                        echo 'value="' . $tcrf_user_params['dialog_top_bar_color'] .
'"';
                    ?>
                    class="color {pickerFaceColor:'transparent', pickerFace:3,
pickerBorder:0, pickerInsetColor:'black', slider:false, pickerPosition:'right',
hash:true}"
                <?php
                    if ($tcrf_user_params['fiona_enabled']) echo " disabled ";
                ?>
                title="Sets the color of your Virtual Robot's widget dialog top bar"
                >
            </td>
        </tr>
    </form>
        <tr valign="top">
            <th>
                <form method="post">
                    <input type="hidden" name="tcrf_action" value="logout">
                    <input class="button button-primary button-no-bold" type="submit"
value="Unlink Account" title="Detach your account from the plugin"/>
                </form>
```

```
            </th>
            <?php
                if (!$tcrf_user_params['fiona_enabled'])
                {
            ?>
            <td>
                <button class="button button-primary" class="" title="Enables your
Virtual Robot integration for this site"
onclick='document.getElementById("fiona_settings").submit();'>Enable Virtual
Robot!</button>
            </td>
            <?php
                }
                else
                {
            ?>
            <td>
                <form method="post">
                    <input type="hidden" name="tcrf_action" value="disable-fiona">
                    <input class="button button-primary" type="submit" value="Disable
Virtual Robot" title="Disables your Virtual Robot's integration for this site"/>
                </form>
            </td>
            <?php
            }
            ?>
        </tr>
    </table>
</div>
```

10.2.3.4 Fichero login-form.php

```
<?php
    wp_enqueue_style('tcrf_login_form_stylesheet', plugins_url('../assets/css/login-
form.css', __FILE__));
?>

<div class="wrap">
<h2><a href="http://www.theclientrelationsfactory.com" target="_blank"> The Client
Relations Factory </a></h2>


<?php
    if ($error_message)
    {
?>
    <div id="login-error">
        <div class="alert-box error-box"><span>error: </span>
        <?php
            if ($error_message)
            {
                echo $error_message;
            }
            else
            {
                echo "Cannot connect with The Client Relations Factory, please try again
in a few minutes. If problem persists, please contact us in
support@theclientrelationsfactory.com";
            }
        ?></div>
    </div>
<?php
    }
?>

<form method="post">
    <input type="hidden" name="tcrf_action" value="login">
    <table class="form-table tcrf-form-table">
        <tr valign="top">
        <th scope="row">Username</th>
        <td><input required type="email" name="tcrf_username" value="" /></td>
        </tr>

        <tr valign="top">
```

```
        <th scope="row">Password</th>
        <td><input required type="password" name="tcrf_password" value="" /></td>
        </tr>
        <tr>
            <th scope="row">

            </td>
            <td>
                <input class="button button-primary" type="submit" value="Link Account!"
title="Link your account and embed your Virtual Robot!"/>
            </td>
        </tr>
    </table>

<div class="tcrf-link">Please Sign In with your The Client Relations Factory
credentials. If you don't have an account yet... <br /><a
href="http://www.theclientrelationsfactory.com/#price" target="_blank" title="Get your
account now!"><p class="tcrf-register-now"><strong>Register now!</strong></p></a></div>

</form>
</div>
```

Capítulo 11. Anexos

11.1 Anexo 1. Datos Métricas

A las páginas siguientes se muestran las métricas cumplimentadas por los usuarios evaluados en este Trabajo Final de Máster.

Las tablas originales pueden verse en el archivo adjunto *DatosMétricas.xlsx*.

Datos del desarrollador	
Nombre	Administrador1
Formación	Máster en Ingeniería Web
Conclusión	Tras haber realizado la instalación de extensiones en los tres Gestores estudiados y en múltiples ocasiones en cada uno, aunque en un principio todos pueden parecer similares en cuanto a funcionamiento, Drupal ha dado más problemas en la instalación. Además, debido a la peculiaridad descrita en el párrafo anterior, se ha desperdiciado una cantidad de tiempo considerable. **Elección: A excepción de Drupal, y basándose en la instalación de extensiones, cualquiera de los otros CMS podría ser válido.**

Métrica para administradores

Criterio	Joomla	Drupal	Wordpress	Comentarios
¿Existe documentación acerca del la instalación de extensiones?	5	5	5	
¿Está disponible en varios idiomas?	0	0	5	
¿Es fácilmente accesible?	3	5	5	
¿Describe con claridad la instalación?	5	5	5	
¿Posee algún sistema de instalación automática o se debe realizar de forma manual? 0 manual, 5 automática	5	3	5	Los 3 CMS estudiados poseen herramientas de instalación para las extensiones, pero concretamente Drupal, es el que más problemas da a la hora de realizarlas, pues la herramienta fallo con facilidad y, según la documentación, no está soportada en todos los servidores.
Una vez instalada la extensión ¿En fácil de activar y configurar?	4	3	5	Wordpress destaca sobre Joomla y Drupal en este apartado, ya que una vez hecha la instalación de la extensión, ofrece automáticamente la posibilidad de activarla. Además, añade en la barra lateral del panel de administración un enlace directo (personalizable por cada desarrollador) a la página de configuración del plugin. Drupal, posee una peculiaridad que puede resultar realmente molesta si no si tiene en cuenta o se olvida, es necesario, una vez instalada la extensión, asignarle una posición en el sitio web a través del menú Estructura.

Datos del desarrollador	
Nombre	Administrador2
Formación	Diseñador gráfico. Conocimientos de HTML, CSS y JavaScript y conocimientos básicos de PHP.
Conclusión	Los tres Gestores son muy similares a la hora de instalar un plugin. El proceso de activación más sencillo es el de Wordpress, ya que tras la instalación muestra directamente la opción de activación del plugin. **Elección: Wordpress**

Métrica para administradores

Criterio	Joomla	Drupal	Wordpress	Comentarios
¿Existe documentación acerca del la instalación de extensiones?	5	5	5	
¿Está disponible en varios idiomas?	0	3	5	Drupal en Drupal Hispano, aunque no está toda la info que hay en inglés
¿Es fácilmente accesible?	5	5	5	
¿Describe con claridad la instalación?	5	5	5	
¿Posee algún sistema de instalación automática o se debe realizar de forma manual? 0 manual, 5 automática	5	5	5	
Una vez instalada la extensión ¿En fácil de activar y configurar?	3	2	5	

Datos del desarrollador	
Nombre	Administrador1
Formación	Diseñador Web, especializado en Frontend Conocimientos de HTML, CSS, JavaScript y PHP y conocimientos básicos de JSPs y Java.
Conclusión	Cualquiera de los tres CMS tiene instalación automática y activación sencilla. **Elección: Cualquiera**.

Métrica para administradores

Criterio	Joomla	Drupal	Wordpress	Comentarios
¿Existe documentación acerca del la instalación de extensiones?	5	5	5	
¿Está disponible en varios idiomas?	0	0	5	
¿Es fácilmente accesible?	5	5	5	
¿Describe con claridad la instalación?	5	5	5	
¿Posee algún sistema de instalación automática o se debe realizar de forma manual? 0 manual, 5 automática	5	5	5	
Una vez instalada la extensión ¿En fácil de activar y configurar?	4	4	5	

Datos del desarrollador	
Nombre	Desarrollador1
Formación	Máster en Ingeniería Web
Conclusión	A la vista de los resultados de las métricas la opción preferida como desarrollador es Wordpress. En el apartado de Documentación, Wordpress es el que posee una referencia más completa y mejor organizada. A su vez en el desarrollo mantiene un buen equilibrio entre complejidad y personalización. Si bien, la manera de ejecutar la funcionalidad del plugin resulta un poco confusa al principio, esto puede deberse a que el desarrollo se a efectuado primero en Joomla y Drupal. Drupal en cambio, es el CMS que más posibilidades ofrece como desarrollador, pero en una primera iteración resulta demasiado complejo. Lo contrario ocurre con Joomla, es muy sencillo, pero muy poco personalizable.**Elección: Wordpress**

Evaluación de la docume

Critero	Joomla	Drupal	Wordpress	Observaciones
¿Existe documentación acerca del CMS?	5	5	5	
¿Existe documentación acerca del sistema de ampliaciones del CMS?	5	5	5	
¿Es fácilmente accesible?	5	4	5	
¿Está disponible en varios idiomas?	5	0	3	El caso de Wordpress es un poco particular, ya que pese a existir información oficial en varios idiomas, la información que se presenta en ellos no es la misma. De la documentación revisada la más completa es sin duda la disponible en inglés.
¿Es concisa? ¿Describe con detenimiento cada método de ampliación, para qué sirve y cómo utilizarlo?	3	4	5	Es uno de los mayores, por no decir el mayor, problema a la hora de comenzar el desarrollo. Hay bastante información de cómo desarrollar extensiones, pero no suficiente como para que un desarrollador novel pueda tomar sin un atisbo de duda la decisión de implementar un tipo u otro de extensión. En el caso de Joomla, tras haber realizado también el desarrollo de las extensiones para Drupal y Wordpress, es posible que la elección de desarrollar un plugin en lugar de un módulo no haya sido la más correcta. Esto podría haberse evitado con una definición más clara de la funcionalidad de cada una de estas extensiones. En el caso de Drupal y Wordpress, la elección es simple, la ampliación de funcionalidad se consigue a través de módulos y plugins respectivamente.
Una vez elegido el método a desarrollar ¿Existe información suficiente para el desarrollo sin recurrir a fuentes externas?	4	4	5	Pese a que Joomla y Drupal tienen documentación muy buena sobre el desarrollo de sus ampliaciones, cabe destacar la labor de Worpress en la elaboración del Plugin Handbook. Dicho manual proporciona información detalladísima de cómo implementar prácticamente todos los aspectos de un plugin.
¿Está dicha información bien organizada? ¿Dispone de un índice claro o un mapa web desde el que tener una visión global de toda la disponible?	1	3	5	
¿Dispone de tutoriales para la creación de ampliaciones básicas?	4	5	5	

Métricas para desarrolladores					
Documentación	¿Están los tutoriales actualizados a las últimas versiones?	5	4	5	En el caso de Drupal, pese a existir buenos tutoriales en la documentación, al tratarse del CMS, de los contemplados en la comparativa, que más lleva al extremo el lema "desarrollado por la comunidad", los tutoriales están creados por administradores, que en resumen, son desarrolladores "externos", y son bastante diferentes unos de otros. Al cambiar la versión del CMS, cambia el desarrollador del tutorial, por lo que su estructura o su forma de programar es completamente diferente, y pueden demorarse varios meses.
	¿Dispone de tutoriales para la creación de ampliaciones avanzadas?	2	4	5	
	¿Existe documentación de otras fuentes?	4	4	5	Los 3 CMS que se han comparado, disponen de muchísima información en páginas de terceros que ayudan a desarrollar las extensiones. Concretamente http://www.inmotionhosting.com/support/edu ofrece tutoriales detalladísimos sobre el desarrollo de las mismas. Con solo echar un vistazo rápido a la tabla puede observarse que el CMS en el que se han encontrado más facilidades, teniendo en cuenta la información disponible y la calidad de la misma, ha sido Wordpress. La documentación de Joomla ha sido, quizás, la más confusa de las 3, en gran parte debido a la falta de documentación inicial debido a los varios métodos de desarrollo de ampliaciones que dispone. Un funcionalidad que podría ser un factor de éxito para este CMS, se convierte en un inconveniente por la falta de documentación que existe. La documentación de Drupal está claramente desarrollada por desarrolladores para desarrolladores. Pese a estar mejor organizada que la de Joomla, es más compleja de seguir, y requiere de bastantes conocimientos previos para llevar a cabo con éxito el desarrollo de un módulo.

Evaluación de la personalizac	Criterio	Joomla	Drupal	Wordpress	Observaciones
	¿Resulta sencillo crear una página básica de configuración?	5	4	4	
	¿Resulta sencillo ampliar y/o modificar dicha página?	3	5	5	
	¿Es sencillo de configurar sin conocimientos técnicos adicionales?	5	5	5	
	¿Es posible proporcionar diferentes campos de entrada al usuario?	4	5	5	
	¿Es configurable la disposición de los elementos en la página?	2	5	5	
	¿Permite la inclusión de código HTML externo?	0	3	5	Este es un de los puntos en los que más difieren los tres CMS estudiados, como se ha visto, la página de configuración de Joomla se configura completamente desde un XML y la de Wordpress en HTML, en cambio Drupal, genera los campos de entrada te datos del usuario mediante funciones predeterminadas, pero también permite la inclusión de HTML externo para otras funcionalidades.

ión	¿Permite la inclusión de JavaScript?	0	5	5

Evaluación de los métodos de desarrollo

Criterio	Joomla	Drupal	Wordpress	Observaciones
¿Es el sistema de asociación evento-funcionalidad fácilmente entendible?	5	5	3	
¿Es sencillo asociar ejecución de código a eventos del CMS?	5	5	3	
¿Se encuentra bien diferenciado el código asociado a funciones propias con el de funciones asociadas a eventos de manera predeterminada por el CMS?	3	3	1	En wordpress, a diferencia de Joomla y Drupal, el código del plugin se ejecuta, estando el plugin activo, cada vez que se muestra una página. Esto hace que haya que definir funciones, como por ejemplo la función tcrf_init(), que se encarguen de asociar el código a los eventos del CMS. En Joomla y Drupal en cambio, mediante una convención de nombres, las funciones se asocian automáticamente a los eventos. Por ejemplo en Joomla la función onContentPrepare() se ejecuta cada vez que se "lanza" el evento que prepara el contenido de la página.
¿Se pueden ejecutar todas las funcionalidades deseadas en las funciones asociadas a los eventos del CMS?	5	5	5	
¿Se puede modificar el código sin necesidad de reinstalar el plugin?	0	0	5	Uno de los aspectos que más se agradecen en Wordpress a la hora de desarrollar es el hecho de que posee en editor de código, de manera que se pueden editar los ficheros del plugin sin necesidad de reinstalarlo, lo que proporciona muchísima fluidez al desarrollo.

Datos del desarrollador	
Nombre	Desarrollador2
Formación	Diseñador gráfico. Conocimientos de HTML, CSS y JavaScript y conocimientos básicos de PHP.
Conclusión	Desde el punto de vista de desarrollo, Joomla parece más sencillo para desarrollar las funcionalidades, pero más complejo para crear la página de configuración por el lenguaje utilizado, admás permite muy poca personalización. Drupal resulta parece demasiado complejo. Wordpress es el que se aproxima más al nivel técnico de un Diseñador. Elección: Wordpress

Métricas para desarrolladores

Evaluación de la documentación

Critero	Joomla	Drupal	Wordpress	Observaciones
¿Existe documentación acerca del CMS?	5	5	5	
¿Existe documentación acerca del sistema de ampliaciones del CMS?	5	5	5	
¿Es fácilmente accesible?	4	3	4	
¿Está disponible en varios idiomas?	5	0	2	
¿Es concisa? ¿Describe con detenimiento cada método de ampliación, para qué sirve y cómo utilizarlo?	2	3	4	
Una vez elegido el método a desarrollar ¿Existe información suficiente para el desarrollo sin recurrir a fuentes externas?	3	1	4	
¿Está dicha información bien organizada? ¿Dispone de un índice claro o un mapa web desde el que tener una visión global de toda la disponible?	1	3	5	
¿Dispone de tutoriales para la creación de ampliaciones básicas?	4	4	4	
¿Están los tutoriales actualizados a las últimas versiones?	5	4	5	
¿Dispone de tutoriales para la creación de ampliaciones avanzadas?	0	0	0	No evaluado
¿Existe documentación de otras fuentes?	5	5	5	

Evaluación de la personalización

Criterio	Joomla	Drupal	Wordpress	Observaciones
¿Resulta sencillo crear una página básica de configuración?	0	3	5	Se hace todo por XML, lenguaje que apenas manejaba.
¿Resulta sencillo ampliar y/o modificar dicha página?	0	3	5	
¿Es sencillo de configurar sin conocimientos técnicos adicionales?	0	0	5	Solamento wordpress se puede configurar solamente con HTML
¿Es posible proporcionar diferentes campos de entrada al usuario?	4	5	5	
¿Es configurable la disposición de los elementos en la página?	0	0	5	
¿Permite la inclusión de código HTML externo?	0	3	5	
¿Permite la inclusión de JavaScript?	0	5	5	

Evaluación de los métodos de desarrollo

Criterio	Joomla	Drupal	Wordpress	Observaciones
¿Es el sistema de asociación evento-funcionalidad fácilmente entendible?	5	5	3	
¿Es sencillo asociar ejecución de código a eventos del CMS?	5	5	3	
¿Se encuentra bien diferenciado el código asociado a funciones propias con el de funciones asociadas a eventos de manera predeterminada por el CMS?	3	3	0	Después de hacer el desarrollo para Joomla y Drupal, es dificil el cambio al sistema de Wordpress.
¿Se pueden ejecutar todas las funcionalidades deseadas en las funciones asociadas a los eventos del CMS?	5	5	5	
¿Se puede modificar el código sin necesidad de reinstalar el plugin?	0	0	5	Gran ventaja en Wordpress

Datos del desarrollador	
Nombre	Desarrollador3
Formación	Diseñador Web, especializado en Frontend Conocimientos de HTML, CSS, JavaScript y PHP y conocimientos básicos de JSPs y Java.
Conclusión	Drupal parece tener unas posibilidades de desarrollo muy superiores a Joomla y Wordpress. Aunque la documentación sobre desarrollo no es muy amplia ni clara en Drupal existe muchísma información y tutoriales de fuentes externas. **Elección: Drupa**l

Metricas para desarrolladores

Evaluación de la documentación

Critero	Joomla	Drupal	Wordpress	Observaciones
¿Existe documentación acerca del CMS?	5	5	5	
¿Existe documentación acerca del sistema de ampliaciones del CMS?	5	5	5	
¿Es fácilmente accesible?	4	4	5	Plugin Handbook
¿Está disponible en varios idiomas?	5	0	3	
¿Es concisa? ¿Describe con detenimiento cada método de ampliación, para qué sirve y cómo utilizarlo?	3	3	4	
Una vez elegido el método a desarrollar ¿Existe información suficiente para el desarrollo sin recurrir a fuentes externas?	3	2	3	Hay más información en fuentes externas
¿Está dicha información bien organizada? ¿Dispone de un índice claro o un mapa web desde el que tener una visión global de toda la disponible?	0	2	4	
¿Dispone de tutoriales para la creación de ampliaciones básicas?	5	5	5	
¿Están los tutoriales actualizados a las últimas versiones?	5	5	5	
¿Dispone de tutoriales para la creación de ampliaciones avanzadas?				No evaluado
¿Existe documentación de otras fuentes?	5	5	5	

Evaluación de la personalización

Criterio	Joomla	Drupal	Wordpress	Observaciones
¿Resulta sencillo crear una página básica de configuración?	5	3	4	Joomla es el más sencillo pero el menos personalizable
¿Resulta sencillo ampliar y/o modificar dicha página?	5	5	5	
¿Es sencillo de configurar sin conocimientos técnicos adicionales?	5	5	5	
¿Es posible proporcionar diferentes campos de entrada al usuario?	3	5	5	Limitado al ser solo mediante XML
¿Es configurable la disposición de los elementos en la página?	3	4	5	
¿Permite la inclusión de código HTML externo?	0	3	5	Wordpress da total libertad
¿Permite la inclusión de JavaScript?	0	5	5	

Evaluación de los métodos de desarrollo

Criterio	Joomla	Drupal	Wordpress	Observaciones
¿Es el sistema de asociación evento-funcionalidad fácilmente entendible?	5	5	5	
¿Es sencillo asociar ejecución de código a eventos del CMS?	3	5	5	Cuesta un poco encontrar el evento que se necesita
¿Se encuentra bien diferenciado el código asociado a funciones propias con el de funciones asociadas a eventos de manera predeterminada por el CMS?				No entiendo la pregunta
¿Se pueden ejecutar todas las funcionalidades deseadas en las funciones asociadas a los eventos del CMS?	5	5	5	
¿Se puede modificar el código sin necesidad de reinstalar el plugin?	0	3	5	Integrado en Wordpress y con extensiones en Drupal