UNIVERSIDAD POLITECNICA DE VALENCIA

Escuela Técnica Superior de Ingeniería Informática

Intranet para la explotación de productos software

Proyecto Final de Carrera

Presentado por: Enrique Belda Moscardó

Dirigido por:

Tutor de la empresa : Carlos Fernández Martínez

Tutor UPV : Eva Vallada Regalado

24/03/2011

Índice

Índice de figuras

Índice de Tablas

1 Introducción

Desde que empezó a comercializarse el *software* como producto informático, las empresas siempre han buscado la forma de poder ofrecer un buen soporte técnico a sus clientes, ya que la facilidad para aprender a utilizar un producto, así como la rapidez a la hora de resolver problemas derivados de su uso, a menudo han marcado el éxito o fracaso del producto en sí.

En anteriores décadas era frecuente que la única ayuda que dispusiera un usuario era un simple manual de uso y una línea telefónica para casos más concretos. Con el tiempo, el desarrollo de Internet y el incremento de usuarios con acceso a la red de redes ha facilitado el modo en el que las empresas desarrolladoras de *software* ofrecen soporte para sus productos, en forma de actualizaciones para resolver fallos en la programación, foros donde los usuarios expertos o desarrolladores ayudan a los usuarios iniciados, vídeos con casos de uso, chats, etc... Actualmente muchas de las empresas dedicadas a la venta y/o explotación de productos *software* disponen de una página web en la que ponen a disposición de sus clientes una serie de elementos diversos para poder prestarles ayuda de forma personalizada, estamos hablando del concepto de Intranet.

Una Intranet es una red de computadores utilizada en una organización de carácter privado que a través de Internet pone a disposición de los usuarios una serie de servicios y recursos informáticos. Dependiendo del propósito de la Intranet y el tamaño de la organización el número de computadores puede variar de un sólo servidor a cientos de computadores y/o servidores. La disponibilidad de los recursos depende del tipo de usuario, al que se le concederán una serie de privilegios dependiendo de la función que deba desempeñar dicho usuario en la Intranet. El acceso a los recursos suele estar jerarquizado y organizado en grupos, existiendo un grupo de usuarios cuyos privilegios le dan permiso de acceso a todos los recursos, limitándose estos privilegios según se descienden niveles de la jerarquía. Para poder acceder a la Intranet los usuarios tienen que estar registrados y deben identificarse mediante un nombre de usuario y una contraseña.

El presente Proyecto Final de Carrera (PFC) se realiza en el marco de prácticas en empresa. La empresa en cuestión es el Instituto Tecnológico de Informática (ITI), en adelante la empresa, que tiene su sede en la Universidad Politécnica de Valencia. Uno de los grupos de investigación la empresa, más concretamente, el grupo de Sistemas de Optimización Aplicada (SOA) tiene en explotación una serie de productos en un conjunto importante de clientes. La creciente cantidad de clientes conlleva un importante trabajo en lo que respecta al mantenimiento del producto y de los clientes. Por ello, dicho departamento plantea la creación de una Intranet en la que centralizar diversos aspectos como los documentos de ayuda, soporte técnico, preguntas frecuentes, actualizaciones, etc... Esta centralización de servicios tendrá que tener en cuenta la diversidad de licencias y contratos que tenga cada uno de los clientes.

El objetivo de este PFC se centra en el desarrollo de una Intranet haciendo uso de un *Content Management System* (CMS) para dar soporte a un conjunto de clientes que tengan contratados unos productos *software*. El soporte se prestará de diversas maneras y siempre teniendo en cuenta la diversidad de usuarios. La Intranet estará centralizada en un servidor y los clientes serán usuarios que deben agruparse en función de los productos que tengan contratados. Un CMS es una herramienta *software* que facilita el desarrollo de una página web proporcionando una interfaz para administrar los recursos, lo que puede ser una gran ayuda a la hora de desarrollar una Intranet.

La estructura que vamos a seguir es la siguiente. En la Sección 2 se realizará un estudio de los CMS existentes actualmente. En la Sección 3 se presentará en detalle el CMS elegido para poder desarrollar la Intranet. Proseguiremos con la Sección 4 en el que presentaremos los requisitos que requiere la Intranet así como las distintas modificaciones que tendrán que llevarse a cabo sobre el CMS. En la Sección 5 se presentan unos casos de uso de la Intranet. Por último, en la Sección se comentan las conclusiones.

2 Estudio de un CMS

Un CMS es un sistema de gestión de contenidos, esto es una herramienta que sirve para la creación y administración de contenidos generalmente de páginas web. La función principal es separar el diseño de los contenidos que generalmente son administrados a través de una o varias bases de datos alojadas en el mismo servidor del CMS. Esto se hace a través de una interfaz administrativa también llamada *back-end* mediante la cual un usuario con privilegios de administrador gestiona los contenidos que serán mostrados en el sitio web o *front-end*.

En la última década han surgido una gran cantidad de CMS. Las características varían mucho de uno a otro y algunos son más adecuados para ciertas funciones. Podemos distinguir algunos tipos de CMS según ciertas características:

- **Según el lenguaje de programación empleado**:

Existen dos tipos de lenguajes de programación orientados a Internet: los que se ejecutan en el servidor que aloja la página y los que son interpretados por el navegador y se ejecutan en el cliente que la visita. Los CMS deben usar ambos lenguajes para adquirir una funcionalidad completa. Del lado del servidor destacamos como lenguajes más usados en los CMS Java, PHP, ASP.NET y Python entre otros. Del lado del cliente destacamos HTML, JavaScript y AJAX.

- **Según la propiedad del código fuente**:

Los CMS, así como cualquier producto *software* pueden publicarse mediante una licencia privativa o una licencia libre. Las licencias libres permiten que nuevos desarrolladores modifiquen el código fuente original de la aplicación con la idea de mejorar o ampliar el producto. Entre este tipo de licencias destaca principalmente la licencia GPL (*General Public License*)[1]. Las licencias privativas por su parte se caracterizan por no entregar el código fuente junto con la aplicación y por tanto sólo permite modificaciones al código original por parte de su autor. Su función principalmente es mantener la dependencia del cliente con el desarrollador.

- **Según el tipo de uso o funcionalidades:**

Los CMS también podemos agruparlos en función de la finalidad con la que fue diseñado. Entre ellos los usos mas comunes son:

  - Plataformas generales: Sin un fin específico, pensados para facilitar la construcción de cualquier tipo de página web.
  - Blogs: pensados para páginas personales.
  - Foros: pensados para compartir opiniones.
  - Wikis: pensados para el desarrollo colaborativo.
  - E-learning: plataforma para contenidos de enseñanza on-line.
  - E-commerce: plataforma de gestión de usuarios, catálogo, compras y pagos.
  - Publicaciones digitales: pensados para periódicos.
  - Difusión de contenido multimedia**.**

2.1 Análisis de CMS

La empresa tiene la intención de poner en marcha una Intranet y para ello será necesario un CMS que pueda dar permiso de acceso a los recursos de la Intranet de forma individual. También, se espera del CMS que sea flexible y se pueda aumentar su funcionalidad mediante la instalación de extensiones. Todo esto debe poder hacerse de forma fácil e intuitiva. Por tanto será necesario hacer un análisis de los distintos CMS disponibles para ver cuales reúnen las siguientes características:

- **Código abierto**

Esto es importante, los CMS de código abierto tienen la posibilidad de desarrollar extensiones personalizadas que cumplan aquellas funciones que el propio CMS no aporta. Estas extensiones bien pueden ser desarrolladas por la comunidad de usuarios y/o desarrolladores del CMS o bien podemos desarrollarlas nosotros mismos si no hay alguna extensión ya hecha que satisfaga nuestros requisitos. Además, la mayoría son a coste cero con lo que se ahorra en el coste anual de las licencias.

- **Comunidad de usuarios**

Una amplia comunidad de usuarios es de suma importancia para un CMS de código abierto. Gracias a disponer de una comunidad con miles de desarrolladores algunos CMS son mejorados constantemente, se publican nuevas versiones cada pocos meses y además los usuarios crean sus propias extensiones que luego ponen a disposición del resto y el CMS gana mucha flexibilidad y funcionalidad. En nuestro CMS va a ser necesario instalar algunas extensiones y se espera programar lo mínimo posible, así que es conveniente que el CMS goce de popularidad y de una amplia comunidad de usuarios que nos brinden una buena gama de extensiones. También se requiere que haya amplia documentación sobre el código del CMS y sobre como desarrollar nuevas extensiones, esto puede ser crucial en un momento dado si no se encuentran extensiones que nos aporten toda la funcionalidad que deseamos.

Administración de usuarios

Es necesario que el CMS pueda dar de alta nuevos usuarios y almacenar la información básica sobre ellos. También sería deseable que se puedan crear grupos a los que vincular los usuarios que compartan ciertas características.

Control de acceso

Puesto que la Intranet debe personalizar los contenidos según el usuario que se conecte es necesario que el CMS permita un control de acceso granulado, es decir, que los permisos deben poder concederse a los usuarios sobre un conjunto de recursos o sobre recursos individuales según el grupo al que pertenezca cada usuario.

- **Amigable**

Es necesario que el CMS sea fácil de aprender a usar y que disponga de una buena documentación para que el administrador aprenda con rapidez las tareas que debe desempeñar para el correcto funcionamiento de la página. Es necesario por tanto analizar su usabilidad web[2].

- **Publicidad**

Además de dar servicios a los clientes para los productos que tienen contratados se plantea la posibilidad de captar nuevos clientes a través de la Intranet. Para este cometido sería interesante que el CMS diera facilidades para mostrar en la web publicidad sobre los nuevos productos que se publiquen.

- **Facilidad de instalación**

Sería interesante que el CMS se pueda instalar de manera fácil e intuitiva a través de una interfaz de usuario. Algunos CMS pueden instalarse a través del navegador siguiendo una serie de pasos bien explicados. Debemos encontrar un CMS cuya instalación y configuración no cuesten demasiado tiempo.

2.2 Estudio en detalle de CMS

Después de analizar las características importantes que debe tener un CMS para desarrollar una Intranet se ha hecho una búsqueda a través de la web para encontrar los CMS que mejor se ajustan. Tras la búsqueda hemos seleccionado para el PFC cuatro CMS que podrían servir para el desarrollo de la Intranet. A continuación pasamos a explicar brevemente las principales características de cada uno de ellos destacando los aspectos positivos y negativos.

- **Typo3**

El proyecto empezó a desarrollarse en Dinamarca por Kasper Skårhøj. Tras varios años de trabajo aparecen tres versiones prototipo en 1998, pero no es hasta el año 2000 cuando aparece una primera versión para evaluación de Typo3, dándose a conocer por primera vez al mundo del *software* libre. Con la colaboración de una creciente comunidad de usuarios y desarrolladores en 2002 aparece la primera versión estable del producto.

Es una herramienta de gestión de contenido muy completa. Permite desarrollar completamente un sitio web de contenidos, con todas las consecuencias que conlleva: estructura multinivel, motor de búsquedas, gestión de autoría y publicación de contenidos, mecanismo de uso de plantillas para la maquetación de páginas, etc.

Typo3 es también un portal. Administra, en particular, la personalización de las páginas según la identidad de los usuarios, es decir sabe integrar una selección de contenidos en una misma página, según los derechos del usuario identificado.

Está desarrollado en PHP y es extensible por módulos. Pueden ser módulos de gestión en el interfaz de administración, como gestión básica MySQL o gestión de servidor LDAP, o módulos orientados a la parte del usuario, incluidos en el sitio como podrían ser encuestas, foros, calendario, noticias, búsquedas, etc...

Como aspectos positivos podemos destacar que tiene un control de acceso personalizado, está publicado bajo una licencia GPL, cuenta con una extensa comunidad de alrededor de 20.000 desarrolladores y está disponible en castellano. También es destacable que tiene un componente para crear plantillas y el proceso de instalación del CMS es bastante simple.

Como aspectos negativos debemos destacar que tiene una curva de aprendizaje elevada tanto para administradores como para desarrolladores Tiene una interfaz poco intuitiva a la que hay que dedicarle tiempo para familiarizarse con el sistema. Por otro lado el componente para crear plantillas usa un lenguaje propietario llamado TypoScript.

Puede descargarse de forma gratuita de la página oficial de Typo3: http://typo3.org/download/

En la Figura 1 podemos ver una imagen de la interfaz administrativa del CMS Typo3

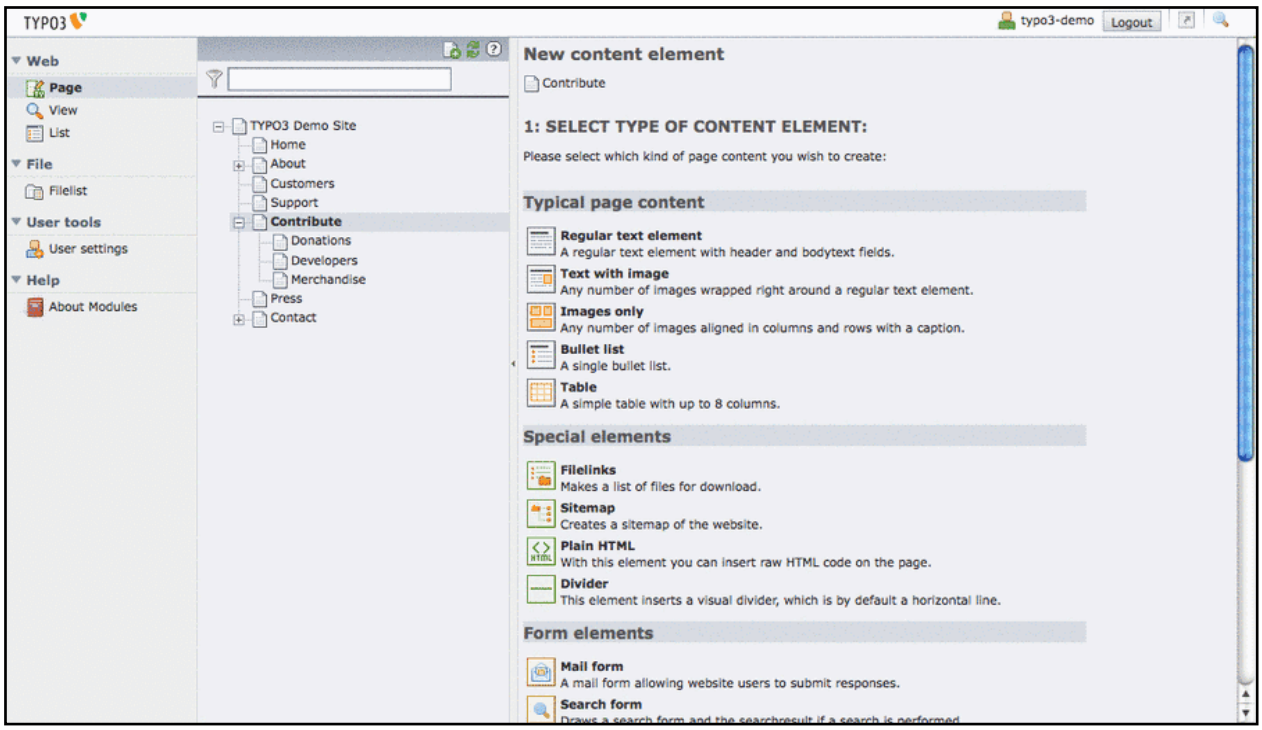

Figura 1. Interfaz administrativa Typo3

- **WordPress**

Wordpress es una evolución de un CMS previo llamado B2/Cafelog desarrollado por Michel Valdrighi que fue publicado en 2001. Poco después Matt Mullenweg y Mike Little comenzaron a desarrollar un fork de este sistema de blogs. La primera versión que vio la luz

fué la WordPress 0.7 en Mayo de 2003, y tuvo una gran acogida, fue reconocido como el mejor CMS para blogs desarrollado hasta la fecha.

Aunque es un CMS enfocado a la creación de blogs, su flexibilidad permite crear cualquier tipo de página web que se desee. Está Desarrollado en PHP y MySQL, bajo licencia GPL y código modificable.

Podemos destacar positivamente entre otros aspectos, su licencia GPL, su facilidad de uso con un corto tiempo de aprendizaje y sus características como gestor de contenidos. Otro aspecto a considerar sobre su éxito y extensión, es la enorme comunidad de desarrolladores y diseñadores, que se encargan de desarrollarlo en general o crear *plugins* y temas para la comunidad, siendo usado en septiembre de 2009 por 202 millones de usuarios, aunque gran parte de ellos no son desarrolladores.

Sin embargo, debemos destacar como aspectos negativos las frecuentes vulnerabilidades en su código que en mayo de 2007 habían dejado en un estado vulnerable el 98% de los servidores que usaban WordPress. Otro aspecto negativo es que es un CMS con una orientación muy marcada a la creación de blogs.Gran parte de su potencial está enfocado a la publicación de contenidos que no es precisamente una característica necesaria en el CMS que buscamos.

Se puede obtener una versión en español en http://es.wordpress.org/

En la Figura 2 podemos ver la interfaz administrativa de WordPress, en concreto la sección de edición de un artículo.

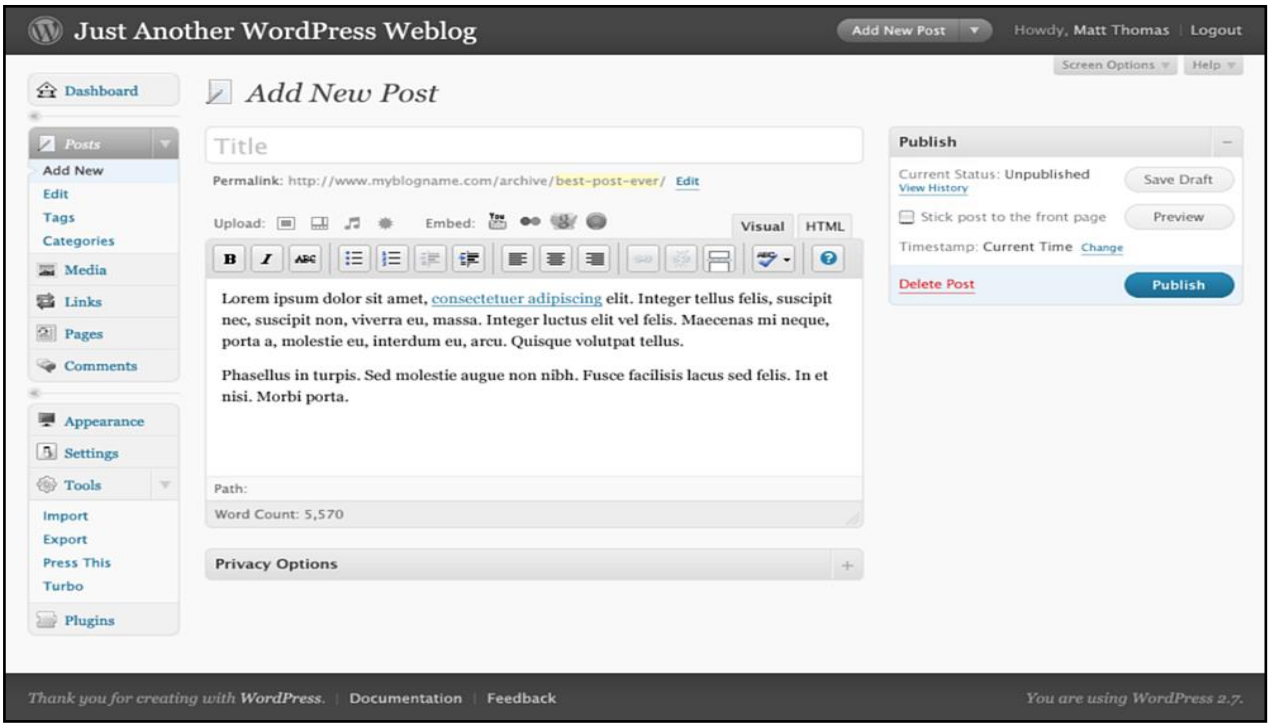

Figura 2. Interfaz administrativa WordPress

- **Drupal**

Dries Buytaert, un informático belga y autor original de Drupal, comenzó a desarrollar un gestor de contenidos para páginas dinámicas en el año 2000. Cinco años más tarde, Drupal había crecido hasta convertirse en un gestor de contenido maduro y flexible, con cientos de desarrolladores trabajando en sus mejoras y extensiones. La versión actual es la 6 y se está trabajando en la 7.0 que puede ver la luz en 2011. Drupal cuenta ya con una comunidad de más de 100.000 usuarios y desarrolladores.

Es un CMS modular multipropósito, muy configurable, que permite publicar artículos, imágenes, u otros archivos y otros servicios como foros, encuestas, votaciones, blogs, etc...

Es un programa libre con licencia GNU/GPL, escrito en PHP, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la web, y un énfasis especial en la usabilidad y consistencia de todo el sistema. Es independiente de la base de datos, soporta MySql, PostgreSQL y puede ser extendido para soportar otras bases de datos. El diseño de Drupal es especialmente idóneo para construir y gestionar comunidades en Internet. No obstante, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hace que sea adecuado para realizar muchos tipos diferentes de sitio web.

Como aspectos positivos destacamos la extensa comunidad de usuarios, la licencia GPL, el soporte para distintas bases de datos, y la existencia de gran cantidad de módulos que permiten crear cualquier tipo de web. Otro aspecto importante es la administración de usuarios que permite agruparlos y dar permisos colectivos de forma fácil sin hacer uso de ningún módulo.

Como parte negativa tenemos que la administración en Drupal tiene una curva de aprendizaje elevada y el panel de control de la parte administrativa no es tan fácil e intuitivo como otros CMS. También tiene un sistema de instalación algo complicado en el que se hace necesario introducir comandos por consola.

Puede obtenerse la última versión de la página oficial del proyecto : http://drupal.org/project/drupal

En la Figura 3 podemos ver una imagen de la interfaz administrativa de Drupal.

Figura 3. Interfaz administrativa Drupal

- **Joomla**

Joomla surge como el resultado de una escisión del proyecto original llamado Mambo. Mambo nace de la corporación Miro de Australia, quien mantenía el grupo principal de desarrolladores. Joomla nace con esta división el 17 de agosto de 2005. La corporación Miro formó una organización sin ánimo de lucro con el propósito inicial de fundar el proyecto y protegerlo de pleitos. El 1 de septiembre de 2005, el nuevo nombre, "Joomla", que es la pronunciación en inglés de la palabra Jumla que significa "todos juntos" o "en su conjunto" .

Es un sistema de gestión de contenidos, y entre sus principales virtudes está la de permitir editar el contenido de un sitio web de manera sencilla. Tiene un editor de contenidos muy potente que usa la tecnología WYSIWYG (*What You See Is What You Get)*, que permite editar texto con formato sin conocimientos de HTML. También destaca por tener una interfaz de instalación guiada muy sencilla, una interfaz administrativa muy intuitiva que permite hacerse con la aplicación muy rápidamente. Tiene una amplia comunidad de usuarios que cuenta con más de 100.000 desarrolladores y existen gran variedad de extensiones para añadir funcionalidad al sistema.

Es una aplicación de código abierto programada mayoritariamente en PHP bajo una licencia GPL. Este administrador de contenidos puede trabajar en Internet o Intranets y requiere de una base de datos MySQL, así como, preferiblemente, de un servidor HTTP Apache.

Destacamos como los aspectos más positivos la fácil instalación, así como facilidad para instalar cualquier extensión para el CMS, la extensa comunidad de usuarios, la facilidad de uso de la interfaz administrativa, la licencia GPL y el potente editor de contenidos.

Como aspectos negativos destacamos la rigidez de su control de acceso, no permitiendo crear grupos de usuarios sin instalar alguna extensión que lo facilite.

Podemos descargar la última versión de Joomla desde http://www.joomlaspanish.org/

En la siguiente imagen podemos ver el menú principal del *back-end* de Joomla.

Figura 4. Interfaz administrativa de Joomla.

A modo de resumen general, en la siguiente tabla mostramos los distintos CMS analizados, así como si tienen o carecen de las principales características que nos interesan..

	Software libre	Comunidad de usuarios	Tiempo de aprendizaje	Control de acceso	Facilidad de instalación
Joomla	Sí	100000	Muy corto	Rígido	Muy sencilla
Drupal	Sí	80000	Largo	Flexible	Compleja
Typo3	Sí	20000	Largo	Flexible	Sencilla
Wordpress	Sí	102000000	Corto	Rígido	Dificultad media

Tabla 1. Características de los CMS

Podemos ver que los cuatro CMS estudiados son *software* libre, esta era una condición necesaria para elegir un CMS por lo que sólo hemos analizado CMS con esta propiedad. De ahí que hayamos excluido aquellos CMS de pago como podrían ser CubeCart o ASPapp.

Por lo que respecta a la comunidad de usuarios, el CMS que destaca muy por encima de los demás es WordPress, pero al ser un CMS orientado principalmente a blogs, la gran mayoría de estos usuarios no se espera que contribuyan a su desarrollo. Después de WordPress destacaría Joomla, un CMS de propósito algo más general cuya comunidad se supone más colaborativa. De cerca le seguiría Drupal y en último lugar estaría Typo3.

En cuanto al tiempo de aprendizaje de la interfaz administrativa Joomla es el más destacado, Drupal y Typo3 salen perdiendo en este aspecto con interfaces algo confusas, WordPress quedaría en segundo lugar.

En el control de acceso los más aventajados son Drupal y Typo3, esto está relacionado con la dificultad de su interfaz administrativa, ambos permiten crear grupos de usuarios y concederles permisos sobre los contenidos y los módulos con la instalación básica de la aplicación. Joomla y WordPress no tienen un control de acceso tan flexible lo que les ha permitido tener una interfaz más amigable.

Finalmente en cuanto a la instalación de la aplicación y las extensiones de ésta, destacamos especialmente Joomla que puede instalarse a través de una interfaz muy simple, la cual permite la migración desde un archivo de respaldo y la instalación de extensiones se hace a través de una interfaz en la que el usuario tan sólo tiene que indicar el paquete de instalación de la extensión. Typo3 es también bastante simple a la hora de instalarlo, aunque no tan rápido como Joomla, ya que la instalación de los módulos requiere algunos pasos más. Por otro lado Drupal presenta algunos problemas de instalación, ya que es necesario introducir algunos comandos por consola y en WordPress hay que modificar algunos ficheros de configuración.

Despúes de analizar los CMS detenidamente finalmente hemos decidido usar Joomla por ser el CMS que mejor se ajusta a los requisitos.

Entre los aspectos positivos que han determinado la decisión de usar Joomla destacamos los siquientes:

- Es un CMS de código abierto con licencia GPL y programado en PHP. De esta manera podremos llevar a cabo todas aquellas adaptaciones que sean necesarias para poder ajustar el CMS a las necesidades de la empresa . Esto es un aspecto que cumplen los cuatro CMS analizados pero debemos destacarlo por ser muy importante.

- La existencia de una gran comunidad de usuarios y desarrolladores nos permitirá resolver aquellas cuestiones que se nos planteen además de poder mantener la aplicación en el futuro. En este aspecto sólo es superado por WordPress aunque debemos recordar que el grado de participación de los usuarios de Joomla es mucho mas alto.

- En el aspecto de la parte administrativa es sin duda el más destacado y la facilidad para aprender el uso del sistema es un aspecto crucial en el CMS.

- En cuanto al control de acceso se puede decir que es la principal debilidad de Joomla con respecto a los demás CMS, sin embargo los desarrolladores de Joomla publican nuevas versiones cada poco tiempo. La versión 1.6 se encuentra en fase de desarrollo y se espera una versión estable pronto con un control de acceso granulado muy mejorado.

- Una razón de peso que debemos añadir es que la empresa ya tiene experiencia con este CMS y lo ha sugerido como posible opción.

3 Joomla

En este apartado vamos a describir brevemente como funciona Joomla, las funcionalidades de que consta el paquete de instalación básico, qué podemos aprovechar y también se comentarán aquellas funcionalidades que es necesario implementar porque no ofrece Joomla.

Joomla funciona sobre una base de datos MySQL y un servidor HTTP, usualmente Apache, está escrito en PHP y es de código abierto. Esto significa que podemos programar aquellas funcionalidades que sean necesarias y Joomla no aporta. Además para este cometido Joomla pone a disposición de los desarrolladores el conjunto de librerías, clases y funciones, conjunto usualmente llamado *framework*[3], que se han usado para programar el CMS. Podemos encontrar la API(*Aplication Programm Interface)*[4] de dicho *framework* en http://api.joomla.org/li_Joomla-Framework.html.

Para organizar los contenidos disponemos de una parte administrativa también llamada *back-end* y para mostrarlos a los usuarios está la parte del sitio web o *front-end.* Para organizar la presentación, tanto en el *back-end* como en el *front-end*, disponemos de cuatro tipos de programas que en Joomla se llaman extensiones, algunas ya incluidas en el paquete básico de instalación. Los tipos de extensiones que pueden añadir son plantillas, componentes, módulos,*plugins* .

3.1 Estructura

- **La plantilla:**

La plantilla es la estructura de la página, la función que desempeña es básicamente dividir la página web en distintas zonas mediante una hoja de estilo en cascada[5], un archivo con extensión CSS(*Cascading style sheet)*. En las distintas zonas en las que se divide la plantilla se visualizarán los distintos módulos y componentes que habilitemos desde el *back-end*. La plantilla suele estar escrita en HTML con sentencias en PHP para cargar los módulos y/o componentes. También puede incluir algún script en JavaScript. Se usan plantillas distintas para el *front-end* y para el *back-end*. El paquete básico de instalación cuenta con algunas plantillas para el *front-end* y una sola para el *back-end*, pero se pueden encontrar una infinidad tanto gratuitas como de pago por Internet. Se ubican en el directorio *"…/joomla/templates/"* (las del *front-end*) y en *"…/joomla/administrator/templates/"* (las del *back-end*).

En la Figura 5 podemos ver el gestor de plantillas y la previsualización de una plantilla. Las partes que vemos en rojo son las posibles zonas donde pueden mostrarse los módulos. La zona central, donde pone "Bienvenidos a la portada" es donde se mostraría el componente.

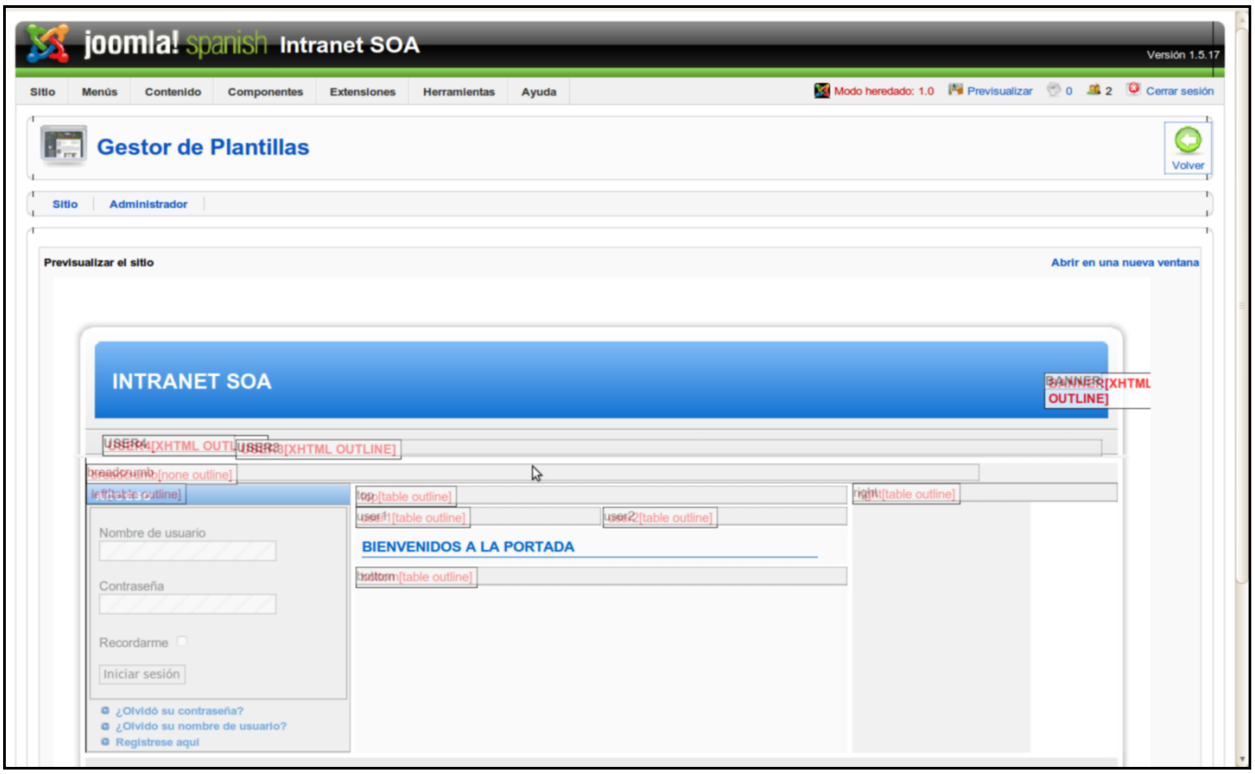

Figura 5. Previsualización de una plantilla.

- **Los componentes:**

Los componentes son programas que la plantilla se encarga de ubicar en la zona central de la página, son por tanto la parte más importante de ésta. Un componente puede existir sólo en el *back-end*, sólo en el *front-end* o en ambos sitios. En el *back-end* suelen usarse como herramientas para gestionar los datos de los que se encargue dicho componente.En el *front-end* para presentarlos en la zona central de la página. Para hacer uso de un componente hay que escribir en la barra de dirección del navegador: *"url_del_dominio"/index.php?option=com_"componente"* en el *front-end* y *"url_del_dominio"/administrator/index.php?option=com_"componente"* en el *back-end*. La ubicación de los archivos y carpetas de los componentes es ".../joomla/components/" (los componentes del *front-end*) y *".../joomla/administrator/components/"* (los del *back-end*).

En la Figura 6 mostramos la sección de componentes del gestor de extensiones. En la imagen podemos ver el menú componentes desplegado, que nos muestra una serie de enlaces a los componentes que tenemos instalados. Detrás del menú podemos ver una lista de componentes con sus atributos más relevantes. En esta lista algunos componentes tienen como autor "Joomla! Project", lo que significa que el componente estaba incluido en el paquete de instalación. De los componentes incluidos en este paquete algunos se ven con un tono de color más claro, significa que forman parte del núcleo de Joomla y no pueden ser desinstalados. El resto pueden ser desinstalados desde este panel, tanto los que formaban parte del paquete de instalación como los que fueron añadidos posteriormente.

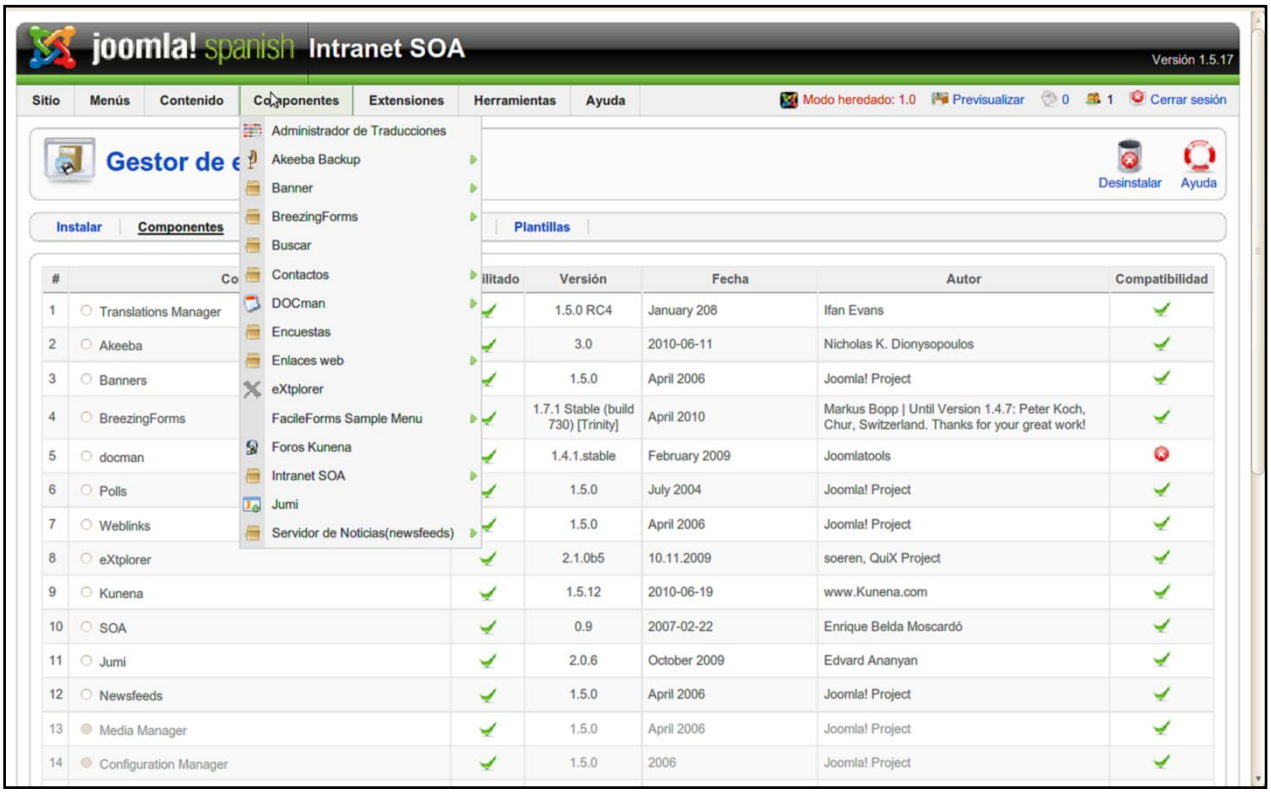

Figura 6. Gestor de extensiones, sección componentes.

- **Los módulos:**

Los módulos son programas que pueden mostrarse en diversos sitios de la página, dependiendo de como esté diseñada la plantilla y como se configure. Por lo general pueden ubicarse en cualquier sitio de la página excepto en la parte central que se reserva para los componentes. Suelen ser programas sencillos encargadas de mostrar enlaces, publicidad, o mostrar algunos datos que no ocupen mucho espacio. Se pueden configurar para que se muestren siempre o se muestren en función del estado de la página. La ubicación de los archivos y carpetas de los módulos es *".../joomla/modules/"* (los del *front-end*) y en *".../joomla/administrator/modules/"* (los del *back-end*).

En la Figura 7 podemos ver las instancias de los módulos del *front-end*, de un módulo instalado podemos crear varias instancias y situarlas en distintos sitios de la plantilla. En la columna "Posición" se indica donde debe mostrarse el módulo y en la columna "habilitado" podemos indicar si queremos que el módulo se muestre o no.

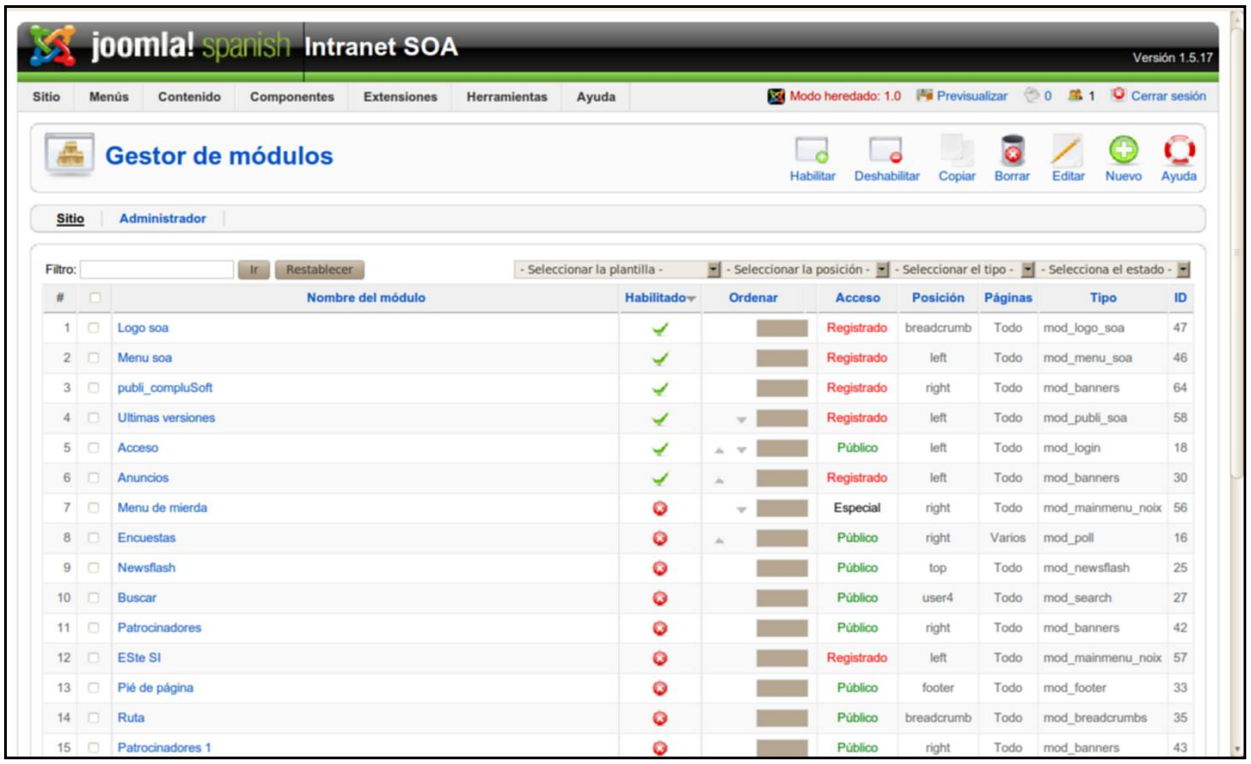

Figura 7. Gestor de módulos.

- **Los *plugins:***

Los *plugins* son extensiones que realizan dentro de Joomla una amplia variedad de funciones relacionadas fundamentalmente con la autenticación de usuarios, el funcionamiento del buscador interno o con la edición de contenidos. Un ejemplo es el editor Wysiwyg TinyMCE con el que podemos editar contenidos desde un entorno más amigable. También el *plugin* Pagebreak que nos permite paginar los artículos. En la serie 1.0 de Joomla los *plugins* se denominaban mambots. En la versión 1.5 han cambiado de nombre al evolucionar también la forma en que se integran en el sistema y han pasado a incluirse entre lo que denominamos extensiones, junto con los módulos y los componentes. La ubicación de los archivos y carpetas de los *plugins* es *"../joomla/plugins/"* y la forma de usarlos varía mucho de un *plugin* a otro.

En la Figura 8 podemos observar el gestor de *plugins* donde se muestra un conjunto de *plugins* instalados. Desde este gestor es posible habilitarlos o deshabilitarlos. Para desinstalarlos habría que ir al gestor de extensiones.

Figura 8. Gestor de plugins

3.2 Instalación

Para la instalación de Joomla se requiere primero instalar un servido http que soporte PHP y una base de datos MySql. Esta instalación se hace usualmente bajo un sistema operativo Linux, y se ha denominado al paquete completo necesario para su instalación tecnología LAMP*(Linux, Apache, MySql, PHP).* Este tipo de tecnología se usa en muchos CMS. Lo primero que hay que hacer para instalar el paquete de Joomla es descargarlo, podemos encontrar una versión en español en http://joomlaspanish.org/ . Una vez obtenido el paquete hay que descomprimirlo en el directorio raíz del servidor http, aunque podemos crear alguna carpeta en dicho directorio y usar dicha carpeta como directorio raíz de Joomla. Una vez hemos descomprimido el paquete, debemos acceder a través del navegador al script de instalación que se situa en la carpeta que hemos fijado como directorio raíz de Joomla. Para ello solo hace falta introducir en la barra de direcciones del navegador la dirección *http://url_instalacion/install.php*. Es un proceso relativamente sencillo, semi-automático y gráfico, en el que, tras cumplir con unos requisitos mínimos, solo hay que seguir unos pocos pasos y cumplimentar algunos detalles desde sus respectivos campos. Es decir, la mayor parte del trabajo lo hace el instalador sí mismo.

Una vez termina la intalación hay que borrar la carpeta *installation* y todo su contenido para concluir el proceso. Una vez finalizado, para acceder a la parte administrativa hay que acceder a la dirección *http://url_instalacion/administrator/*, y para la parte del usuario *http://url_instalacion/*. Puede verse una guía detalla del proceso de instalación en http://comunidadjoomla.org/component/content/article/147-manual-de-instalacion-para-joomla-15x.html?start=8 .

4 Desarrollo

4.1 Los Requisitos

Los requisitos de una Intranet básicamente son que cada usuario vea una página distinta personalizada según su tipo y estado. En nuestro caso, para acceder a la Intranet hay que estar previamente registrado, no permitiremos la entrada a usuarios invitados. Estos usuarios serán por lo general clientes de la empresa dueña del sitio y tendrán acceso a distintos servicios en función de los productos ofrecidos por la empresa que tengan contratados.

Estos servicios principalmente son:

- Acceso a descargar el producto contratado.
- Acceso a información acerca de las licencias de los productos contratados.
- Acceso a una ayuda del producto navegable escrita en HTML y con vídeos.
- Acceso a información y/o descarga de las nuevas versiones de los productos.
- Acceso a foros públicos y/o privados sobre los productos con la idea de dar soporte técnico, detectar errores y mejorar los productos.
- Publicidad acerca de nuevas versiones así como de otros productos que pudieran interesar al cliente.

Para poder ofrecer estos servicios será necesario mantener información acerca de los productos, versiones, clientes, usuarios y licencias. El mantenimiento de esta información es una tarea administrativa y por tanto debe poder realizarse desde el *back-end.*

4.1.1 Requisitos del back-end

- Los clientes pueden ser particulares o pueden ser de una empresa, además necesitaremos mantener información de contacto con nuestros clientes, ya sean particulares o de empresa. También debemos saber las cuentas de usuario asignada a cada cliente que podrían ser una o varias.

- Los productos pueden tener varias versiones, las cuales serán accesibles si se tiene una licencia vigente para dicho producto. De cada producto se desea mantener un artículo de presentación, un logotipo, un foro privado y una ayuda navegable que puede incluir vídeos.

- De cada versión debemos saber: de qué producto es la versión, nombre de la versión, tipo de versión (desarrollo, estable, de prueba, etc..), fecha de la publicación, archivo de descarga, número de descargas efectuadas, si está cerrada o abierta a modificaciones, si continúa su mantenimiento o no, si ha sido lanzada recientemente

y si es de libre distribución o de pago.

- De las licencias deberemos saber: de que cliente y para qué producto es la licencia, numero de licencia para identificarla, tipo de licencia (*lite, extended*, etc...), fecha de inicio de vigencia, fecha que se ordenó la compra, fecha de pago, cantidad pagada, número de equipos que permite y número de renovaciones.

- De las empresas que sean clientes debemos conocer el nombre, una persona de contacto, el cif, una dirección física de la empresa, una dirección de correo y a ser posible un número de fax.

- También se requiere la posibilidad de hacer copias de respaldo tanto del conjunto de archivos y carpetas como de la base de datos del sitio.

- Otro requisito es que podamos acceder a los archivos y carpetas del servidor a través de la interfaz administrativa de Joomla, ya que no tenemos acceso directo al servidor ni acceso mediante protocolos de comunicación típicos como FTP o SSH.

4.1.2 Requisitos del front-end

En la parte pública de nuestro sitio web, es decir, el *front-end*, vamos a tener los siguientes requerimientos:

- Una vez el usuario se haya validado se le debe cargar una página en la que tenga un menú personalizado acorde a lo que tenga contratado. Si tiene un sólo producto, se le carga el único menú que tendrá.

- En caso de tener más de un producto, esta primera pantalla puede ser un menú selector del producto donde aparezcan los logos de cada producto en grande de tal manera que cuando pinche sobre uno se cargue ya la página acorde al producto. La posibilidad de cambiar de un producto a otro en cualquier momento también debe estar contemplada. Para esto podemos usar un *ComboBox* situado en la parte superior.

- Cuando ya está seleccionado el producto debe cargarse un menú en la parte izquierda dando soporte al producto, aquí deben verse los enlaces para los manuales de ayuda, el foro, la zona de descargas y un enlace para volver a la presentación del producto.

- También pondremos en la parte izquierda un pequeño apartado con publicidad en el que mostraremos enlaces a las versiones nuevas de los productos contratados por el cliente, así como enlaces a versiones de prueba de productos que puedan ser de interés. En esta parte debe mostrarse la interfaz de acceso para iniciar o finalizar sesión.

- La zona de descargas de un producto debe ser una lista con las versiones disponibles

y un enlace para su descarga, que nos enviará a un nuevo apartado con la descripción de la versión y el enlace a la descarga del archivo.

4.2 Análisis

Para cumplir con estos requisitos haremos uso de una serie de componentes, módulos y *plugins.* Algunos de ellos vienen con la instalación de Joomla, otros podemos obtenerlos mediante la instalación de extensiones que podemos encontrar en Internet, pero algunos de los requisitos vamos solucionarlos mediante la implementación e instalación de tres módulos y un componente.

Componentes de la instalación

- Para mantener la información acerca de los usuarios y la información de contacto haremos uso de dos componentes nativos de Joomla, estos son: el componente *Usuarios* y el componente *Contactos***.** Estos componentes nos permiten configurar una cuenta de usuario y guardar información que es necesaria conocer de un contacto: nombre, dirección, teléfono, correo electrónico, etc... Además estos componentes están relacionados entre sí y es posible enlazar un contacto con una cuenta de usuario.

- Por lo que respecta a la parte del usuario podemos usar el componente *Banner* del paquete básico de Joomla para crear un tipo de publicidad dinámica, en la que podemos enlazar a la presentación de productos en fase de lanzamiento.

- Para el acceso usaremos el componente *Login* de Joomla que permite al usuario registrarse, acceder o recuperar la contraseña en caso de perderla. Este componente también proporciona un módulo que permite iniciar o cerrar sesión, el cual situaremos en la parte izquierda.

Extensiones obtenidas en Internet

- Para las copias de seguridad buscaremos alguna extensión que sea capaz de hacerlo, puesto que es una tarea muy común en los sitios web, existen extensiones libres capaces de realizar esta tarea. En concreto usaremos un componente llamado *Akeeba Backup* cuyo funcionamiento describiremos más adelante.

- En cuanto al acceso a los ficheros y carpetas existe también una popular extensión llamada *eXtplorer* que nos permite subir y descargar ficheros así como crear, borrar o mover estos ficheros a través de la jerarquía de carpetas del servidor con raíz en el directorio *.../joomla/*. Además este componente nos proporciona un editor para poder modificar los ficheros en línea sin necesidad de volver a subirlos.

- La ayuda la almacenaremos en ficheros HTML por lo que tendremos que buscar alguna forma efectiva de mostrarla, además junto con los ficheros también puede ser interesante mostrar vídeos. Para resolver esto usaremos el componente *Content*, un componente básico que viene con la instalación de Joomla el cual se encarga de gestionar los contenidos en forma de artículos. Pero además haremos uso de un *plugin* llamado *Jumi***,** el cual nos permitirá cargar scripts de código en el interior de los artículos con lo que podremos cargar los ficheros de ayuda.

- Para mostrar los vídeos haremos uso de un *plugin* llamado *AllVideos* que nos permite insertar vídeos en los artículos de forma sencilla.

- La posibilidad de introducir foros en nuestro sitio web la obtendremos instalando un componente llamado *Kunena Forum*, que proporciona una parte administrativa que se maneja desde el *back-end* y que permite crear foros públicos y/o privados, asignar moderadores, crear usuarios independientes para la aplicación, etc...

Extensiones implementadas

- Para mantener la información acerca de los productos, las versiones, las licencias, y los clientes será necesario crear nuevas tablas en la base de datos y establecer una serie de relaciones entre ellas. Para ello vamos a diseñar nuestro propio componente con el objetivo de manipular dichas tablas añadiendo, borrando o editando objetos. Además este componente se encargará de controlar el acceso a los distintos servicios en la parte del usuario.

- El *ComboBox* y el logotipo los mostraremos mediante un módulo que diseñaremos con tal propósito y situaremos en la parte superior izquierda.

- La publicidad estática y el menú los implementaremos también mediante módulos propios.

En este apartado hemos analizado cómo vamos a cumplir con los requisitos de la Intranet, la mayoría de ellos será instalando extensiones que proporcionen la funcionalidad necesaria, pero hay una parte de los requisitos que los vamos a solucionar implementando nuestras propias extensiones. Para ello en primer lugar diseñaremos un modelo relacional[6], que nos servirá para almacenar la información necesaria mediante la creación de nuevas tablas en la base de datos y nos permitirá establecer una relación entre dichas tablas y aquellas tablas de la base de datos original que nos sean útiles.

4.3 El diseño

Para mantener la información vamos a hacer uso de la base de datos de Joomla, a la que vamos a añadir nuevas tablas que podamos manipular a través de la parte administrativa.

A continuación presentamos el modelo relacional que mantiene la información:

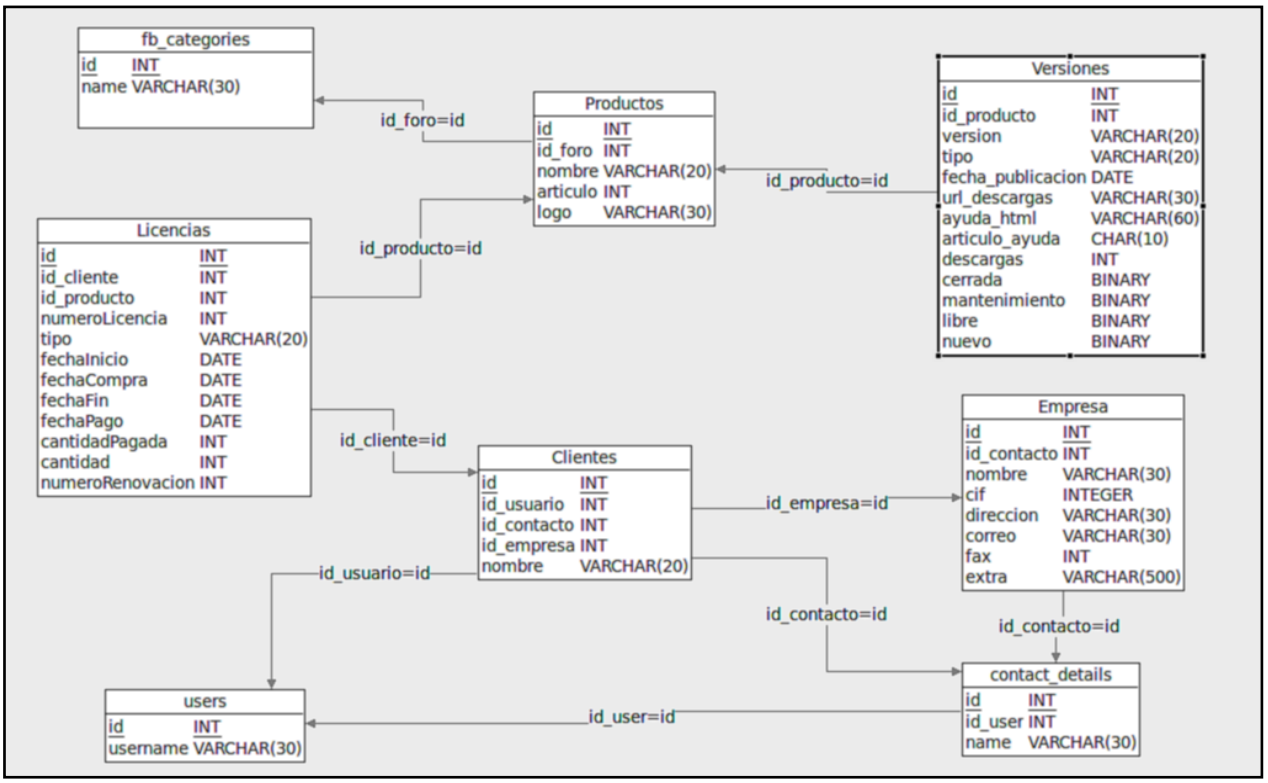

Figura 9. Modelo relacional de las tablas añadidas a la base de datos original

En el modelo se muestran las tablas que hemos añadido para mantener nuestra información y además aparecen las tablas *users* y *contact_details* que son nativas del Joomla y almacenan la información de los componentes *Usuarios* y *Contactos*, las cuales nos han servido para ayudar a organizar las nuestras. En el modelo sólo mostramos de éstas tablas aquellos atributos que nos son relevantes.

En este modelo también se muestra una tabla llamada *fb_categories.* Es una tabla que pertenece al componente *Kunena Forum* y guarda información acerca de los foros. Esta tabla nos será útil a la hora de crear foros privados para los productos.

El resto de tablas las crearemos al instalar nuestro componente principal. Dicho componente es el que se encargará de manipular dichas tablas y recibirá el nombre de SOA.

Con este modelo relacional podremos mantener la información necesaria para el funcionamiento de la Intranet, sin embargo será necesaria una serie de clases y funciones para poder manejar esta información y mostrarla al usuario.

4.4 La implementación

Antes de comenzar con la implementación vamos a comentar con algo más de detalle las funcionalidades que podemos aprovechar del paquete básico de Joomla, las que podemos obtener mediante alguna extensión que ya exista y aquellas que vamos a implementar.

En el *back-end* haremos uso de los siguientes componentes básicos:

- **El gestor de usuarios:** El acceso a la intranet podemos hacerlo a través del método básico de acceso de Joomla, disponemos de un gestor de usuarios que nos permite dar de alta a los usuarios manualmente o bien que estos se registren a través del componente Login. Este componente registra los usuarios, su contraseña y una dirección de correo. También le asigna un grupo a través del cual podemos permitir o restringir el acceso del usuario a ciertas áreas, pero estos atributos son insuficientes para nuestro proyecto.

- **El componente Content:** Este componente es el principal gestor de los contenidos, permite editar artículos que posteriormente se podrán mostrar en el *front-end*. Estos artículos se organizan en categorías y estas en secciones, es obligatorio definir al menos una categoría y una sección a las que asignar los artículos. El editor del paquete básico de Joomla permite insertar imágenes de forma sencilla en los artículos pero no vídeos. También permite editar el artículo en HTML para ganar flexibilidad, pero la funcionalidad HTML no es completa ya que ignora las sentencias que no visualizan texto.

- **El componente Banner:** Este componente nos permite crear “banners” o anuncios. Estos anuncios se configuran escribiendo una dirección en un campo denominado “Haz Clic URL” y después tenemos la opción de escribir un código en HTML o ponerle una imagen al anuncio. Si ponemos una imagen, el código HTML no tendrá efecto, sino, podemos personalizar algo más el anuncio mediante dicho código, incluyendo imágenes, texto, enlaces, etc... Además a cada anuncio se le puede asignar un cliente No se debe confundir el apartado cliente de este componente con los clientes de nuestros productos. En nuestro caso se puede poner como cliente el nombre de un producto y el conjunto de anuncios que tengan dicho producto como cliente serían anuncios para promocionar dicho producto, estableciendo como URL la dirección de la presentación de dicho producto o una versión de promoción del mismo.

- **El componente contactos:** El paquete básico de la instalación también nos proporciona un componente llamado contactos que nos permite almacenar extensa información sobre un cliente y además permite enlazarlo con un usuario, de modo que para cada usuario podremos almacenar la información de contacto. Además podemos organizar los contactos por categorías, cada una de estas categorías podría ser una empresa y una categoría podría usarse para aglutinar los clientes particulares. De esta forma podemos relacionar usuarios, contactos y empresas.Sin embargo no

será realmente necesario establecer esta relación a través de este componente, ya que lo haremos en otro componente diseñado específicamente para ello el cual describiremos más adelante.

Y hasta aquí los componentes proporcionados por la instalación básica que nos son útiles. Como los componentes básicos no nos proporcionan funcionalidad suficiente tendremos que usar algunas extensiones. Para tareas más generales no resulta difícil encontrar extensiones, por lo que nos podemos ahorrar el trabajo de programarlas.

Las extensiones que hemos utilizado para la construcción de nuestra intranet son las siguientes:

- **El componente Akeeba Backup:** Este componente sirve para hacer copias de respaldo del sitio completo. Permite hacer por separado respaldos de la base de datos, de la jerarquía de ficheros o un respaldo completo. Además permite excluir las tablas de la base de datos o los ficheros y directorios que se deseen. Es el sustituto del antiguo *JoomlaPack*, es de distribución gratuita, aunque tiene una versión de pago que incluye algunas características más. Sin embargo la restauración del sitio a partir de los archivos de respaldo no está en la versión gratuita, por lo que implica seguir un proceso un poco más complicado: borrar todos los ficheros de la instalación y a continuación ejecutar desde el navegador un script en PHP que el desarrollador del componente proporciona en un archivo llamado *kickstart.php*. Dicho script debe ejecutarse una vez hemos eliminado la instalación y nos guiará paso a paso para la restauración a partir de nuestra copia de respaldo. Puede obtenerse la última versión del componente desde la página web oficial del proyecto: http://www.akeebabackup.com/download/akeeba-backup/akeeba-backup-3-1.html.

- **El componente eXtplorer:** Este componente nos permite acceder a la jerarquía de ficheros del sitio web a través de la interfaz administrativa para poder subir y/o borrar archivos y directorios del sitio. Esto podría hacerse a través de una conexión FTP o SSH, pero como no disponemos de ella, la forma adecuada para acceder a los archivos es haciendo uso de este componente. Es gratuito e incluye un editor para que podamos abrir y editar los archivos desde la propia aplicación sin tener que descargarlos y subirlos de nuevo. Podemos obtenerlo en: http://extplorer.sourceforge.net/ .

- **El plugin AllVideos:** Este *plugin* nos permite insertar vídeos en los artículos de forma sencilla. Tan solo hay que subir el vídeo a algún directorio del sitio y en el texto del artículo escribir {“ext”}”nombre del fichero”{/“ext”}. Donde debemos sustituir “ext” por la extensión del fichero del vídeo. De este modo podremos incluir vídeos en los artículos y mejorar así la ayuda navegable gracias a los vídeos de ejemplo. Esta extensión es muy popular en los sitios Joomla y es gratuita. Podemos encontrarla en: http://extensions.joomla.org/extensions/multimedia/multimedia-players/video-players-a-gallery/812 .

- **El componente Jumi:** Este componente incluye en su instalación un *plugin* que nos servirá para poder añadir scripts PHP en los artículos. Esto puede ser útil para presentar la ayuda, y para mantener información sobre el producto con el que estamos trabajando en el momento de mostrar el artículo. Para añadir un script PHP o html en un artículo basta con escribir en el texto del artículo {jumi [“ruta relativa del script”][“argumentos”]}. Podemos descargarlo de la siguiente dirección: http://code.google.com/p/jumi/downloads/list

- **Kunena Forum:** Este componente permite la creación y administración de foros. En su administración permite restringir el acceso a un grupo determinado, pero para el desarrollo de nuestra Intranet necesitaremos algo más de flexibilidad, por lo que será necesario hacer algún pequeño cambio en el código de este componente para que permita o restrinja el acceso en función de si el usuario es cliente de un determinado producto. Podemos descargar la última versión en español en la siguiente dirección: http://www.kunenaspanish.com/descargas/cat_view/3-componentes

Finalmente para darle la funcionalidad necesaria al sitio web y mantener la información hemos tenido que implementar en PHP los siguientes módulos y componentes:

- **El módulo para el *ComboBox*:** Este módulo se muestra en la parte del *front-end* y su función es mostrar un *ComboBox* en la parte superior izquierda de la página para poder cambiar de producto en cualquier momento. Lo ideal es usar un módulo para esta tarea ya que desde la parte administrativa podemos configurarlo para mostrar en cualquier sitio de la plantilla y no requiere demasiado código. También hemos aprovechado este mismo módulo para mostrar el logotipo del producto actual de trabajo y los principales parámetros de la licencia.

- **El módulo para el menú:** Este módulo también es para la parte del *front-end* y su función es mostrar un menú con los enlaces a los distintos servicios que ofrecemos para los productos: zona de descargas, ayuda, foros... En el paquete básico de Joomla existe un componente para crear menús, pero no podemos hacer que estos aparezcan o no en función de los productos que tengan contratados los clientes, ya que Joomla solo nos permite restringir el acceso a usuarios registrados o no registrados, por eso se ha tenido que implementar esta parte.

- **El módulo para la publicidad:** Este módulo nos permite hacer publicidad de las nuevas versiones que salgan para los productos que el cliente tiene contratados, estas nuevas versiones pueden ser un nuevo producto que requiera una nueva licencia o simples actualizaciones. Este módulo complementa la función del componente banner mencionado más arriba.

- **El componente SOA:** Este será nuestro componente principal y comprenderá una parte administrativa y otra parte para el *front-end*.

  La función de la parte administrativa es dar de alta los distintos clientes, productos, versiones, empresas o licencias y guardar la información en la base de datos.

La parte del sitio tendrá la función de mostrar los datos guardados en función de los servicios que demande el cliente en cada momento, estos datos se mostrarán en la zona central de la página que es donde la plantilla muestra la salida de los componentes.

Lo primero que debe verse nada más acceder a la Intranet será una presentación donde se mostrarán los logos de los distintos productos contratados por el cliente y los parámetros de la licencia. Los logos serán también un enlace a la presentación del producto en concreto.

También será este componente el encargado de mostrar la ayuda. Cuando se seleccione el enlace correspondiente, el componente buscará la ayuda de la versión del producto y la mostrará en la zona central de la página.

Por último, también tendrá la función de gestionar las descargas de los productos mostrando las distintas versiones de cada producto y los detalles de cada versión una vez se seleccione, para finalmente mostrar un enlace a la descarga del archivo. El componente también se encargará de evitar que usuarios no autorizados tengan acceso a estas descargas.

4.5 Codificación

A continuación pasamos a explicar la estructura del código de nuestro componente principal así como de los módulos. Vamos a profundizar un poco más y vamos a explicar las clases principales que se usan dentro de nuestro componente y su código.

Podemos ver la jerarquía de archivos y carpetas en la siguiente figura.

Figura 10. Componente SOA. Jerarquía de archivos y carpetas en el paquete de instalación.

En la parte superior de la jerarquía tenemos la carpeta *admin*, que contiene la parte administrativa del componente, la carpeta *site*, que contiene la parte del *front-end*, y el archivo *install.xml* que es un script de instalación en lenguaje XML que usa la aplicación de Joomla para instalar extensiones. Tras la ejecución de dicho script lo que vemos dentro de la carpeta *admin* se ubicará en *.../joomla/administrator/components/com_soa/* y lo que vemos en la carpeta *site* se situará en *.../joomla/components/com_soa/*. A partir de este punto nombraremos las carpetas como si ya se hubieran instalado.

El componente principal, que hemos llamado *soa* se compone de dos partes bien diferenciadas de código. En primer lugar explicaremos el código de la parte administrativa o *back-end*.

4.5.1 Back-end

Esta parte sigue el patrón de diseño MVC (Modelo-Vista-Controlador). Este patrón es típico en los componentes desarrollados para Joomla y permite diferenciar la lógica de los datos y de la presentación.

- **El modelo** se encarga de establecer la estructura de los datos y de conectar con la base de datos para obtener los datos o para guardarlos. El modelo contiene los datos que utilizaremos en la aplicación que puede coincidir o no con lo que se almacena en la base de datos. Por ejemplo en la base de datos podemos guardar una tabla con un campo que sea la id de otra tabla (clave ajena) y en el modelo añadir campos adicionales de dicha tabla.
  El modelo hace servir una clase *JTable* que representa de forma fiel el objeto en la base de datos, esta clase se utilizará para guardar cambios en los objetos o crear objetos nuevos.

- **La vista** presenta los datos obtenidos en el modelo, permite seleccionar objetos, que en nuestro caso serán productos, versiones, etc... para editarlos (cambiar sus atributos), guardarlos o borrarlos.
  Para los objetos que manejamos tendremos dos vistas:

  La vista “objetos” que muestra todos los objetos registrados y permite seleccionarlos para borrarlos o editarlos mediante los botones de la barra de tareas que indican estas funciones. También mostraremos en esta vista un botón para crear un objeto nuevo.

  La vista “objeto”. Cuando seleccionamos un objeto para editarlo o creamos uno nuevo, se muestra una vista que presenta los atributos del objeto para modificarlos. En la barra se mostrarán los botones de cancelar o guardar.

- **El controlador** se encarga de la lógica, es decir coordina la vista y la gestión de los datos ejecutando las tareas que tiene registradas. En nuestro caso solo haremos uso de las tareas editar() y borrar() en la vista “objetos” y guardar() y cancelar() en la vista “objeto”. Cuando pulsamos un botón de la barra de herramientas en alguna de las vistas anteriores, se llama al controlador y se ejecuta la tarea indicada por el

botón. El controlador establece la vista que se mostrará tras ejecutar la tarea indicada y permite realizar las funciones necesarias antes de llamar al modelo para obtener o guardar los datos. En la parte administrativa tendremos cinco controladores, uno por cada tabla, y un sexto controlador del que heredan todos. Este controlador no se usa y sirve como plantilla para crear los demás controladores.

Una vez conocido el patrón de diseño podemos pasar a profundizar un poco más en el código. Dentro de la carpeta *admin* podemos ver cinco archivos y cinco carpetas. Ahora explicaremos su contenido uno a uno comenzando por los archivos:

- **controller.php:** Es el controlador principal, tenemos un controlador por cada tabla, que heredan de este controlador. Es el que se ejecutaría si no llamamos a ninguno de los sub-controladores, en nuestro caso no se ejecuta nunca, por lo que podemos considerarlo una clase abstracta.

- **index.html:** Este archivo es un HTML vacío y su función es evitar el acceso a la carpeta desde la web por parte de un usuario. Este archivo debe estar presente en casi todas las carpetas de Joomla.

- **install.sql:** Tras ejecutar el script de instalación se ejecuta este script sql que crea las tablas necesarias para el uso de nuestro componente, también existe un **unistall.sql** que se ejecuta tras la desinstalación y su cometido es borrar dichas tablas.

- **soa.php:** Es el archivo principal del componente, su función es encontrar el controlador que debe usarse y mandarlo a ejecutar la tarea asignada.

```
<?php

// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

// Require the base controller

require_once( JPATH_COMPONENT.DS.'controller.php' );

// Require specific controller if requested
if($controller = JRequest::getWord('controller')) {
        $path = JPATH_COMPONENT.DS.'controllers'.DS.$controller.'.php';
        if (file_exists($path)) {
                require_once $path;
        } else {
                $controller = '';
        }
}
// Create the controller
$classname      = 'SoasController'.$controller;
$controller     = new $classname( );
// Perform the Request task
$controller->execute( JRequest::getVar( 'task' ) );
```

Figura 11. Código fuente del fichero *.../administrator/com_soa/soa.php*

En la Figura 11 podemos ver el código para obtener el nombre del controlador de la variable de entorno *controller*.Busca dentro de la carpeta *controllers* de nuestro componente un archivo con ese nombre, si lo encuentra crea un controlador de ese tipo, sino, crea un controlador del tipo *SoasController*, el cual es el controlador por defecto que está definido en el archivo *controller.php* que hemos comentado antes. A continuación manda al controlador creado ejecutar la tarea que lee de la variable de entorno *task*.

El controlador *producto*

Ahora vamos a adentrarnos en las carpetas del componente empezando por la carpeta *controllers* que contiene los cinco controladores. Estos controladores son muy similares entre sí, a excepción del controlador de la clase producto, que se le ha añadido unas líneas de código para que cuando creamos un producto nuevo se cree una carpeta con el nombre de dicho producto dentro de la carpeta *Productos*, y otra carpeta en su interior de nombre *Ayuda* donde colocaremos los ficheros con la ayuda en HTML.

En la Figura 12 mostramos el código fuente del fichero que se encarga de controlar los objetos *producto.*

El archivo contiene una clase *class SoasControllerProducto* la cual hereda de la case *SoasController*, la cual hereda a su vez de la clase *JController* que forma parte del *framework* de Joomla.

Esta clase trabaja con un objeto *producto* y puede realizar las funciones de eliminar, guardar o editar el objeto. Para editar el objeto se mostrará la vista *producto,* desde donde se podrán editar sus atributos.

En la función *save()* se delegará en el modelo para guardar el objeto. Además se han añadido algunas líneas de código que se encargarán de crear una carpeta con el nombre del *producto* en *.../administrator/com_soa/Productos/* y otra carpeta con el nombre *Ayuda* en su interior si se trata de un objeto *producto* nuevo, o de renombrar la carpeta existente si se ha renombrado un objeto *producto.*

```php
<?php
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
class SoasControllerProducto extends SoasController
{
	function __construct()
	{
		parent::__construct();
		// Register Extra tasks
		$this->registerTask( 'add'  , 'edit' );
	}
	function edit()
	{
		JRequest::setVar( 'view', 'producto' );
		JRequest::setVar( 'layout', 'form'  );
		JRequest::setVar('hidemainmenu', 1);
		parent::display();
	}
	function save()
	{
	  $model = $this->getModel('producto');
	  if ($model->store($post)) {
	//creamos la carpeta propia del producto y sus subcarpetas
		$origen = JPATH_BASE.DS.'components'.DS.'com_soa'.DS.'Productos'.DS;
		$file = 'index.html';
		$basedir=JPATH_BASE.DS.'components'.DS.'com_soa'.DS.'Productos'.DS;
		$dir = $basedir.$model->_data[nombre];
		$dir_viejo = $basedir.$model->_data[nombre_viejo];

		if (!file_exists($dir) ){
			if($dir_viejo!=$dir && $model->_data[nombre_viejo]!=null){

				$yea = rename ($dir_viejo, $dir);
				$creado = 'Producto y carpetas renombrados  '
					.$model->_data[nombre_viejo].' '.$dir;
				}
				else{
					mkdir($dir);
					copy($origen . $file, $dir.DS.$file);
					$dir = $dir.DS.'Ayuda';
					mkdir($dir);
					copy($origen . $file, $dir.DS.$file);
					$creado = 'Creadas carpetas del producto,';
				}
			}
			$msg = JText::_( $creado.' Producto guardado' );
		} else {
			$msg = JText::_( 'Error guardando producto' );
		}

		// Check the table in so it can be edited.... we are done with it anyway
		$link = 'index.php?option=com_soa';
		$this->setRedirect($link, $msg);
	}

	function remove()
	{
	 $model = $this->getModel('producto');
	 if(!$model->delete()) {
	  $msg = JText::_( 'Error: uno o mas productos no han sido borrados' );
	 } else {
	  $msg = JText::_( 'Producto(s) borrado(s), Se han dejado las carpetas' );
	 }
```

Figura 12. Codigo fuente del fichero .../*controllers/producto.php*

Los modelos *producto* y *productos*

A continuación vamos a explicar los modelos:

Existen dos modelos por cada tabla. Los modelos contienen los datos que se representarán en las vistas. Como tenemos dos vistas tendremos dos modelos y estos tendrán el mismo nombre que la vista correspondiente. Uno de los modelos contiene los datos relevantes de todos los objetos de una tabla y el otro modelo contiene todos los datos de un objeto, siendo una fiel representación del objeto en la base de datos.

En concreto estudiaremos los modelos *producto* y *productos* que son muy similares al resto de modelos. Cada uno se encuentra en un fichero PHP dentro de la carpeta *models*. A continuación vemos el código fuente del modelo productos.

```
<?php
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport( 'joomla.application.component.model' );

class SoasModelProductos extends JModel
{
        var $_data;

        function _buildQuery()
        {
           $query = ' SELECT P.*, C.title as presentacion,  F.name as nombre_foro'
                        . ' FROM #__OProductos P'
                        . ' LEFT JOIN #__content C ON P.articulo = C.id   '
                        . ' LEFT JOIN #__fb_categories F  ON  P.foro = F.id'
           ;
           return $query;
        }

        function getData()
        {
                if (empty( $this->_data ))
                {
                        $query = $this->_buildQuery();
                        $this->_data = $this->_getList( $query );
                }
```

Figura 13. Modelo productos. Fichero *.../models/productos.php*

Los modelos son clases que heredan de la clase *JModel* del *framework* de Joomla. Este modelo se encarga de obtener una lista con todos los objetos *producto* de la base de datos. Para ello hace una consulta a la base de datos añadiendo además atributos de las tablas *content* y *fb_categories*. Los objetos de esta lista se mostrarán en la vista *productos*.

En la siguiente figura vemos el código correspondiente al modelo *producto*.

```php
<?php
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport('joomla.application.component.model');

class SoasModelProducto extends JModel
{
	function __construct()
	{
		parent::__construct();
		$array = JRequest::getVar('cid',  0, '', 'array');
		$this->setId((int)$array[0]);
	}
	function setId($id)
	{
		$this->_id		= $id;
		$this->_data	= null;
	}
	function &getData()
	{	// Load the data
		if (empty( $this->_data )) {
			$query = ' SELECT * FROM #__OProductos '.
					'  WHERE id = '.$this->_id;
			$this->_db->setQuery( $query );
			$this->_data = $this->_db->loadObject();
		}
		if (!$this->_data) {
			$this->_data = new stdClass();
			$this->_data->id = 0;
			$this->_data->nombre = null;
			$this->_data->articulo = null;
			$this->_data->logo = null;
			$this->_data->foro = null;
			$this->_data->extra = null;
		}
		return $this->_data;
	}
	function store()
	{
		$row =& $this->getTable();
		$this->_data = JRequest::get( 'post', JREQUEST_ALLOWHTML );
		if (!$row->bind($this->_data)) {
			$this->setError($this->_db->getErrorMsg());
			return false;
		}
		if (!$row->check()) {
			$this->setError($this->_db->getErrorMsg());
			return false;
		}
		if (!$row->store()) {
			$this->setError( $this->_db->getErrorMsg() );
			return false;
		}
		return true;
	}
	function delete()
	{
		$cids = JRequest::getVar( 'cid', array(0), 'post', 'array' );
		$row =& $this->getTable();

		if (count( $cids )) {
			foreach($cids as $cid) {
				if (!$row->delete( $cid )) {
					$this->setError( $this->_db->getErrorMsg() );
					return false;
```

Figura 14. Modelo producto. Fichero *.../models/producto.php*

Este modelo trabaja con un objeto *producto* almacenado en la base de datos y contiene los datos que veremos en la vista *producto* cuando queramos editar o crear un nuevo objeto.

Cuando desde la vista *productos* se muestra una lista de los productos almacenados en la base de datos, podemos crear uno nuevo o seleccionar uno de la lista para editarlo o varios objetos para borrarlos. Estas tareas las lleva a cabo el controlador pero delegando la tarea al modelo *producto.*

Para editar un objeto, el controlador llamará a la vista *producto*, la cual mostrará los atributos del objeto en campos editables. Para ello obtendrá los datos del objeto a través del modelo pasándole su *id.* El propio modelo se encargará de hacer una consulta a la base de datos y obtener el objeto íntegro a través de su *id* haciendo uso de la función *&getData().* Si en lugar de editar tratamos de crear un objeto nuevo el proceso es el mismo, pero el *id* del objeto sería 0, y los campos editables se presentarían con un valor por defecto.

Para entender bien como funciona el modelo *producto* es conveniente conocer la clase *TableProducto.* Esta clase hereda de la clase *JTable* del *framework* de Joomla y su función es ser una clase persistente a través de la cual guardaremos y borraremos los objetos de la base de datos.

Una vez editado los campos, si se pulsa el botón guardar la ejecución vuelve al controlador, el cual invocará la función *store().* Esta función obtiene los campos editados en la vista *producto* en forma de variables, después obtiene una instancia de la clase *TableProducto.* La tarea de almacenar el objeto en la base de datos se realiza a través del objeto *TableProducto* el cual se encarga de la conexión con la base de datos y de ejecutar las sentencias correspondientes en SQL.

Desde la vista *productos* también es posible seleccionar uno o varios objetos para borrarlos. Cuando se pulsa el botón borrar, la ejecución vuelve de nuevo al controlador, el cual delega la tarea al modelo invocando la función *delete().* Esta función obtendrá una lista con el *id* de cada objeto seleccionado y una instancia del objeto *TableProducto* al que delegará la tarea final de borrar el objeto. Por último mediante un bucle *for* llamará repetidamente a la función *delete($cid)* de la clase *TableProducto* donde la variable *$cid* contiene el *id* del producto que se debe borrar.

La vista *producto* y *productos*

Ahora vamos a explicar la parte de *la vista* que es la más extensa y complicada del componente. De cada tabla de nuestro componente tendremos dos vistas, igual que con los modelos. Una de las vistas es la encargada de mostrar en forma de tabla una lista con los objetos almacenados, añadiendo si es preciso algún atributo a los objetos. La otra vista muestra los atributos de un objeto determinado en forma de campos editables, pudiendo crear nuevos objetos o editar objetos existentes.

Cada vista se compone de dos ficheros, en uno obtenemos los datos que queremos mostrar haciendo uso del modelo para esta tarea. El segundo fichero es una plantilla que se

encarga de organizar la presentación del los datos previamente obtenidos.

En la siguiente imagen puede verse la jerarquía de archivos y carpetas que componen la vista.

Figura 15. Jerarquía de carpetas y archivos de la vista

A continuación vamos a analizar las dos vistas correspondientes a los productos. Comenzamos explorando la carpeta *.../com_soa/view/productos/* que contiene la vista *productos*. En esta vista mostramos los atributos relevantes de los objetos *producto* en forma de tabla con algunos atributos añadidos.

El fichero que se encarga de obtener los datos tiene de nombre *view.html.php* y se sitúa en la carpeta *.../views/productos/*. En la siguiente imagen podemos ver su código.

```php
<?php
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );

jimport( 'joomla.application.component.view' );

class SoasViewProductos extends JView
{

	function display($tpl = null)
	{
		JToolBarHelper::title(   JText::_( 'Intranet SOA: Productos' ), 'generic.png' );
		JToolBarHelper::deleteList();
		JToolBarHelper::editListX();
		JToolBarHelper::addNewX();

		// Get data from the model
		$items		= & $this->get( 'Data');
		$this->assignRef('items',		$items);
		parent::display($tpl);
```

Figura 16. Vista *productos*. Fichero *.../views/productos/view.html.php*.

La vista es en realidad una clase heredada de la clase *JView* de Joomla. En esta clase reescribimos la función *display()*. Aquí se llama a la clase *JToolBarHelper* de Joomla para indicar los botones que deben verse. Después se obtiene la lista de objetos que deben mostrarse con la ayuda del modelo *productos*. Esta lista es transformada en una variable de clase de nombre *items* que será accesible desde el otro fichero, el encargado de organizar la presentación de los datos. Por último se llama a la función *display()* de la clase heredada que se encargará de mostrar los datos haciendo uso del código del otro fichero de la vista.

El segundo fichero se encuentra en la carpeta *.../views/productos/tmpl/* y lleva por nombre *default.php*. Este fichero está escrito en su mayor parte en HTML, aunque hace uso de sentencias en PHP para acceder a los datos que se han obtenido mediante el código del fichero anterior. En realidad no es más que una plantilla para presentar los datos obtenidos (más detalles se pueden ver en el Anexo1: Fichero *.../views/productos/tmpl/default.php*. Página 60)

Hasta aquí la primera parte de la vista, la vista "productos", que mostraba la lista de objetos contenidos en la base de datos con sus atributos añadidos de otras tablas. Ahora vamos a explicar la vista *producto*, que es la vista que vemos al modificar o crear un producto. Esta vista se encuentra en *.../views/producto/* y al igual que la anterior tenemos una clase *JView* en un archivo llamado *view.html.php* y una plantilla.

Analizamos ahora el fichero *.../views/producto/view.html.php* que es muy similar al que vimos en la carpeta */views/productos/*. Podemos ver su código en la siguiente figura.

```php
<?php
// No direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport( 'joomla.application.component.view' );

class SoasViewProducto extends JView
{
	function display($tpl = null)
	{
		$producto		=& $this->get('Data');
		$isNew			= ($soa->id < 1);

		$text = $isNew ? JText::_( 'New' ) : JText::_( 'Edit' );
		JToolBarHelper::title(   JText::_( 'Productos' ).': <small><small>[ ' . $text.' ]</small></small>'
);
		JToolBarHelper::save();
		if ($isNew)  {
			JToolBarHelper::cancel();
		} else {
			// for existing items the button is renamed `close`
			JToolBarHelper::cancel( 'cancel', 'Close' );
		}

		$db = & JFactory::getDBO();

		$query ="SELECT id , title FROM #__content ";
		$db->setQuery( $query );
		$larticulos = $db->loadObjectList();

		$query ="SELECT id , name FROM #__fb_categories ";
		$db->setQuery( $query );
		$lforos = $db->loadObjectList();

		$this->assignRef('producto',$producto);
		$this->assignRef('larticulos',$larticulos);
```

Figura 17. Vista *producto*. Fichero *.../views/producto/view.html.php*

El fichero es muy parecido al mismo archivo de la vista *productos* con la principal diferencia de que hemos añadido unas lineas de código que conectan con la base de datos. La función de estas lineas es obtener los atributos *id* y *nombe* de las tablas donde se encuentran los artículos y los foros y las hemos asignado a variables de la vista para tener acceso en la plantilla.

Veamos ahora la plantilla. La plantilla para esta vista tiene de nombre *form.php* y se encuentra en la carpeta *.../views/producto/tmpl*

Esta plantilla tiene algunas similitudes con la anterior en la idea de un formulario con una tabla para presentar los atributos, pero en este caso solo vemos los atributos de un objeto y los campos son editables, pues el objetivo en esta vista es crear o modificar un objeto.

Vamos a hacer una breve descripción de los campos de edición de este formulario, aunque cada formulario puede tener tipos de campos distintos, en esencia son iguales. La parte izquierda de cada campo será una etiqueta con la descripción del campo, la parte derecha será el campo en sí identificado por el nombre del atributo del objeto.

- **nombre**: Este campo es un área de texto donde se edita el nombre del producto.

- **artículo:** Este campo es un *ComboBox* que muestra el título de los artículos de la tabla *content* y toma como valor el *id* del artículo mostrado. En este campo se elige el artículo con la descripción del producto y se usa como presentación. Se utiliza la sentencia *select* de HTML y se obtienen los datos mediante un bucle *for* en PHP que lee la variable *$this->lartículos* que cargamos previamente en clase *JView.*

- **Logo:** Para seleccionar el logo utilizaremos otro *ComboBox* como el anterior pero para obtener los logos los buscaremos en la carpeta *.../com_soa/Productos/logos.*

- **Foro:** Este campo es idéntico al de selección de artículos, lo único que varía es la variable de la que obtenemos los datos. El foro que seleccionemos aquí será el foro privado del producto y los sub-foros que creemos en su interior también serán privados, lo que hace que solo los usuarios con una licencia para el producto tienen acceso.

- **Extra:** En este campo podemos poner una descripción del producto. Para facilitar la edición y poder mostrarla en una página hemos hecho uso del editor por defecto de Joomla.

Cuando pulsemos sobre el botón guardar o cancelar enviaremos el formulario con los datos a la página *index.php***,** que se encargará de llamar al archivo principal del componente *.../com_soa/soa.php*, el cual llama al controlador correspondiente, en este caso *producto*, y ejecuta la tarea indicada. Si la tarea es guardar, el controlador obtiene el identificador del producto de la variable *id* del formulario (0 en caso de producto nuevo) y también obtiene el nombre anterior del producto (cadena vacía en caso de producto nuevo). Si el nombre del producto no coincide con el anterior, o bien se trata de un producto nuevo o hemos renombrado el producto, el controlador se encargará de crear la carpeta correspondiente o renombrarla. Después delegará en el modelo para guardar el objeto *(*más detalles se pueden ver en el Anexo 2: Fichero *.../views/producto/tmpl/form.php.* Página 61)

Con esto quedan explicadas todas las vistas ya que son casi idénticas pero cambiando los datos que se muestran. Sin embargo existe una vista adicional. Para poder ver la lista de licencias que tiene contratadas un cliente se ha añadido a la vista *cliente* una plantilla adicional que obtiene sus licencias de la base de datos y las muestra en una tabla.

4.5.2 Front-end

Vamos a explicar a continuación la parte del usuario empezando con el componente principal, más adelante explicaremos los módulos.

La parte del sitio del componente no sigue el mismo patrón de diseño que la parte administrativa. En esta parte hemos usado un controlador y un *helper*. El *helper* hará tanto de modelo como de vista y su función es proveer de funciones de soporte para obtener los datos por parte del controlador, así como para presentarlos en la página. Este *helper* es usado en los módulos para obtener los datos necesarios para su funcionamiento.

La parte del sitio del componente principal consta tan solo de tres ficheros que se sitúan en la carpeta *.../joomla/components/com_soa/***.**

En primer lugar explicaremos brevemente los ficheros del componente y luego profundizaremos en el código.

- **soa.php:** Tiene la misma función que el fichero con su mismo nombre de la parte administrativa. Contiene el código que se ejecuta cuando se llama al componente y su función es obtener el controlador a usar y mandarlo a ejecutar la tarea indicada.

- **controller.php**: Como en la parte administrativa es el controlador principal del componente y es el que se ejecuta por defecto. En este caso si le damos uso, puesto que solo tendremos un controlador.

- **helper.php**: Como hemos explicado antes, este fichero hace de soporte para obtener los datos y proporciona también funciones de ayuda que usaremos para aligerar las tareas del controlador. El helper tiene la función, como su nombre indica, de ayudar a los componentes y módulos que lo requieran aportando las estructuras de datos y clases que se requieran, así comofunciones que devuelvan código en html para presentar estos datos.

Ahora pasamos a comentar los códigos desarrollados. Como el fichero inicial *soa.php* es casi idéntico al fichero con el mismo nombre de la parte administrativa, comenzaremos directamente viendo el controlador.

Cada uno de los métodos de esta clase son tareas del controlador y pueden ser requeridas por alguno de los módulos de la parte del sitio para mostrar datos en la parte principal de la página.

En las primeras lineas podemos encontrar la función require_once(JApplicationHelper::getPath('helper' , 'com_soa')). Esta orden llama al *helper* para poder usar sus funciones como si estuvieran escritas en este mismo fichero. Esta orden la encontraremos en todos los módulos ya que el *helper* tiene las funciones que se encargan de conectar con la base de datos.

Este fichero es el más importante de la parte pública del componente y se encarga de las tareas que proporcionan los servicios a los clientes, por tanto haremos un análisis algo más profundo sobre las funciones que comprenden el controlador.

- function cambio(): Esta función se llama desde el módulo *logo_soa*, el cual se encarga, además de mostrar el logotipo, de mostrar el *ComboBox* de selección de producto. El módulo le pasa por el método POST el artículo de presentación del producto seleccionado, si no se selecciona ningún producto se redirige a la página de presentación.

- function redirigir($pagina): Esta función toma como argumento una cadena que indica la dirección a la que redirigir. La función -header()- de php se encarga de la redirección.

- function ayuda(): Esta función se llama desde el módulo menu_soa. La función obtiene de la barra de direcciones el id del producto y obtiene el objeto producto con esa id gracias a la definición de la clase producto incluida desde el helper. Esta clase tiene un método que permite obtener la lista de versiones del producto una vez obtenida se muestra un enlace a la ayuda en html de cada versión disponible.

- function mostrar_ayuda(): Una vez seleccionada la versión de la que se quiere ver la ayuda, esta función obtiene del objeto versión el fichero de ayuda y lo carga en un iframe. Un iframe es un marco donde se carga el fichero como si de una nueva página se tratase, pero incluida dentro la parte reservada para nuestro componente.

- function descargar(): A la hora de descargar una versión de un producto se hace en 3 fases. Esta función realiza la primera fase, que muestra una tabla con las versiones del producto seleccionado. Se llama desde el menú en el enlace “descargas”. La

función primero comprueba que el usuario tiene licencia para este producto, ya que aunque el enlace no apareciera en el menú, se puede escribir en la barra de direcciones la dirección que lleva aquí. Si el usuario tiene la licencia se obtiene la lista de versiones del producto y con la función --getTabla_descargas($producto->getListaVersiones())-- obtenemos una tabla con los parámetros de cada versión y un enlace a su descarga. En caso de que el usuario no tenga licencia para este producto, la función redirige a la página de presentación.

- function descargar_version(): Una vez seleccionada la versión que queremos descargar entramos en la segunda fase. De nuevo comprobamos que el usuario tiene la licencia correspondiente del producto por el mismo motivo que antes. Si tiene la licencia se llama a la función descarga_producto($version,$producto); definida en el helper, que tomado un objeto producto y un objeto versión muestra el logo del producto, el nombre de la versión y el número de descargas realizadas. A continuación se muestra la descripción de la versión, si se ha guardado una.

- function contar(): Una vez pulsamos el enlace de descarga se llama a esta función, el último paso para descargar la versión. Primero se hace la comprobación pertinente y después se llama a la función descargado($version); que se encarga de incrementar el contador de descargas de dicha versión. A continuación se obtiene la url de descarga que es un atributo del objeto versión y se redirige a ella, lo que muestra en pantalla la opción de descargar del navegador. A continuación se redirige a la dirección en la que nos encontrábamos para dar la apariencia de que la página no ha cambiado tras la descarga.

- function actualizar_tablas(): Esta función es una función de mantenimiento. Solo puede llamarse desde la barra de direcciones y está pensada para realizar cambios en la base de datos de Joomla. La función comprueba que el usuario que la llama es un Super Administrador, (gid = 25) y ejecuta la sentencia sql que se escriba en el código.

- function presentacion(): Esta función es la primera que se llama cuando un usuario accede al sitio web. Con la ayuda de la clase listasSoa definida en el helper, la función obtiene una lista con los productos del usuario y otra con las licencias. Después las presenta en forma de tabla mostrando los parámetros de la licencia más relevantes haciendo uso de la función presentar_licencia($licencia,$producto); definida en el helper, que toma un producto y una licencia y devuelve la tabla correspondiente en html *(*más detalles se pueden ver en el anexo 3*: .../components/com_soa/controller.php.* Página 63)

  A continuación pasamos a explicar el helper:

El siguiente fichero importante, es *helper.php.* Este fichero es el más extenso de todos. En él están definidas las clases que dan soporte a la estructura de datos utilizada y las funciones que ayudan a la presentación de los datos, dejando el resto de ficheros que se encarguen tan solo de la lógica para enlazar los datos y la presentación.

Tenemos definidas tres clases en este fichero: listasSoa, Producto y Version, cada una con sus métodos para instanciarla y para obtener algunos datos más.

- **listasSoa:** Esta clase se encarga de obtener las listas de los productos de un cliente, las versiones de esos productos y las licencias.

  Las principales funciones de la clase son:

- static function getInstance(): Obtiene una instancia de la clase para poder usar sus métodos y que las listas se puedan cargar en las variables de la clase.

- function getVerDis(): Obtiene una lista con las versiones de los productos de los que el usuario tiene licencia e incluye aquellas versiones que son de libre distribución. Para ello hace uso de la función function queryVersiones() que contruye la consulta Sql para obtener estos datos. Una vez tiene la lista de versiones, mediante un bucle for crea un objeto Version, con el objetivo de que ese objeto tenga acceso a los métodos de la clase.

- function getProductos(): Obtiene una lista con los productos de los que el usuario tiene licencia. Se ayuda de la función function queryProductos() para contruir la consulta Sql. Del mismo modo que la función anterior también obtiene una lista con objetos Producto.

- function getLicencias(): Obtiene una lista con las licencias del usuario, al ser la consulta algo más simple, no se hace uso de una función para contruirla. No se construye una lista de objetos Licencia porque esta clase no era necesaria, ya que no se necesitaban métodos para la clase.

- **Producto:** Esta clase además de almacenar los atributos de un producto, proporciona métodos para obtener las versiones de este producto, la licencia del usuario para el producto y un método para mostrar el logo.

  Veamos sus funciones:

- function Producto($id, $data): Construye un objeto Producto bien a partir de su id, accediendo a la base de datos, o bien tomando como parámetro los atributos del producto y guardándolos en la variable $datos.

- function getLicencia(): Mediante una consulta a la base de datos, obtiene la licencia que el usuario tiene para ese producto.

- function getListaVersiones(): Mediante otra consulta a la base de datos, obtiene las versiones del producto y construye una lista con objetos Version.
- function show_logo(): A partir del id del artículo de presentación del producto, construye un enlace a este artículo con la imagen del logo.

- **Version:** Esta clase es similar a la clase anterior. Se utiliza para construir un objeto *version* a partir de su *id* o teniendo ya sus datos. Además proporciona un método para obtener la url del fichero descargable de la versión.

- function Version($id, $data): Es el constructor de la clase. Si la variable $data no está vacía construye un objeto Versión con los atributos de esa variable, en caso contrario utiliza la variable $id para obtener el objeto de la base de datos.

- function enlace(): Esta función devuelve la ruta completa del fichero de descarga de la versión .

El resto de funciones que hay en el *helper* son funciones de apoyo para el controlador, con la excepción de una función que restringe el acceso a los foros a los usuarios no permitidos.

- function Comprobar_acceso_foro($idforo): Esta función toma como parámetro el id de un foro del que se desea determinar si el usuario debe tener acceso o no. Si el foro es privado, solo estará permitido el acceso a los usuarios que tengan licencia de uso del producto que se trata en el foro. Los foros descendientes de este también serán privados. La función hace una consulta a la base de datos para comprobar si el foro es privado y si lo es, comprueba si el usuario tiene acceso. Para poder hacer esta comprobación ha sido necesario modificar un fichero del componente *kunena_forum.* En concreto en el fichero *.../joomla/components/com_kunena/Class.kunena.php* se ha añadido tras la linea 944:
  --require_once(JApplicationHelper::getPath('helper' , 'com_soa'));--
  y la linea originalmente 950 se ha sustituido por:
  -- ($row->pub_access == -1 and comprobar_acceso_foro($row->id) == true) or --
  Es importante tener en cuenta que este fichero y estas lineas corresponden a la versión 1.5.12, ya que podrían cambiar en versiones diferentes (más detalles se pueden ver en el Anexo 4: *.../components/com_soa/helper.php.* Página 65)

4.5.3 Módulos

Los módulos son extensiones cuya finalidad es mostrar datos en las zonas periféricas de la página. Para el desarrollo de la Intranet hemos implementado tres módulos, todos para el *front-end.* Se ubican en carpetas dentro de *.../joomla/modules/* y ahora procedemos a su explicación.

- **mod_logo_soa:** Aquí hemos programado un *ComboBox* para cambiar el producto actual de trabajo y también mostramos el logotipo y lo principales parámetros de la licencia. Consta de un solo fichero llamado *mod_logo_soa.php* ubicado en .../joomla/modules/mod_logo_soa.

A continuación mostramos su código.

```
<?php

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

 require_once(JApplicationHelper::getPath( 'helper' , 'com_soa' ));

 $lista_p = listasSoa::getInstance();
 $lproductos = $lista_p->getProductos();

 $prodId = JRequest::getVar('prod');
 if($prodId == null)
       $prodId = JRequest::getVar('producto');



  echo '<TABLE CELLPADDING=5 CELLSPACING=5  >';
  echo '<div style="padding-left: 1px" >';
  echo '<tr>';
  echo '<td  ><form action="index.php?option=com_soa&task=cambio" method="POST">
        <select name="cambiar" onchange="this.form.submit()">';
  echo  '<option value="0" selected>'.productos.'</option>';
        foreach($lproductos as $producto){
           if($prodId == $producto->datos->id){
             echo  '<option value="'.$producto->datos->articulo.'" selected>'.$producto->datos-
>nombre.'</option>';
           }
           else echo  '<option value="'.$producto->datos->articulo.'">'.$producto->datos->nombre.'</option>';
        }
  echo '</select></td>
       </form>';
  echo '</tr>';


   if($prodId!=null){
       $producto = new Producto($prodId,'');
       $licencia = $producto->getLicencia();

       echo '<tr>';
      echo '<td>';
       $producto->show_logo();
       echo '</div>';
       echo '<div style="padding-left: 25px">';
       echo '<td>';
                echo '<BR>'.$producto->datos->nombre.'<BR>';
                echo 'Licencia: '.$licencia->Tipo.'<BR>';
                echo 'Expira: ';
                if($licencia->FechaFin != '0000-00-00')
                        echo $licencia->FechaFin.'<BR><BR>';
                else echo 'Nunca<BR><BR>';
       echo '</td>';
       echo '</tr>';
       echo '</div>';
```

Figura 18. Módulo logo_soa. Fichero *.../modules/mod_logo_soa*

El módulo obtiene, con la ayuda del *helper* del componente *com_soa*, una lista con los productos del cliente, y leyendo la variables de entorno *producto* obtiene el producto actual, si hay uno. Después crea una tabla en html con tres celdas. En la

parte superior muestra el *ComboBox* con la lista de productos y si hay un producto actual de trabajo, en la parte inferior muestra el logotipo y la licencia del producto, el *ComboBox* dejará este producto seleccionado.

- **mod_menu_soa:** Aquí mostramos un menú con enlaces a la presentación de cada producto contratado, al foro, a la zona de descarga y a la página de ayuda. Consta de un solo fichero llamado *mod_menu_soa.php* ubicado en *.../joomla/modules/mod_menu_soa*.

A continuación mostramos su código.

```
<?php

   defined( '_JEXEC' ) or die( 'Restricted access' );
   require_once(JApplicationHelper::getPath( 'helper' , 'com_soa' ));

       $lista_prod = listasSoa::getInstance();
       $lproductos = $lista_prod->getProductos();

     if ($lproductos!=null){
          foreach ($lproductos as $producto) {
                echo '<div style="padding-left: 5px"><BR>';
               echo '<a href="index.php?option=com_content&view=article&id='. $producto->datos->articulo
                      .'&prod='.$producto->datos->id.'"><span>'. $producto->datos->nombre .'</span></a><br/>';
               echo '</div>';
               echo '<div style="padding-left: 16px">';
               echo '<a href="index.php?option=com_soa&task=ayuda&prod='.$producto->datos->id
                      .'"><span>Ayuda</span></a><br/>';
               echo '<a href="index.php?option=com_kunena&Itemid=0&func=showcat&catid='.$producto->datos->foro
                     .'&prod='.$producto->datos->id.'"><span>'.Foro.'</span></a><br/>';
               echo '<a href="index.php?option=com_soa&task=descargar&prod='.$producto->datos->id
                      .'"><span>'.Descargas.'</span></a><br/>';
               echo '</div>';
              }
          echo '<BR>';
      }
      else{
         echo 'No tiene Productos contratados<BR>';
      }
```

Figura 19. Módulo menu_soa. Fichero *.../modules/mod_menu_soa.*

Con la ayuda del *helper,* obtiene una lista con los productos contratados, después mediante un bucle *for* para cada producto se muestra un enlace a la presentación del producto, al foro, a la zona de descarga y a la página de ayuda. En caso de no tener ningún producto con licencia vigente se muestra un texto indicando que no hay productos contratados.

- **mod_publi_soa:** Este módulo muestra una publicidad de forma estática, se muestran las versiones nuevas de los productos contratados y las versiones de libre distribución de otros productos. Consta de un solo fichero llamado *mod_publi_soa.php* ubicado

en *.../joomla/modules/mod_publi_soa*.

```php
<?php

// no direct access
defined( '_JEXEC' ) or die( 'Restricted access' );
#
#  // require the helper
require_once( JApplicationHelper::getPath( 'helper', 'com_soa' ) );
#
    $listas_publi = listasSoa::getInstance();

    $listaVD = $listas_publi->getVerDis();


    echo '<div style="padding-left: 5px" >';
     echo '<TABLE CELLPADDING=2 CELLSPACING=7 >';
    foreach($listaVD as $version){
       echo '<tr><td>';
                echo '<a href="index.php?option=com_soa&task=descargar_version&ver='.$version->id.'"><span>'
                . $version->nombre_producto.' '
                .$version->version.'</span></a></td><td width="50" align="center">';
                if($version->nuevo == Si)
                          echo '<IMG SRC="images/new.png" align="center"></td>';
                if($version->libre == 'Si')
                 echo '<td><IMG SRC="images/gratis3.png" align="center"></td></tr>';


     }
     echo '</TABLE>';
     echo '</div>';
```

Figura 20.Módulo publi_soa.Fichero *.../modules/mod_publi_soa/mod_publi_soa.php*

Obtiene una lista con las versiones cuyo atributo *nuevo* es *'Si'* de los productos contratados y de versiones libres de otros productos con la ayuda del helper. Después muestra un enlace a la descarga de la versión.

Para instalar el componente y los módulos hemos creado un paquete de instalación para cada extensión y hay que instalarlos por separado a través de la administración de Joomla. Los paquetes contienen un archivo *install.xml* que usa Joomla para instalar cada extensión.

5 Casos de uso

Este apartado pretende ser un pequeño tutorial donde explicamos cómo crear un producto y dar de alta un cliente para poder ofrecerle servicio paso a paso, incluyendo la creación de un foro, versiones del producto, etc...

Cuando queremos dar de alta un producto en nuestra base de datos, antes que nada debemos crear un foro para el producto y un artículo para su presentación, también es conveniente subir el logotipo del producto antes de su creación. No es estrictamente necesario seguir este orden, podríamos crear un producto y dejar sus atributos vacíos para editarlos posteriormente, aunque el método que vamos a explicar pretende crear todo lo necesario en el mínimo número de pasos posibles.

5.1 Dar de alta un producto

Aquí explicamos cómo crear un producto el cual llamaremos "Routing Maps" para darle un toque de realismo, ya que es uno de los productos ofertados por la empresa.

1. **Crear un artículo de presentación:**
   Vamos al gestor de artículos y creamos un artículo nuevo. Previamente es necesario haber creado una sección y una categoría dentro de ésta. El artículo podemos llamarlo *Presentación Routing Maps*. Debemos crearlo con acceso *Registrado* y debemos marcarlo como publicado.

2. **Crear un foro:**
   En segundo lugar será necesario un foro. Para esta tarea debemos ir al componente *Kunena Forum* y en el administrador de foros crear un foro que podemos llamar *Zona Routing Maps*. Dentro del foro creamos dos foros más uno será un foro público y otro será privado. Podemos llamarlos "Foro público RM" y "Foro privado RM" respectivamente. Todos los foros deben crearse con nivel de acceso "Todos registrados".

3. **Subir el logo:**
   Antes de crear el producto aún nos queda un paso, que es subir el logotipo al servidor. Para ello podemos hacer uso del componente *eXtplorer* y debemos subir el logo a la carpeta *.../joomla/administrator/components/com_soa/Productos/logos/***.** Esta carpeta se crea cuando se instala el componente com_soa.

4. **Crear el producto:**
   Por último vamos al componente *Intranet SOA* y en la sección *productos* creamos un producto nuevo poniendo como nombre *Routing Maps*, artículo de presentación *Presentación Routing Maps*, foro *Foro privado RM* y logo el que subimos previamente. En el campo Descripción del producto podemos añadir un texto con formato para describirlo.

En la siguiente imagen se muestra la interfaz para dar de alta un nuevo producto.

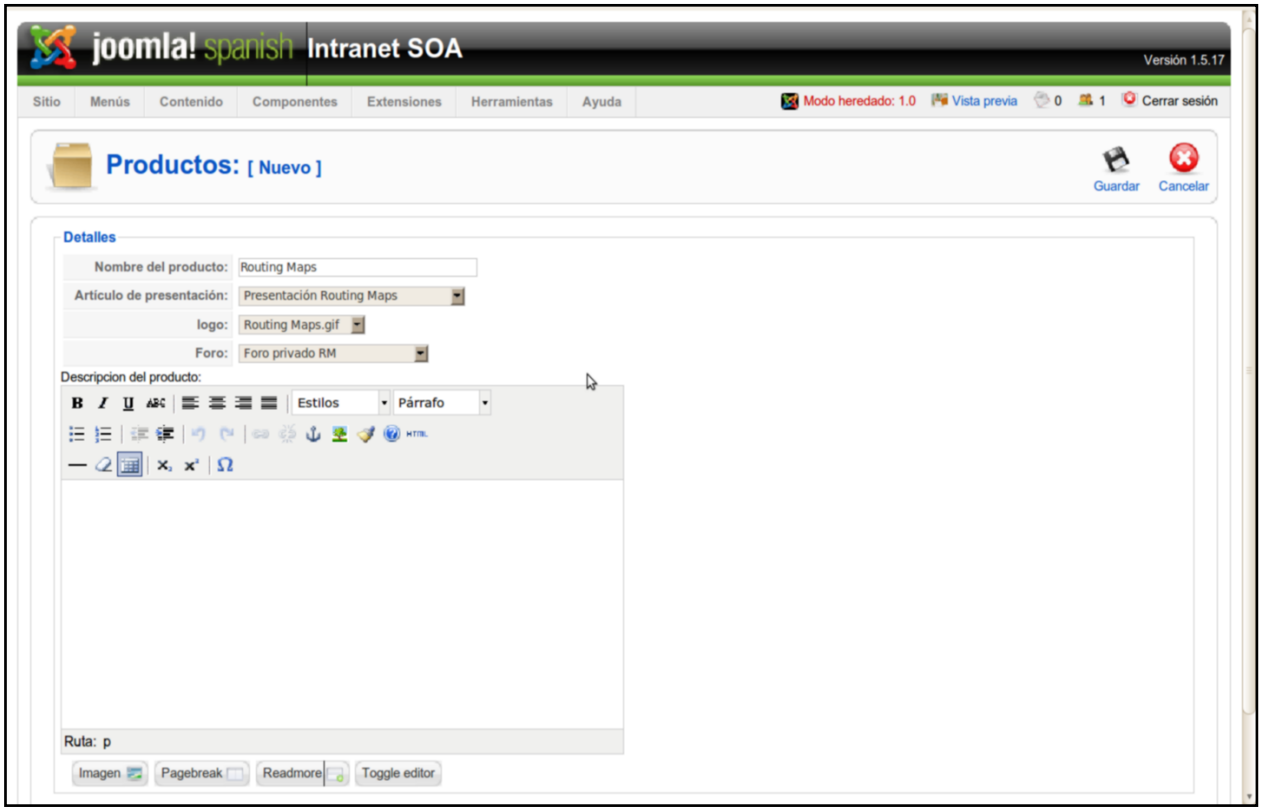

Figura 21. Interfaz alta producto.

5.2 Dar de alta una versión

Ahora vamos a dar de alta la versión 1.0 del producto. Antes de crear el objeto versión será necesario dar algunos pasos previos, asumimos que hemos creado ya un producto.

1. **Subir fichero descarga**

Lo primero que tendremos que hacer es subir el fichero de descarga de la versión. Para ello usaremos como antes el *eXtplorer* y lo subiremos a la carpeta *.../joomla/administrator/components/com_soa/Productos/Routing Maps/*. Esta carpeta se crea automáticamente cuando creamos el producto y si renombramos el producto también se renombrará automáticamente.

2. **Subir fichero ayuda**

Igual que el primer paso, pero la carpeta destino será *.../joomla/administrator/components/com_soa/Productos/Routing Maps/Ayuda/*. Esta carpeta también se ha creado al crear el producto.

3. Crear el artículo ayuda

Además de un fichero de ayuda vamos a necesitar un artículo, ya que será necesario si queremos añadir algún vídeo para complementar la ayuda. Para añadir un vídeo debemos poner {“ext”}”nombre vídeo”{/”ext”} donde “ext” es la extensión del fichero que contiene el vídeo. También será necesario añadir una linea referente al *plugin jumi* para que en el artículo se incluya un script en PHP que le indiquemos. En concreto le vamos indicar que ejecute el scipt ubicado en *.../joomla/components/com_soa/jumi_soa.php* y para ello habrá que escribir una linea que ponga {jumi [components/com_soa/jumi_soa.php]}.

4. Crear el objeto version

Con esto ya podemos ir a la sección *versiones* del componente *Intranet SOA* y crear nuestra versión. El resto de parámetros se dejan a opción del administrador.

En la siguiente figura se muestra la interfaz para dar de alta una versión en la base de datos.

Figura 22. Interfaz alta versión.

5.3 Dar de alta un cliente

Para dar de alta un cliente, es necesario haber dado previamente de alta un usuario. El cliente puede relacionarse con un usuario, una empresa y un contacto, sin embargo el contacto y la empresa no son estrictamente necesarios, puesto que guardan una información que no urge a la hora de dar servicio, puesto que la cuenta de usuario ya guarda una cuenta de correo que sería lo mínimo necesario para poder comunicarnos con el cliente. La información de contacto está bien tenerla como información adicional, pero no es imprescindible y en cuanto a la información acerca de la empresa tampoco es imprescindible ya que un cliente podría ser un particular y ni siquiera tendría una empresa asociada.

1. **Dar de alta al cliente:**

Una vez hayamos dado de alta al usuario podemos pasar directamente a dar de alta al cliente, solo tenemos que asociar un nombre para identificar el objeto cliente y podemos dejar la información de contacto y de empresa vacía por el momento.

5.4 Dar de de alta una licencia

Este es el último paso necesario para poder ofrecer servicio al cliente. Tenemos que crear una licencia que relacione al cliente previamente dado de alta y el producto que ha contratado.

1. **Dar de alta una licencia**

Aquí solo es necesario especificar un número de licencia que identificará el contrato, seleccionar el cliente y el producto que relaciona el contrato. El resto de campos sirven para mantener información de interés pero no son esenciales para el funcionamiento de la página. Una vez dado de alta el contrato podemos comprobarlo yendo a la sección *cliente* y pinchando en *ver* en la columna *licencias***.**

6 Conclusiones

Para desarrollar toda esta tarea en primer lugar ha sido necesaria un largo tiempo dedicado a documentación acerca de los CMS, y más concreto de Joomla. Primero fue necesario entender las herramientas sobre las que trabaja Joomla, la tecnología LAMP que se ha comentado al principio. Para poder usar Joomla era necesario instalar el servidor web Apache, el sistema de gestión de bases de datos MySQL y las librerías necesarias para poder ejecutar código en PHP. A falta de un servidor donde todo esto estuviera disponible se hizo en un ordenador personal con la idea de disponer de una versión local de la Intranet en la que trabajar. Después de instalar los paquetes básicos para poder usar Joomla fue necesario descargar el paquete de Joomla y hacer una instalación en el ordenador personal.

La primera tarea fue familiarizarse con el entorno Joomla y analizar qué requisitos de la Intranet se podían satisfacer con el paquete básico de instalación. Joomla, como un gestor de contenidos que es, está pensado para crear páginas web de contenidos. Este tipo de páginas son muy abundantes, algunos ejemplos son blogs, portales de noticias, periódicos digitales, etc... Por tanto usar Joomla con el objetivo de crear un sitio web de este tipo es relativamente sencillo y no requiere la instalación de muchas extensiones. Sin embargo para crear una Intranet es necesario darle algunas funcionalidades que no tiene, aquí es donde empieza la tarea de buscar extensiones.

Tras mucho tiempo buscando y probando extensiones fue posible darle a la Intranet gran parte de su funcionalidad. Buscando en los portales y foros dedicados a los usuarios de Joomla fue posible encontrar algunas de las extensiones más básicas para el funcionamiento requerido en la Intranet. Entre las extensiones fue fácil encontrar el administrador de foros *Kunena Forum*, el *plugin* para insertar vídeos en los contenidos *AllVideos* y el componente para realizar copias de respaldo *Akeeba Backup.* Por parte de la empresa se sugirió también la instalación del componente *eXtplorer,* su búsqueda e instalación fue algo trivial. Estos componentes están bastante extendidos y su uso es muy fácil e intuitivo, por lo que no fue necesario emplear mucho tiempo para aprender a usarlos. Sin embargo los problemas surgieron al tratar de mostrar contenidos personalizados según el usuario.

Aunque el sistema de control de acceso de Joomla permite diferenciar entre varios tipos de usuarios y restringir el acceso a usuarios no registrados, este sistema era insuficiente para ofrecer vistas personalizadas como se requiere en una Intranet. Para solucionar esto, primero se hizo una búsqueda sobre extensiones que proporcionaran un control de acceso más flexible. La única extensión libre que se recomendaba en los foros de usuarios era *NoixACL*. Esta extensión daba la posibilidad de crear grupos de usuarios nuevos y restringir el acceso a módulos y componentes según el grupo. Aun así no fue suficiente porque no era posible ocultar los elementos no accesibles, simplemente se denegaba el acceso mostrando un mensaje. Al final se optó por programar un componente que permitiera controlar el contenido según el usuario y aquí comenzó una larga tarea que culminó con la programación de un componente y tres módulos que permitían controlar el acceso y constituían el principal funcionamiento de la Intranet.

Esta tarea comenzó con un largo tiempo de documentación en los foros de desarrolladores de Joomla, y el seguimiento de tutoriales tanto para desarrollar componentes y módulos como para aprender PHP, HTML, SQL y algunas nociones de JavaScript. Después de aprender lo suficiente para poder desarrollar las aplicaciones se comenzó la implementación a partir de componentes de prueba que fueron modificados de forma incremental hasta conseguir una parte administrativa, donde configurar lo que debía ver cada usuario, y la parte del usuario que mostraba los contenidos según la configuración que se había definido.

Como resultado final, se ha obtenido una Intranet donde los usuarios pueden tener acceso a los servicios requeridos según los productos que se han contratado y una parte administrativa donde se pueden poner servicios para nuevos productos, así como otorgar permisos a los usuarios para acceder a éstos. Además del servicio ofrecido, la parte administrativa ofrece la posibilidad de guardar los datos relevantes de los usuarios, así como de las empresas que son clientes.

Desarrollar una Intranet a partir de un CMS es una tarea que en un principio no requiere grandes conocimientos informáticos y que podría estar al alcance de cualquiera. Si se tiene experiencia en el uso del CMS y se tiene una idea clara de la funcionalidad que debe tener la Intranet y de como puede conseguirse dicha funcionalidad sería una tarea relativamente sencilla. Además de lo sencillo que resulta aumentar la funcionalidad de un CMS a partir de la instalación de extensiones, el desarrollo de nuevos componentes para conseguir funcionalidades más concretas no resulta una tarea muy costosa. Pero para aprovechar estas facilidades se requiere un conocimiento previo del entorno de Joomla, de las posibles extensiones que pueden ser útiles y de las herramientas de programación necesarias para implementar nuevas funcionalidades.

Sin embargo para el desarrollo de la Intranet el punto de partida era la idea de lo que se quería conseguir pero sin conocimiento previo de las herramientas necesarias para su desarrollo. Bajo estas circunstancias el desarrollo de la Intranet ha requerido un largo trabajo de documentación acerca del entorno y las posibles herramientas para llevar a cabo el proyecto. Las tareas de documentación han sido el verdadero trabajo a la hora de realizar este PFC, lo que ha llevado a adquirir valiosos conocimientos acerca sobre HTML, PHP, SQL y sobre el Framework de Joomla, también algunos conocimientos, aunque en menor medida, sobre JavaScript.

7 Bibliografía

Para el desarrollo de la Intranet se han consultado principalmente las siguientes fuentes, aunque en el transcurso de la documentación se han visitado infinidad de páginas.

[1]Licencia GPL (General Public License): GPL es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software.
General Public License [en línea]: Wikipedia. Disponible en Internet: http://es.wikipedia.org/wiki/Licencia_p%C3%BAblica_general_de_GNU

[2]Usabilidad web: Según El organismo de estandarización ISO (*International Standarisation Organization*) una posible definición del término usabilidad es la ISO 9126 (1991): "La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso."
International Standard Organization (1991). ISO 9126. Software engineering-Product Quality.

[3]*Framework:* En el desarrollo de software, un *framework* es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.
Framework [en línea]: Wikipedia. Disponible en Internet: http://es.wikipedia.org/wiki/Framework.

[4]API: Una interfaz de programación de aplicaciones o API (del inglés *application programming interface*) es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las bibliotecas.
Interfaz de programación de aplicaciones [en línea]. Wikipedia. Disponible en Internet: http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones.

[5]Hojas de estilo en cascada: Las hojas de estilo en cascada (en inglés Cascading Style Sheets), CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML.
Hojas de estilo en cascada [en línea]: Wikipedia. Disponible en Internet: http://es.wikipedia.org/wiki/CSS.

[6]Modelo relacional: El modelo relacional es una representación de las entidades que forman una base de datos. En él, todos los datos están estructurados a nivel lógico como tablas(relaciones) formadas por filas(tuplas) y columnas(atributos), aunque a nivel físico pueden tener una estructura completamente distinta. Cada instancia de la entidad encontrará sitio en una tupla de la relación, mientras que los atributos de la relación representan las

propiedades de la entidad.
Javier Quiroz. *El modelo relacional de bases de datos [en línea]*. Boletín de Política IInformáticaNúm.[6, 2003] . La esencia del modelo . Disponible en Internet: http://www.inegi.org.mx/inegi/contenidos/espanol/prensa/Contenidos/Articulos/tecnologia/relacional.pdf

Páginas de los CMS:
Typo3: http://typo3.org/download/
WordPress: http://es.wordpress.org/
Drupal: http://drupal.org/project/drupal
Joomla: http://www.joomlaspanish.org/

Descargas de extensiones:
Akeeba Backup: http://www.akeebabackup.com/download/akeeba-backup/akeeba-backup-3-1.html.
eXtplorer: http://extplorer.sourceforge.net/ .
AllVideos: http://extensions.joomla.org/extensions/multimedia/multimedia-players/video-players-a-gallery/812 .
Jumi: http://code.google.com/p/jumi/downloads/list.
Kunena Forum: http://www.kunenaspanish.com/descargas/cat_view/3-componentes

Documentación acerca de CMS:
http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_contenidos
http://es.wikipedia.org/wiki/Anexo:Sistemas_de_gesti%C3%B3n_de_contenidos
http://es.wikipedia.org/wiki/Joomla
http://es.wikipedia.org/wiki/Drupal
http://es.wikipedia.org/wiki/Typo3
http://es.wikipedia.org/wiki/WordPress

Instalación de Joomla:
http://comunidadjoomla.org/component/content/article/53-ayuda-manuales-joomla-15x/147-manual-de-instalacion-para-joomla-15x.html

Tutorial HTML:
http://gias720.dis.ulpgc.es/Gias/Cursos/Tutorial_html/guia_rap.htm#mimg

Tutorial PHP:
http://www.webestilo.com/php/

Tutorial SQL:
http://sql.1keydata.com/es/

Desarrollo Joomla:
http://www.joomlaspanish.org/foros/showthread.php?t=13795
http://api.joomla.org/elementindex_Joomla-Framework.html
http://www.nosolocodigo.com/tutorial-componente-para-joomla-15-iv
Extensiones Joomla:
http://www.joomlaspanish.org/component/option,com_remository/Itemid,28/

http://www.webempresa.com/

8 Anexos

Anexo 1: *.../views/productos/tmpl/default.php*

```
<?php defined('_JEXEC') or die('Restricted access'); ?>
<form action="index.php" method="post" name="adminForm">
<div id="editcell">
        <table class="adminlist">
        <thead>
                <tr>
                        <th width="5">
                                <?php echo JText::_( 'ID' ); ?>
                        </th>
                        <th width="5">
                                <input type="checkbox" name="toggle" value="" onclick="checkAll(<?php echo count(
$this->items ); ?>);" />
                        </th>
                        <th>
                                <?php echo JText::_( 'Nombre' ); ?>
                        </th>
                        <th>
                                <?php echo JText::_( 'ArtÃculo' ); ?>
                        </th>
                        <th>
                                <?php echo JText::_( 'logo' ); ?>
                        </th>
                        <th>
                                <?php echo JText::_( 'foro' ); ?>
                        </th>
                </tr>
        </thead>
        <?php
        $k = 0;
        for ($i=0, $n=count( $this->items ); $i < $n; $i++)  {
                $row = &$this->items[$i];

                $checked          = JHTML::_('grid.id',   $i, $row->id );
                $link_producto              = JRoute::_(
'index.php?option=com_soa&controller=producto&task=edit&cid[]='. $row->id );
                $link_presentacion          = JRoute::_( 'index.php?option=com_content&sectionid=-
1&task=edit&cid[]='. $row->articulo );?>
                <tr class="<?php echo "row$k"; ?>">
                        <td>
                                <?php echo $row->id; ?>
                        </td>
                        <td align="center">
                                <?php echo $checked; ?>
                        </td>
                        <td align="center">
                                <a href="<?php echo $link_producto; ?>"><?php echo $row->nombre; ?></a>
                        </td>
                        <td align="center">
                                <a href="<?php echo $link_presentacion; ?>"><?php echo $row->presentacion; ?></a>
                        </td>
                        <td align="center">
                                <?php echo $row->logo; ?>
                        </td>
                        <td align="center">
                                <?php echo $row->nombre_foro; ?>
                        </td>

                </tr>
                <?php $k = 1 - $k;}?>
```

Anexo 2: *.../views/producto/tmpl/form.php*

```
<?php defined('_JEXEC') or die('Restricted access'); ?>

<form action="index.php" method="post" name="adminForm" id="adminForm">
<div class="col100">
        <fieldset class="adminform">
                <legend><?php echo JText::_( 'Details' ); ?></legend>

                <table class="admintable">
                <tr>
                        <td width="100" align="right" class="key">
                                <label for="nombre">
                                        <?php echo JText::_( 'Nombre del producto' ); ?>:
                                </label>
                        </td>
                        <td>
                                <input class="text_area" type="text" name="nombre" id="nombre" size="32"
maxlength="250" value="<?php echo $this->producto->nombre;?>" />
                        </td>
                </tr>
                <tr>
                        <td width="100" align="right" class="key">
                                <label for="articulo">
                                        <?php echo JText::_( 'Artículo de presentación' ); ?>:
                                </label>
                        </td>
                        <td>
                        <select name="articulo" id="articulo" class="inputbox" size="1">

                                <?php
                                     echo '<option value="null">no asignado</option>';
                                     foreach($this->larticulos as $articulo)
                                         if ($this->producto->articulo == $articulo->id)

                                            echo '<option value="'.$articulo->id.'" selected>'.$articulo-
>title.'</option>';
                                         else
                                            echo '<option value="'.$articulo->id.'">'.$articulo-
>title.'</option>';
                                ?>

                        </select>
                        </td>
                </tr>
                <tr>
                        <td width="100" align="right" class="key">
                                <label for="logo">
                                        <?php echo JText::_( 'logo' ); ?>:
                                </label>
                        </td>
                        <td>
                        <?      $dir = JPATH_BASE.DS.'components'.DS.'com_soa'.DS.'Productos'.DS
                                .'logos'.DS;
                                if(@$handle = opendir($dir)) {
                                        echo '<select name="logo" id="logo" class="inputbox" size="1">';
                                        echo '<option value="null">no asignado</option>';
                                        while (false != ($file = readdir($handle))) {
                                                if ($file != '.' && $file != '..' && $file !='index.html') {
                                                        if ($this->producto->logo == $file)

                                                                echo '<option selected>'.$file.'</option>';
                                                        else
                                                                echo '<option >'.$file.'</option>';
                                                }
                                        }
                                        echo'</select> ';
```

```
                <tr>
                        <td width="100" align="right" class="key">
                                <label for="foro">
                                        <?php echo JText::_( 'Foro' ); ?>:
                                </label>
                        </td>
                        <td>
                        <select name="foro" id="foro" class="inputbox" size="1">

                                <?php
                                     echo '<option value="null">no asignado</option>';
                                     foreach($this->lforos as $foro)
                                         if ($this->producto->foro == $foro->id)

                                            echo '<option value="'.$foro->id.'" selected>'.$foro->name.'</option>';
                                         else
                                            echo '<option value="'.$foro->id.'">'.$foro->name.'</option>';

                                ?>

                        </select>
                        </td>
                </tr>


        </table>

                <label for="extra">
                        <?php echo JText::_( 'Descripcion del producto' ); ?>:
                </label>
                <?php
                        $editor = &JFactory::getEditor();
                        // parameters : areaname, content, width, height, cols, rows
                        echo $editor->display( 'extra',  $this->producto->extra , '50%', '300', '75', '20' ) ;
                ?>
        </fieldset>
</div>
<div class="clr"></div>

<input type="hidden" name="option" value="com_soa" />
<input type="hidden" name="id" value="<?php echo $this->producto->id; ?>" />
<input type="hidden" name="task" value="" />
<input type="hidden" name="nombre_viejo" value="<?php echo $this->producto->nombre; ?>" />
```

Anexo 3: *.../components/com_soa/controller.php*

```
<?php
defined( '_JEXEC' ) or die( 'Restricted access' );
jimport('joomla.application.component.controller');
require_once(JApplicationHelper::getPath( 'helper' , 'com_soa' ));

class SoaController extends JController
{
	function __construct()
	{
		parent::__construct();

	}

	function cambio(){
		$articulo = $_POST["cambiar"];
		if($articulo != 0){
			$pagina = 'Location: index.php?option=com_content&view=article&id='.$articulo;
			$this->redirigir($pagina);
		}
		else $this->presentacion();
	}

	function redirigir($pagina)
	{
		header($pagina);
	}

	function ayuda()
	{
		$id_producto = JRequest::getVar( 'prod' );
		JRequest::setVar('producto',$id_producto);
		$producto = new Producto($id_producto,'');
		$versiones = $producto->getListaVersiones();

		foreach ($versiones as $version){
			echo '<br><a href="index.php?option=com_soa&task=mostrar_ayuda&ver='.$version->datos->id
			     .'">Ayuda para la version '.$version->datos->version.'</a><br><br>';
		}
	}

	function mostrar_ayuda(){
		$id_Version = JRequest::getVar( 'ver' );
		$version = new Version($id_Version,'');
		$producto = new Producto($version->datos->id_producto);
		JRequest::setVar('producto',$producto->datos->id);
	    if($version->datos->ayuda){
			$pag = 'administrator'.DS.'components'.DS.'com_soa'.DS.'Productos'.DS.$producto->datos-
>nombre.DS
			        .'Ayuda'.DS.$version->datos->ayuda;
			echo '<iframe src ="'.$pag.'" width="100%" height="600">';
			echo '<p>Texto alternativo para navegores sin soporte para iframes.</p>';
			echo '</iframe>';
		}
	}

	function descargar()
	{
		$id_producto = JRequest::getVar( 'prod' );
		if(comprobarLicencia($id_producto)){
			JRequest::setVar('producto',$id_producto);
			$producto = new Producto($id_producto,'');
			$descargas =  getTabla_descargas($producto->getListaVersiones());
			echo $descargas;
		}
		else{
```

```
Function descargar_version()
{
        $id_Version = JRequest::getVar('ver');
        $version = new Version($id_Version,'');
        JRequest::setVar('producto',$version->datos->id_producto);
        $producto = new Producto($version->datos->id_producto,'');
        if(comprobarLicencia($version->datos->id_producto) || $version->datos->libre == 'Si')            {

                descarga_producto($version,$producto);
        }
        else{
                echo ' NO TIENES LICENCIA PARA ESTE PRODUCTO<BR>';
                $pagina = "Refresh: 4; URL=index.php";
                $this->redirigir($pagina);
        }
}

function contar()
{
        $version = new Version(JRequest::getVar('ver'),'');

        if(comprobarLicencia($version->datos->id_producto) || $version->datos->libre == 'Si')            {
                descargado($version);
                $enlace = $version->enlace();
                $pagina = "Refresh: 1; URL=$enlace";
                header ($pagina);
                $this->descargar_version($version->datos->id);
        }
        else{
                echo ' NO TIENES LICENCIA PARA ESTE PRODUCTO<BR>';
                $pagina = "Refresh: 4; URL=index.php";
                $this->redirigir($pagina);
        }
}

 function actualizar_tablas()
 {
        $user = JFactory::getUSER();
        if($user->gid != 25){
                echo 'Esta tarea solo esta disponible en modo Super Administrador';
        }else{
                $db = & JFactory::getDBO();
                $query="";
                $db->setQuery($query);
                if(!$db->query())
                        echo $db->getErrorMsg();
        }
 }

 function presentacion()
 {
        $listas = listasSoa::getInstance();
        $lproductos = $listas->getProductos();
        $licencias = $listas->getLicencias();

        foreach($licencias as $licencia){
```

Anexo 4: .../components/com_soa/helper.php

```
<?php

defined( '_JEXEC' ) or die( 'Restricted access' );

class listasSoa
{
	var $mySelf = null;
	var $listaProductos = array();
	var $listaVersiones = array();
	var $listaLicencias = array();

	static function getInstance()
	{
		if($mySelf == null)
			$mySelf = new listasSoa();

		return $mySelf;
	}

	function queryVersiones()
	{
		$user = Jfactory::getUSER();
		$query = ' SELECT DISTINCT  V.*, P.nombre as nombre_producto'
			. ' FROM #__OProductos P, #__OLicencias L, #__OClientes C, #__OVersiones V '
			. ' WHERE (L.id_producto = P.id AND L.id_cliente = C.id AND C.id_usuario = '.$user->id
			. ' AND V.id_producto=P.id AND V.nuevo like "Si") OR (V.libre like "Si" AND
P.id=V.id_producto)'
			. ' ORDER BY V.libre, V.id_producto'
		;
		return $query;
	}

	function getVerDis()
	{
		$db = &JFactory::getDBO();
		if (empty( $this->listaVersiones ))
		{
			$query = $this->queryVersiones();

			$db->setQuery( $query );
			$this->listaVersiones = $db->loadObjectList();

		}
		return $this->listaVersiones;
	}

	function queryProductos()
	{
		$user = JFactory::getUSER();
		$query = ' SELECT  P.*'
			. ' FROM #__OProductos P, #__OLicencias L, #__OClientes C '
			. ' WHERE L.id_producto = P.id  AND L.id_cliente = C.id AND C.id_usuario = '.$user->id
		;
		return $query;
	}

	function getProductos()
	{
		$db = &JFactory::getDBO();
		if (empty( $this->listaProductos ))
		{
			$query = $this->queryProductos();
			$db->setQuery( $query );
			$datos = $db->loadObjectList();
```

```
function getLicencias()
	{
		if (empty( $this->listaLicencias ))
		{
			$user = JFactory::getUSER();

			$db = &JFactory::getDBO();
			$query = 'SELECT L.* FROM #__OLicencias L, #__OClientes C '
				.'WHERE L.id_cliente=C.id AND C.id_usuario='.$user->id;

			$db->setQuery( $query );
			$this->listaLicencias = $db->loadObjectList();
		}
		return $this->listaLicencias;
	}
}

class Producto
{
	var $datos;
	var $versiones;
	var $licencia;

	function Producto($id, $data)
	{
		if(empty($data)){
			$db = &JFactory::getDBO();
			$query = " SELECT * FROM #__OProductos  WHERE id = {$id} ";
			$db->setQuery($query);
			$this->datos = $db->loadObject();
		}
		else{
			$this->datos = $data;
		}
	}

	function getLicencia()
	{
		if(empty($this->$licencia)){
			$user = Jfactory::getUSER();
			$db = &JFactory::getDBO();
			$query = 'SELECT L.* FROM #__OLicencias L, #__OClientes C '
				.'WHERE L.id_producto='.$this->datos->id
				.' AND L.id_cliente=C.id AND C.id_usuario='.$user->id;
			$db->setQuery($query);
			$this->licencia = $db->loadObject();
		}
		return   $this->licencia;
	}

	function getListaVersiones()
	{
		if(empty($this->versiones)){

			$db = &JFactory::getDBO();
			$query = " SELECT DISTINCT V.*
				    FROM  #__OVersiones V
				   WHERE V.id_producto = {$this->datos->id} ";
			$db->setQuery($query);
			$datos = $db->loadObjectList();
			foreach($datos as $version)
				$this->versiones[] = new Version('',$version);

		}
		return $this->versiones;
	}
```

```
class Version
{
	var $datos;
	var $licencia;

	function Version($id, $data)
	{
		if(empty($data)){
			$db = &JFactory::getDBO();
			$query = " SELECT V.*, P.nombre as nombre_producto
				FROM #__OVersiones V, #__OProductos P  WHERE V.id = {$id} AND V.id_producto =
P.id ";
			$db->setQuery($query);
			$this->datos = $db->loadObject();
		}
		else{
			$this->datos = $data;
		}
	}


	function enlace(){

		$enlace = 'administrator'.DS.'components'.DS.'com_soa'.DS.'Productos'.DS.$this->datos-
>nombre_producto;
		$enlace = $enlace.DS.$this->datos->url_descarga;
		return $enlace;
	}
}

function getTabla_descargas($lista_versiones)
{

	$tabla = '<TABLE CELLPADDING=5 CELLSPACING=5 ALIGN="center" >'
		.'<div style="padding-left: 1px" >'
		.'<tr><thead ALIGN="center"><th ALIGN="center">Producto<th ALIGN="center">'
		.'Version<th ALIGN="center">Tipo<th ALIGN="center">Fecha de publicacion<th ALIGN="center">'
		.'Cerrada<th ALIGN="center">mantenimiento</thead></tr>';

	foreach($lista_versiones as $version){
		$tabla = $tabla. '<tr><td ALIGN="center">'.$version->datos->nombre_producto.'<td align="center">'
		.$version->datos->version.'<td ALIGN="center">'.$version->datos->tipo.'<td ALIGN="center">'
		.$version->datos->fecha_publicacion.'<td ALIGN="center">'.$version->datos->cerrada
		.'<td ALIGN="center">'.$version->datos->mantenimiento
		.'<td ALIGN="center"><a href="index.php?option=com_soa&task=descargar_version&ver='
		.$version->datos->id.'">descargar</td></tr>';
		}
	$tabla= $tabla.'</div></TABLE>';

	return $tabla;
}

function descarga_producto($version,$producto){

     echo '<TABLE CELLPADDING=5 CELLSPACING=5 >';
	 echo '<div style="padding-left: 1px" >';
	 echo '<tr><td>';
	 $producto->show_logo();
	 echo '<td ALIGN="center">'.$version->datos->nombre_producto.'<br>version '.$version->datos->version.'<br>';
	 echo 'Total descargas:  '.$version->datos->descargas.'</td></tr>';
	 echo '<tr ><td collspan=2>'.'<a href="index.php?option=com_soa&task=contar&ver='
		.$version->datos->id.'">descargar <IMG SRC ="images/filesave.png" align="center"></a>';

	 echo '</div>';
	 echo '</TABLE>';
	 echo $version->datos->extra;
    }
```

```
function descargado($version){
        $db = & JFactory::getDBO();

        $version->datos->descargas++;
        $query = "UPDATE  #__OVersiones V SET V.descargas={$version->datos->descargas}
                  WHERE V.id={$version->datos->id}";
        $db->setQuery( $query );
        $db->query();
   }

function  comprobarLicencia($id_producto){
        $db = & JFactory::getDBO();
        $user = JFactory::getUSER();

        $query = "SELECT L.* FROM #__OLicencias L, #__OClientes C
                  WHERE  C.id_usuario = {$user->id} AND L.id_cliente = C.id AND L.id_producto = {$id_producto} ";
        $db->setQuery( $query );
        $result = $db->loadRowList();

        if(!empty( $result))
                return true;
        else return false;
}

function comprobar_acceso_foro($idforo) {
        $db = & JFactory::getDBO();

        $query = "SELECT id,parent FROM #__fb_categories WHERE published='1'";
        $db->setQuery($query);
        $lista_foros = $db->loadObjectList('id');
        $foro = $lista_foros[$idforo];
       $query = ' SELECT P.id,P.foro FROM #__OProductos P';
       $db->setQuery($query);
       $all_productos =  $db->loadObjectList('id');

        $listas = listasSoa::getInstance();
        $lproductos_contratados = $listas->getProductos();
        $foro_padre = $foro->parent;

        while($foro_padre != 0){

                foreach($all_productos as $producto){
                        if($producto->foro == $foro->id){
                                $acceso = false;
                                foreach($lproductos_contratados as $contratado){
                                        if($contratado->datos->foro == $foro->id)
                                                $acceso = true;
                                }
                                return $acceso;
                        }
                }
                $foro = $lista_foros[$foro_padre];
                $foro_padre = $foro->parent;
        }
        return true;
   }

   function presentar_licencia($licencia, $producto){
        echo '<TABLE  CELLPADDING=7 CELLSPACING=5 BORDER>';
        echo '<thead><tr><td rowspan="2">';
        $producto->show_logo();
        echo '</td><th align="center">Numero de Licencia</th><th align="center">Tipo'
             .'</th><th align="center">Licencias adquiridas</th><th align="center">Fecha Inicio'
             .'</th><th align="center">Fecha Fin</th><th align="center">Precio</th><th align="center">'
             .'Renovaciones</th></tr><td align="center">'.$licencia->NumeroLicencia.'</td><td align="center"
width="100">'
             .$licencia->Tipo.'</td><td align="center">'.$licencia->Cantidad.'</td><td align="center">'
             .$licencia->FechaInicio.'</td><td align="center">'.$licencia->FechaFin.'</td><td align="center">'
```