ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO DE TELECOMUNICACIÓN,
ESPECIALIDAD EN SONIDO E IMAGEN

Título del proyecto:

SOLUCIÓN INTEGRAL PARA EL PORTAL DE COMERCIO ELECTRÓNICO WWW.TRAJESDELUCES.COM

Alumno: Daniel Biedma Ramos

Tutor: Marko Galarza

Pamplona, 16 Noviembre 2012

ÍNDICE

1.INTRODUCCIÓN.

TRAJES DE LUCES es un proyecto de Paco Ramos, profesional taurino que ha vivido en primera persona desde niño el mundo del toro. Sabe como muchos otros la sensación única que siente un torero estando a gusto con el toro.

También ha conocido la cara menos amable: ha derramado su sangre en la plaza y a punto estuvo de perder la vida, respeta y ama profundamente el toreo. Transmite en sus creaciones la esencia de sus vivencias, en ellas quiere impregnar con la magia de la tauromaquia a quienes aman o no este mundo.

Esta tienda taurina nace en el 2008, y poco a poco va creciendo hasta que en 2012 debido a la aceptación del publico va a tomar un rumbo diferente.

Va a dejar de ser tienda una tienda taurina para convertirse en una firma de moda taurina, para ello remodelaran la web dándole un aspecto más apropiado.

Su tienda online esta hecha con el gestor de contenidos Joomla 1.5 y Virtuemart 1. Para mejorar la web y a la vez ganar velocidad de gestión de la web actualizaremos el gestor a la versión 2.5 y Virtuemart 2, incrementando la seguridad y estabilidad del sitio.

Además de todo esto, el cliente tiene pensado en un futuro dar cursos y charlas, para ello realizaremos un sencillo componente para Joomla 2.5. Con esto lograremos una gestión de los eventos así como integrarlos en su web.

2.OBJETIVO.

El objetivo de este PFC es ofrecer una solución global a Trajesdeluces SL que consista en, actualización, mejora estética de la web, y creación de un portal para poder gestionar las actividades que tiene proyectadas.

Se desarrollará una migración del sistema Joomla a la última versión y del sistema de tienda online que utilizan (Virtuemart) también a una versión superior, todo esto tanto para mejorar su gestión de la tienda (mas rapidez y estabilidad), como para ofrecerles más seguridad ante ataques informáticos.

Además, se llevará a cabo la creación de una plantilla la cual será óptima para el SEO (Posicionamiento natural en buscadores). Por otra parte, su aspecto será limpio y acorde con el giro que ha dado la empresa hacia la creación de no sólo una tienda taurina, sino una firma de moda taurina.

Al mismo tiempo, se implementará un componente para Joomla, que tendrá como objetivo la gestión de las actividades que tienen proyectadas y también un plugin, que dotará a la web de una comodidad para los usuarios que podrán iniciar sesión con el e-mail y con el usuario.

Todo esto se logrará teniendo en cuenta los patrones de desarrollo de Joomla, así como el modelo MVC (Modelo-Vista-Controlador) que se utiliza a día de hoy en cualquier lenguaje de programación de nivel superior.

3.METODOLOGÍA.

Comenzaremos haciendo una breve explicación de las herramientas y tecnologías utilizadas para alcanzar dicho objetivo. Consideraremos el siguiente orden:

- TECNOLOGÍAS COMPLEMENTARIAS PARA EL DESARROLLO WEB.
  Comenzaremos dando un repaso a las tecnología utilizadas actualmente para el diseño web. Instalaremos un servidor local, para ir haciendo las pruebas.
- JOOMLA.
  Una vez acabado el repaso, aprenderemos como funciona Joomla internamente, que es el patrón MVC y como utilizarlo, así como su funcionamiento externo.
- DESARROLLO DEL COMPONENTE.
  Pasaremos ahora al desarrollo del componente. Con el patrón MVC, y siguiendo los patrones de desarrollo en Joomla crearemos el componente.
- DESARROLLO DE LA PLANTILLA.
  Al haber desarrollado el componente, repasado las tecnologías y entendiendo como funciona Joomla, pasaremos a la creación de la plantilla.
- MIGRACION DEL SISTEMA.

  Por ultimo, migraremos la web a una versión superior, e instalaremos los módulos desarrollados. Utilizaremos para esto un servidor local, y una vez realizado todo con éxito haremos el cambio en el hosting del cliente. Siempre asegurándonos el funcionamiento (pues al ser un sistema ya en marcha, debemos cuidarnos de errores).

4. JOOMLA.

En este capítulo se presenta el Sistema de Gestión de Contenidos de Joomla, dando una pequeña introducción de ésta, cuál es su arquitectura de desarrollo, que son los componentes, como parte fundamental para la extensión de Joomla, su implementación y un ejemplo de estructura del componente.

4.1 Introducción.

Un sistema de Gestión de Contenidos CMS (Content Management System) es una aplicación que permite mantener una infraestructura Web. Con la interfaz se mantiene de forma independiente tanto el contenido como el diseño. Lo que permite en cualquier momento variar el diseño y la presentación sin necesidad de reformatear todo el contenido.

Joomla es un sistema de Gestión de Contenidos CMS de código abierto (gratuito tanto para su uso como para su desarrollo) que permite la edición de contenido de un sitio Web de una forma sencilla. Está programado en PHP y requiere de una base de datos MySQL, así como de un servidor HTTP sobre el que funciona.

Sus principales características de la instalación básica son:

- Publicación de Contenidos organizados por Categorías y Secciones.
- Editor WYSIWYG es el acrónimo de What You See Is What You Get (en inglés, "lo que ves es lo que obtienes").
- Posibilidad de definir tantos menús y submenús como necesites.
- Administración de imágenes y ficheros, tantas imágenes y ficheros como necesites.
- Administración de usuarios para crear contenidos específicos u accesos a determinadas partes de tu web solo para usuarios registrados.
- Encuestas que podrás definir y mostrar en tu web para que los usuarios voten y tengas estadísticas.
- Diseño basado en plantillas.
- Creación de módulos adicionales para poder colocar contenidos en determinadas partes de tu web.
- Sindicación de noticias para poder publicar tus contenidos RSS automáticamente.
- Gestión de Banners que te permitirán tener publicidad en tu web.

- Posibilidad de instalar idiomas, o hacer traducciones a medida.

Todas estas características crecen de manera considerable, gracias a la posibilidad de incorporar diferentes extensiones.

Existen diferentes tipos de extensiones:

- Plantillas: Cambian el formato de la Web
- Paquetes de idiomas: Permite la implementación de pagina multi-idioma
- Componentes: Los componentes son aplicaciones independientes entre sí que gestionan la información dentro de Joomla, pueden contener a su vez Módulos y plugins.
- Módulos: Son aplicaciones pequeñas que muestran información en un sitio especifico de la web.
- Plugins: Son aplicaciones aun mas pequeñas que amplían las funcionalidades del Código base de Joomla.

4.2 El Patrón MVC.

El patrón de diseño de software que utiliza Joomla es el patrón Modelo, Vista, Controlador MVC (Model View Controller) (Figura 1).

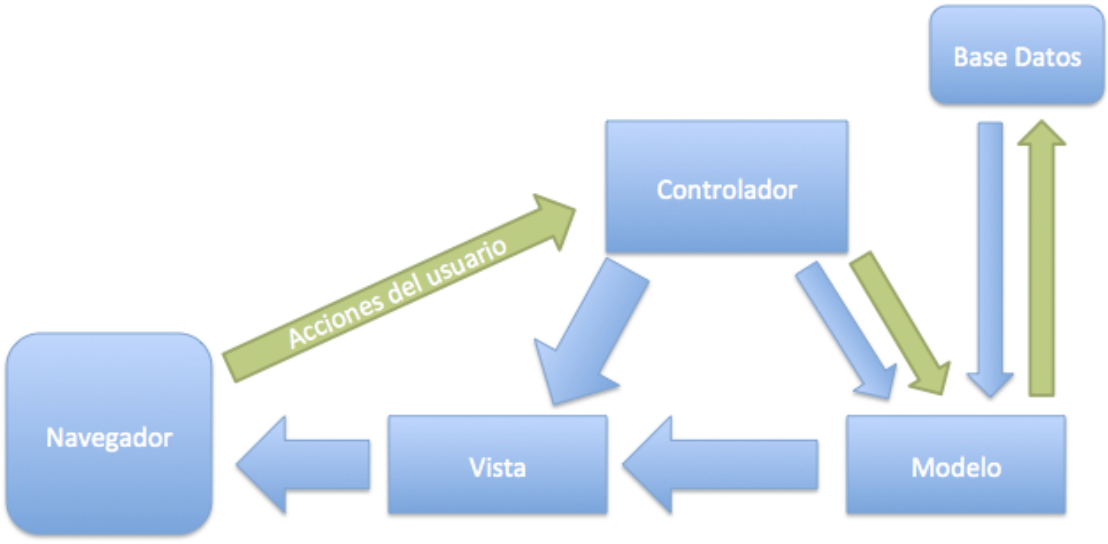


Figura 1. Esquema MVC

El Modelo, Vista, Controlador MVC, es un patrón de arquitectura del software en el que se separan los datos que requiere la aplicación, de la lógica de

control y ésta a su vez de la presentación. Este patrón es muy común en las aplicaciones web.

- *El Modelo*: Contiene el código necesario para suministrar los datos que le sean solicitados, independientemente de que estos provengan de una base de datos, ficheros XML o cualquier otro soporte con el fin de generar la vista solicitada.
- *La Vista*: En ella se incluye el código correspondiente a la presentación del modelo, posibilitando además la interacción del usuario con el controlador, se corresponde con la interfaz de usuario.
- *El Controlador*: Es el punto de entrada de la aplicación y el que ejecuta la mayor parte de la lógica de la aplicación. Permanece en ciclo continuo, atendiendo las peticiones recibidas a través de la interfaz, e interactúa con el modelo, actualizándolo en función de la petición recibida, con el fin de generar la vista solicitada por el usuario.

4.3 Creación de un Componente.

Los componentes son fundamentales en Joomla, todo lo que se muestra en la interfaz utilizando Joomla se realiza a través de éstos. El componente se encarga de generar el contenido que se visualiza.

En Joomla podemos observar dos lados claramente diferenciados:

- Presentación ó Front-End: Es la parte visible para el usuario.
- Administración ó Back-End: Es la parte que se hace visible cuando se accede como administrador para gestionar el sitio.

Esta forma de trabajar en Joomla afecta al desarrollo de los componentes, puesto que una vez desarrollado tendrá una parte de visualización para el usuario y otra para la administración.

No es obligatoria la existencia de las dos partes de un componente, todo depende de la finalidad del éste.

Dentro de la estructura de directorios de Joomla los componentes se ubican en sitios diferentes: la parte de presentación se ubica partiendo de la raíz de la instalación de Joomla en el directorio *components*, mientras que la parte de administración se ubica en *administrator/components*.

Una vez dentro del directorio *components* (cualquiera de los dos) se puede observar la existencia de un conjunto de subdirectorios, cada uno de estos subdirectorios es un componente, que ha sido instalado por defecto durante el proceso de instalación de Joomla. Así mismo, se puede comprobar que todos siguen un patrón y es que todos tienen el prefijo com_ y a continuación el nombre del componente. Esto nos lleva a la primera norma que impone Joomla.

Por ejemplo, para la creación de un componente que denominaremos "prueba", se debe ubicar dentro de un directorio que se denomine com_nombredelcomponente, por ello, hemos de crear un directorio denominado com_prueba.

Una vez creado el directorio, creamos la estructura del componente, es decir, todos aquellos ficheros y directorios que componen el despliegue del mismo, dado que hemos de seguir el modelo de desarrollo MVC. Los ficheros los creamos en blanco y posteriormente los iremos completando.

Para ello crearemos en el directorio raíz de nuestro componente el fichero prueba.php. El fichero tiene que llamarse igual que el nombre del componente, dado que Joomla, cuando intente cargar el componente lo que hará será buscar en el directorio de nuestro componente el nombre de archivo con extensión php (el desarrollo en Joomla se realiza con PHP) que coincida con nuestro componente, en nuestro caso, prueba.php. Éste será el punto de entrada de nuestra aplicación.

En el mismo directorio crearemos además el fichero controller.php, que será llamado por el fichero de entrada a la aplicación y será el encargado de atender todas las peticiones de usuario y realizar las diferentes tareas que éste requiera de la aplicación. Contendrá la parte de desarrollo correspondiente al apartado Controlador del patrón MVC.

A continuación, para completar el patrón, y partiendo de la raíz de nuestro componente se crearán los directorios *models* y *views* que contendrán la parte correspondiente al Modelo y la Vista del patrón MVC.

Nos situamos dentro del directorio *models*, que será donde se encontrarán los posibles modelos de acceso a datos con el fin de recuperar y/o modificar información y preparar los datos para entregárselos a la vista. Crearemos un fichero denominado prueba.php siguiendo la nomenclatura de Joomla de poner en cada una de las tres partes del patrón MVC, el mismo nombre de fichero que el del componente.

Si además de la conexión con base de datos fuera necesario el acceso a otro tipo de tablas o ficheros de datos (XML, texto, etc.) adicionales de la aplicación, podríamos crear más directorios si fuera necesario.

Tras la creación del modelo y el controlador, pasamos a crear la parte de la estructura correspondiente a la vista. Nos situamos en el directorio *views* y creamos un directorio con el mismo nombre que el componente *prueba*, dentro del cual, a su vez, creamos un fichero denominado view.html.php que obtiene la información del modelo. Quien ha obtenido la información previamente de la Base de Datos, y una vez obtenida dicha información pudiendo incluso modificarla, la pone a disposición de la vista a través de la plantilla (layout). Esta será la encargada de transformar la vista en el formato solicitado por el usuario(HTML, PDF, ...). Dentro del directorio de prueba, se crea un directorio tmpl (template) que contiene las diferentes plantillas que se pueden utilizar para generar la vista solicitada por el usuario. Cuando solamente existe una plantilla, ésta tiene un nombre por defecto para Joomla. En nuestro caso como sólo existirá una plantilla, se crea dentro del directorio tmpl, el fichero default.php.

Una vez realizada la secuencia de pasos descrita hasta ahora, ya tenemos establecida la estructura básica en la creación del componente. La imagen siguiente muestra gráficamente dicha estructura.

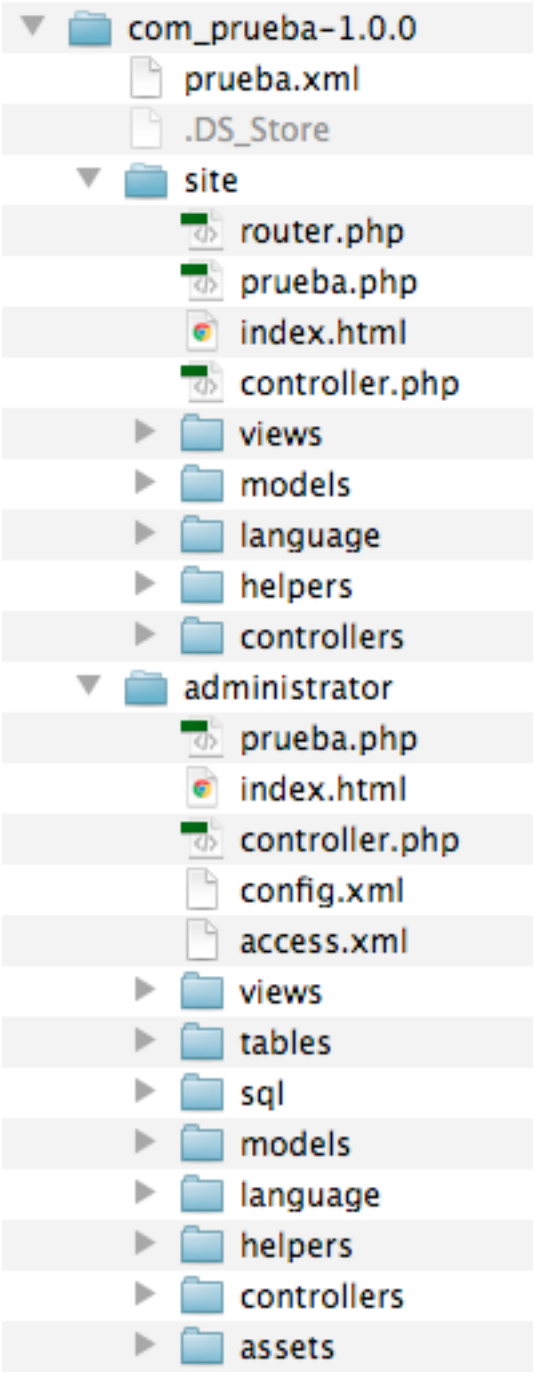


Figura 2 Componente de prueba Estructura Básica.

El archivo “prueba.xml” contiene tanto la información del componente, como las carpetas que contiene para su instalación, es el archivo que lee Joomla al instalarlo. Todas las extensiones de Joomla lo tienen, así Joomla sabe diferenciar qué tipo de componente es y dónde debe ubicarlo.

Un ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<extension type="component" version="1.6.0" method="upgrade">
    <name>com_prueba</name>
```

```
    <creationDate>2012-10-23</creationDate>
    <copyright>Copyright (C) 2012. All rights reserved.</copyright>
    <license>GNU General Public License version 2 or later; see
LICENSE.txt</license>
    <author>dani</author>
    <authorEmail>danibram@hotmail.com</authorEmail>
    <authorUrl>http://www.danibram.es</authorUrl>
    <version>1.0.0</version>
    <description>Prueba para el PFC</description>

    <install> <!-- Runs on install -->
        <sql>
            <file driver="mysql"
charset="utf8">sql/install.mysql.utf8.sql</file>
        </sql>
    </install>
    <uninstall> <!-- Runs on uninstall -->
        <sql>
            <file driver="mysql"
charset="utf8">sql/uninstall.mysql.utf8.sql</file>
        </sql>
    </uninstall>

    <files folder="site">
        <filename>index.html</filename>
        <filename>prueba.php</filename>
        <filename>controller.php</filename>
        <filename>router.php</filename>
        <folder>views</folder>
        <folder>models</folder>
        <folder>controllers</folder>
        <folder>helpers</folder>
    </files>
    <languages folder="site">
        <language tag="en-GB">language/en-GB.com_prueba.ini
        </language>
    </languages>
    <administration>
        <menu img="components/com_prueba/assets/images/s_com_prueba.png"
>COM_PRUEBA</menu>
        <submenu>

            <menu link="option=com_prueba&amp;view=pfcs" view="pfcs"
img="components/com_prueba/assets/images/s_pfcs.png"
alt="Prueba/Pfcs">COM_PRUEBA_PFCS</menu>

        </submenu>
        <files folder="administrator">
            <filename>access.xml</filename>
            <filename>config.xml</filename>
            <filename>controller.php</filename>
            <filename>index.html</filename>
            <filename>prueba.php</filename>
            <folder>controllers</folder>
            <folder>assets</folder>
            <folder>helpers</folder>
            <folder>models</folder>
            <folder>sql</folder>
            <folder>tables</folder>
```

```
            <folder>views</folder>
        </files>
        <languages folder="administrator">
            <language tag="en-GB">language/en-GB.com_prueba.ini
            </language>
            <language tag="en-GB">language/en-GB.com_prueba.sys.ini
            </language>
        </languages>
    </administration>
</extension>
```

5.Tecnologias Complementarias.

5.1 PHP.

PHP es un acrónimo que significa *PHP Hypertext Pre-processor*. Es un lenguaje de programación del lado del servidor diseñado principalmente para el desarrollo web de contenido dinámico.

Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP puede ser usado en todos los servidores web sin ningún costo adicional.

5.2 CSS.

CSS son las siglas de Cascading Style Sheets - Hojas de Estilo en Cascada - que es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe como se va a mostrar un documento en pantalla, por impresora, por voz o en dispositivos táctiles basados en Braille.

¿Para qué sirve?

CSS es una especificación desarrollada por el W3C (World Wide Web Consortium) para permitir la separación de los contenidos de los documentos escritos en HTML, XML, XHTML, SVG, o XUL de la presentación del documento con las hojas de estilo, incluyendo elementos tales como los colores, fondos, márgenes, bordes, tipos de letra..., modificando así la apariencia de una página web de una forma más sencilla, permitiendo a los desarrolladores controlar el estilo y formato de sus documentos.

¿Cómo funciona?

El lenguaje CSS se basa en una serie de reglas que rigen el estilo de los elementos en los documentos estructurados, y que forman la sintaxis de las hojas de estilo. Cada regla consiste en un selector y una declaración, ésta última va entre corchetes y consiste en una propiedad o atributo, y un valor separados por dos puntos.

5.3 HTML.

HTML, siglas de HyperText Markup Language («lenguaje de marcado de hipertexto»), hace referencia al lenguaje de marcado predominante para la elaboración de páginas web que se utiliza para describir y traducir la estructura y la información en forma de texto, así como para complementar el texto con objetos tales como imágenes. El HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta cierto punto, la apariencia de un documento y puede incluir un script (por ejemplo JavaScript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

5.4 XML.

XML proviene de eXtensible Markup Language (“Lenguaje de Marcas Extensible”). Se trata de un metalenguaje (un lenguaje que se utiliza para decir algo sobre otro lenguaje) extensible de etiquetas que fue desarrollado por el Word Wide Web Consortium.

El XML es una adaptación del SGML (Standard Generalized Markup Language), un lenguaje que permite la organización y el etiquetado de documentos. Esto quiere decir que el XML no es un lenguaje en sí mismo, sino un sistema que permite definir lenguajes de acuerdo a las necesidades. El XHTML, el MathML y el SVG son algunos de los lenguajes que el XML está en condiciones de definir.

Las bases de datos, los documentos de texto, las hojas de cálculo y las páginas web son algunos de los campos de aplicación del XML. El metalenguaje aparece como un estándar que estructura el intercambio de información entre las diferentes plataformas.

5.5 JQuery.

Es una librería de funciones escritas en JavaScript, que posibilita interactuar con los documentos HTML, el árbol del DOM y la tecnología AJAX que implementan los diferentes navegadores web existentes, de manera transparente a las características especificas de cada uno de ellos. Es decir, hace independiente el uso de los recursos de los navegadores Web de la forma en los que el fabricante haya definido como han de ser utilizados estos.

Aunque el estándar W3C ha definido estándares tanto para el uso de HTML como del DOM y de AJAX, la forma de acceso a los recursos de cada agente de usuario (navegador) ha sido establecida por sus creadores y no siempre se

corresponde con el estándar. Es por ello que desarrollar código en la parte cliente se hace complejo dadas las peculiaridades y especificidades de cada navegador.

Con la finalidad de evitar esta serie de restricciones, JQuery se desarrolla como un conjunto de funciones con una sintaxis propia, que implementa toda la funcionalidad necesaria para acceder a las tecnologías que se utilizan en los clientes de navegación, adaptando su ejecución a la tecnología que lo ejecuta.

Su finalidad es la de facilitar con una sintaxis sencilla y un conjunto de funciones comunes la forma en gestionar los diferentes eventos y funcionalidades de los agentes de usuario, tanto eventos, estilos, propiedades y características.

5.6 JTable.

Se trata de una clase abstracta que forma la base para la construcción de objetos de la clase tabla en Joomla. Para cada tabla que se desee implementar es necesario crear una nueva clase extendiendo JTable. Se trata de la implementación del patrón de diseño de registro activo y permite la construcción de una tabla física en Joomla, con el fin de leer, actualizar y eliminar registros de tabla de base de datos.

Es de gran utilidad cuando se trata de mantener registros en memoria o en una estructura física del servidor que requiera del uso de tablas de trabajo, sin necesidad de hacer uso de ningún gestor de base de datos.

6. DESARROLLO DE LA PLANTILLA.

En este capítulo se presenta como realizar una plantilla en Joomla, dando una pequeña introducción de qué es una plantilla, cuál es su estructura, la técnica básica para su implementación, y el ejemplo de una plantilla.

6.1 Introducción.

Una plantilla para un gestor de contenidos es la manera en que a nosotros nos va a mostrar en la pantalla el resultado de nuestra interacción con la web, en este sentido el diseño de la plantilla no difiere mucho del diseño de una página web normal, sólo que al diseñar la web normal ya está añadiendo el contenido directamente, y la plantilla no es más que un contenedor visual, como si de una web sin contenido se tratara.

El proceso de creación tiene varios pasos comunes. La creación de la estructura (Figura 1), por ejemplo, podemos observar una estructura que utilicé para este caso en concreto, comparándola con la de la web en la Figura 2

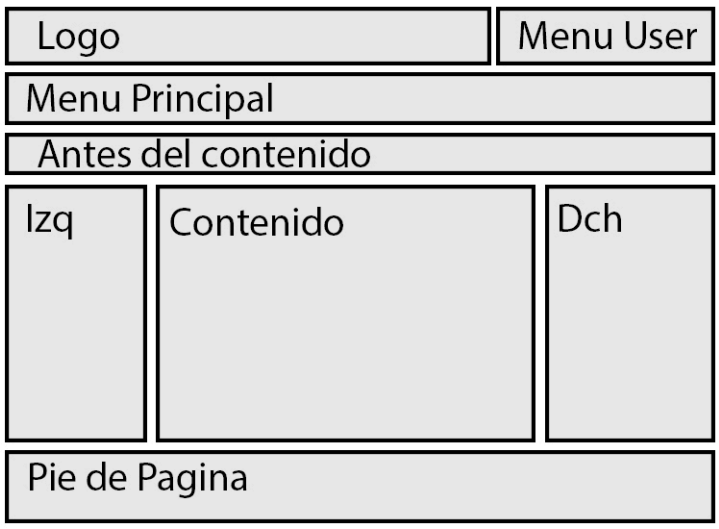


Figura 3 Estructura Visual

Figura 4 Pagina Web Estructura Visual II

6.2 Estructura de la Plantilla.

Ésta es la estructura de archivos de una plantilla en Joomla. La Figura 3 muestra la estructura básica, las plantillas podrán tener más carpetas según las necesidades.

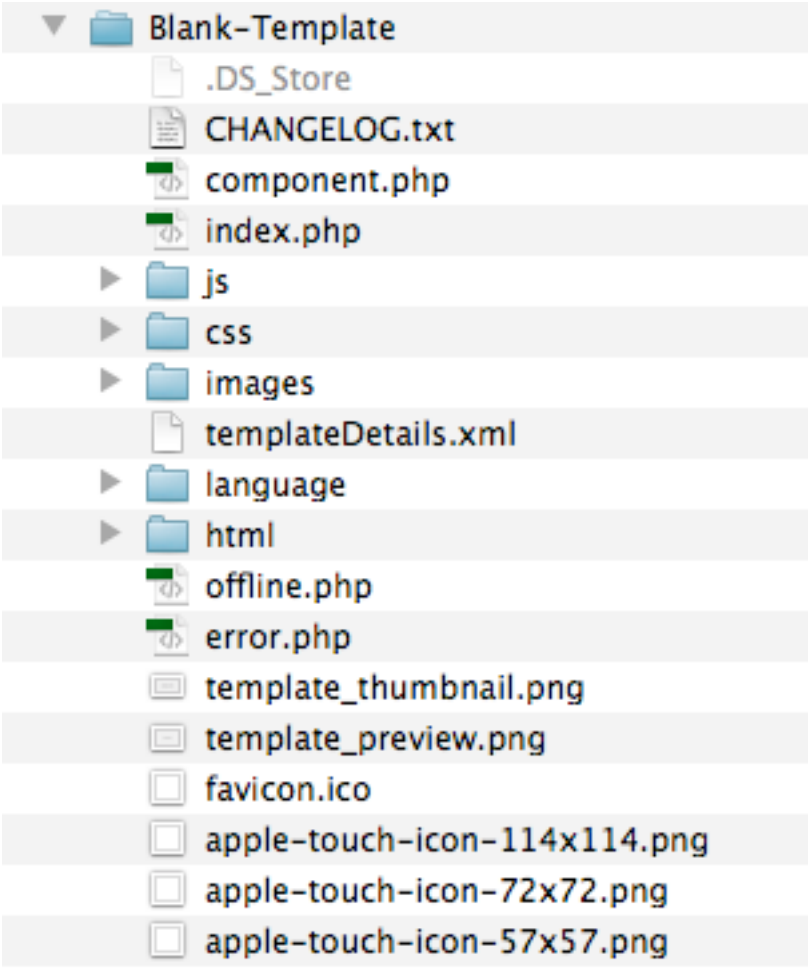


Figura 5 Estructura de Plantilla

Carpetas y archivos obligatorios

Obligatorios pues si ellos la plantilla no funcionará.

- **index.php**: Archivo principal de la plantilla. Será donde se llamará a los archivos CSS y JavaScript. Su contenido está formado por etiquetas HTML y PHP.
- **templateDetails.xml**: Archivo XML que sirve para instalar de manera automática la plantilla en Joomla! Posee el nombre de la plantilla, autor, licencia, versión, estructura de carpetas, archivos y lenguajes, nombres de las posiciones de módulos y opciones de configuración de la plantilla. Joomla leerá este archivo y sabrá los que contiene esta plantilla y como instalarla.
- **template_preview.png**: Imagen de la plantilla, que será la que saldrá en el listado de plantillas.

- **template_thumbnail.png**: Pequeña imagen que también actúa como vista previa al momento de seleccionar una plantilla en el panel de administración.
- **index.html**: Archivo HTML en blanco que nos servirá como método de seguridad en servidores que permiten explorar directorios en archivos desde el navegador.
- **css**: Carpeta contenedora de los estilos CSS.

Carpetas y archivos no obligatorios

Aquí ponemos una lista de los archivos no obligatorios, puede haber más según necesitemos añadir contenido.

- **component.php**: Archivo PHP que se usa como vista previa al querer imprimir un artículo o enviarlo por correo electrónico. En caso de no existir, se utiliza el archivo component.php ubicado en la carpeta /templates/system/. Este método se llama override, que lo explicare mas abajo.
- **error.php**: Archivo PHP que se mostrará cuando en el CMS ocurra un error.
- **favicon.ico**: Imagen que se utilizará como icono de la página.
- **images**: Carpeta con imágenes para utilizar en la plantilla.
- **js**: Carpeta con archivos JavaScript para utilizar en la plantilla.
- **html**: Los archivos alojados en esta carpeta permiten sobrescribir la salida HTML que imprime el CMS de forma predeterminada en componentes y módulos. Otro ejemplo mas de override que explicare luego.
- **language**: Contendrá archivos del idioma .ini.

Un ejemplo del archivo templateDetails.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE install PUBLIC "-//Joomla! 2.5//DTD template 1.0//EN"
"http://www.joomla.org/xml/dtd/2.5/template-install.dtd">

<extension version="2.5" type="template" method="upgrade">

     <name>templatename</name><!-- name the template -->
     <creationDate>xxxx-xx-xx</creationDate><!-- the date your are beginn to
code -->
     <author>your name</author>
     <copyright>Copyright © xxxx example.com</copyright>
     <authorEmail>your.name@example.com</authorEmail
     <authorUrl>http://www.example.com</authorUrl>
     <version>1.0.0</version><!-- give your template a number -->
     <description><![CDATA[
          <h1>templatename</h1>
          <p><img src="../templates/templatename/template_preview.png" /></p>
```

```
            <h2>Module positions</h2>
            <ol>
                 <li>debug</li>
            </ol>
            <p>Created by <a href="http://www.example.com" target="_blank">your
name | example.com</a>.</p>
      ]]></description><!-- change the description as you like -->

      <files><!-- no need to change -->
            <folder>css</folder>
            <folder>html</folder>
            <folder>images</folder>
            <folder>js</folder>
            <filename>apple-touch-icon-57x57.png</filename>
            <filename>apple-touch-icon-72x72.png</filename>
            <filename>apple-touch-icon-114x114.png</filename>
            <filename>component.php</filename>
            <filename>error.php</filename>
            <filename>favicon.ico</filename>
            <filename>index.php</filename>
            <filename>offline.php</filename>
            <filename>template_preview.png</filename>
            <filename>template_thumbnail.png</filename>
            <filename>templateDetails.xml</filename>
      </files>

      <languages folder="language"><!-- no need to change -->
            <language tag="de-DE">de-DE.tpl_templatename.ini</language>
            <language tag="en-GB">en-GB.tpl_templatename.ini</language>
      </languages>

      <positions><!-- define the positions for modules here -->
            <position>debug</position>
      </positions>
</extension>
```

Este archivo es similar al de cualquier extensión de Joomla, como hemos visto en el ejemplo de la creación del componente.

6.3 Override.

Template override es una técnica para redefinir la forma de representación en pantalla de un componente o módulo de Joomla. Esta funcionalidad se incluyó a partir del Joomla 1.5. El objetivo principal de esta técnica es permitir actualizar Joomla sin tener que preocuparse de que se vean afectados los cambios realizados previamente para personalizar el portal.

Así puedes de una manera relativamente fácil cambiar el aspecto de un componente o módulo sin modificar su propio código, con añadir a la carpeta de tu plantilla “html”, la carpeta del componente o módulo que quieres redefinir “com_X ó Mod_X” Joomla al leer esa carpeta ya tiene en cuenta qué estructura tiene que representar de ese componente o módulo.

6.4 Plantilla Joomla 2.5.

Una vez definidas la estructura visual empezamos con el diseño CSS. Primero, haremos un reset en las CSS con una hoja de estilo preparada, que reseteará todos los estilos de los navegadores, ya que hay navegadores que ya tienen sus propias hojas de estilo internas, y así sabemos que empezamos desde el mismo sitio para todos los navegadores.

Después de este punto ya estamos listos para diseñar nuestras hojas de estilo. Empezaremos por una hoja general, donde especificaremos el aspecto global.

A partir de aquí crearemos nuestra plantilla sin ningún tipo de estilo (Figura 4) utilizando las herramientas para desarrolladores de Google Chrome, así podremos ir viendo como Joomla estructura los menús y demás componentes e ir escribiendo nuestro código directamente en el editor. De esta manera, poco a poco vamos construyendo el aspecto de la web. Una vez escrito el código nos reunimos con el cliente (Trajesdeluces) para ir concretando todo el diseño.

Figura 6 Plantilla en Blanco

6.5 Plantilla Virtuemart 2.

Virtuemart viene con una plantilla básica instalada, pero en nuestro caso no se adapta con el diseño de la web, así que trasladaremos el archivo vm.css a nuestra carpeta de nuestra plantilla (*override*) para modificar los colores de la web.

En las próximas figuras (Figura 5, Figura 6, Figura 7) podemos observar cómo hemos realizado varios *overrides* más de los módulos de las categorías, la visión de las mismas, y la modificación de la vista de cada producto.

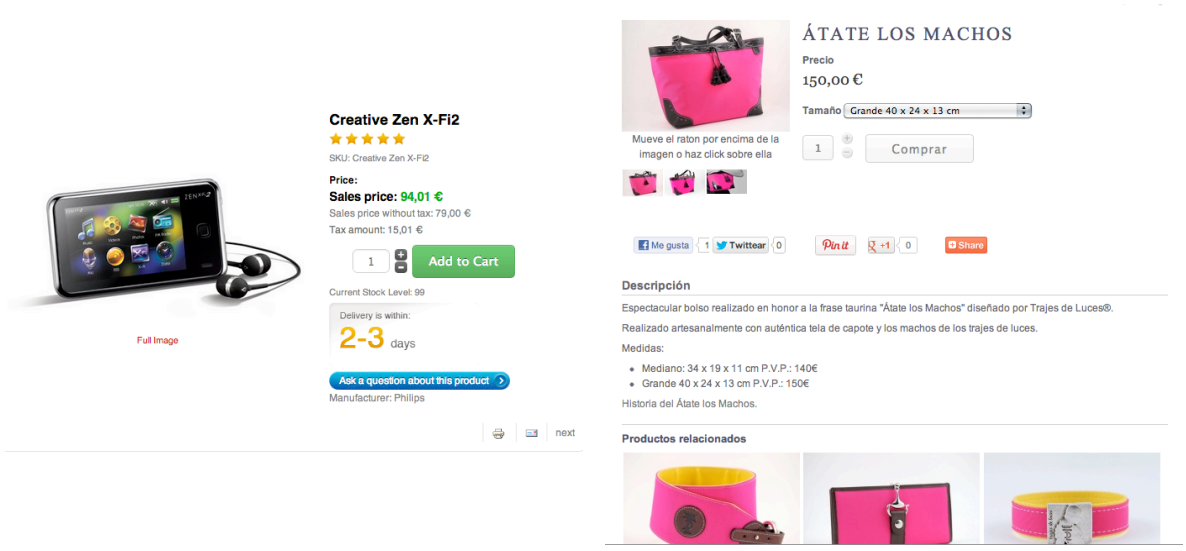


Figura 7. Cambio entre la pagina de detalles producto

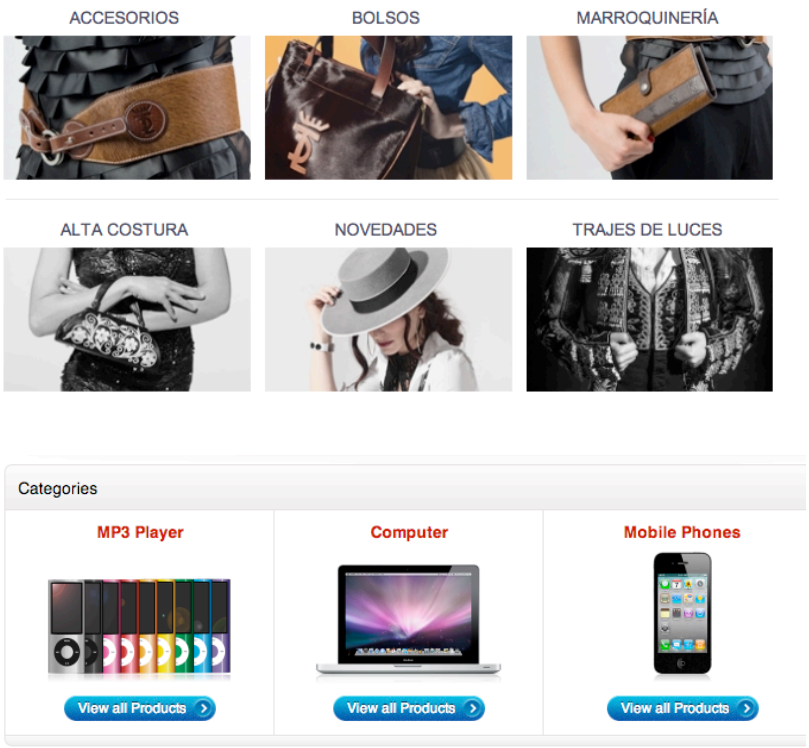


Figura 8. Cambio entre la pagina de categorías

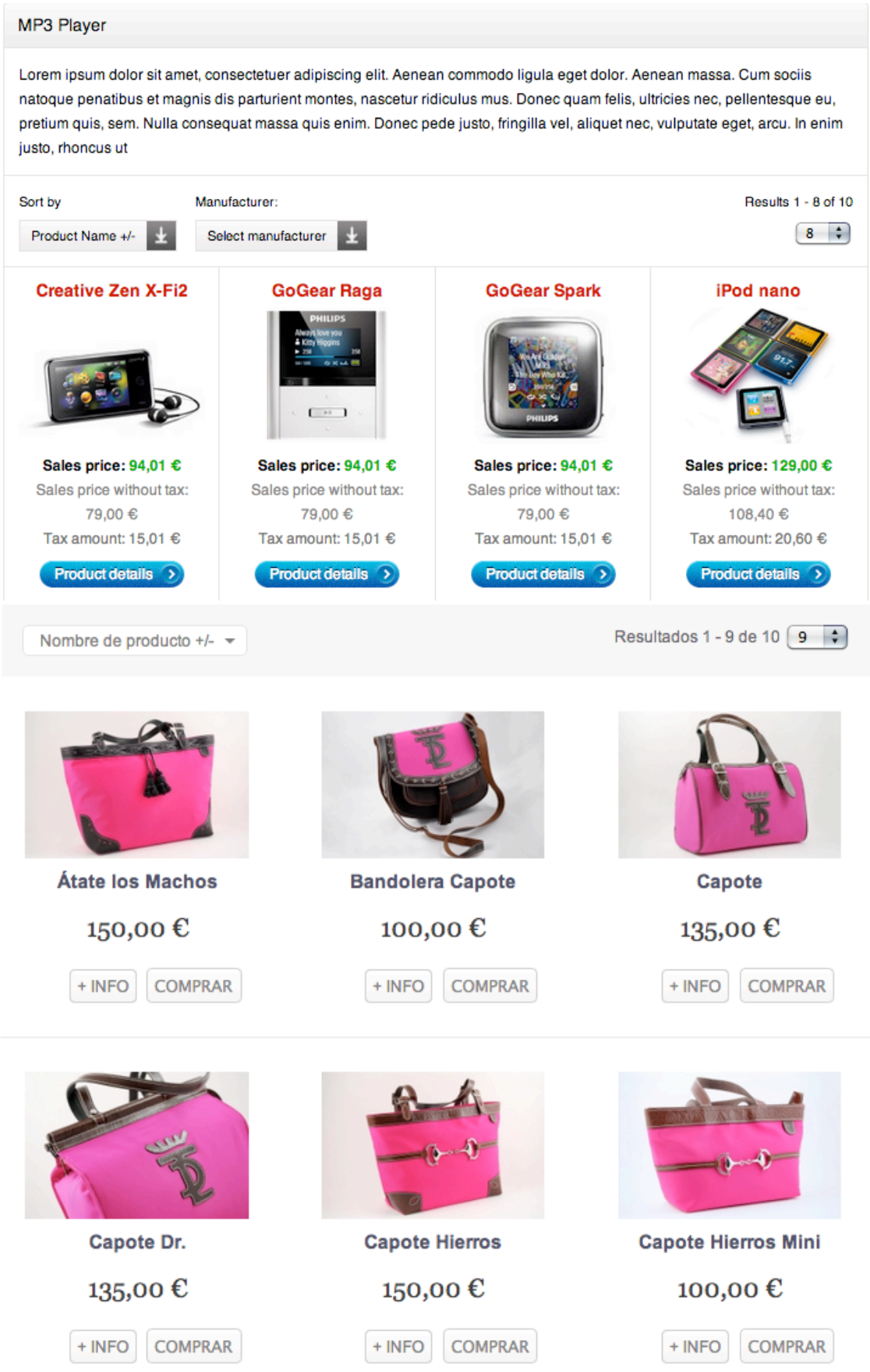


Figura 9. Detalle de lista de articulos

7. DESARROLLO DEL PLUGIN.

Debido a las especificaciones del trabajo a realizar, era necesaria la implementación de un plugin que realice la función de iniciar sesión al usuario con el e-mail o el nombre de usuario.

En este capítulo se presenta el desarrollo de ese Plugin para Joomla, viendo cual es su estructura y el desarrollo del mismo.

7.1 Estructura del Plugin.

El plugin lo hemos llamado *demail*, y aquí esta su estructura de archivos:

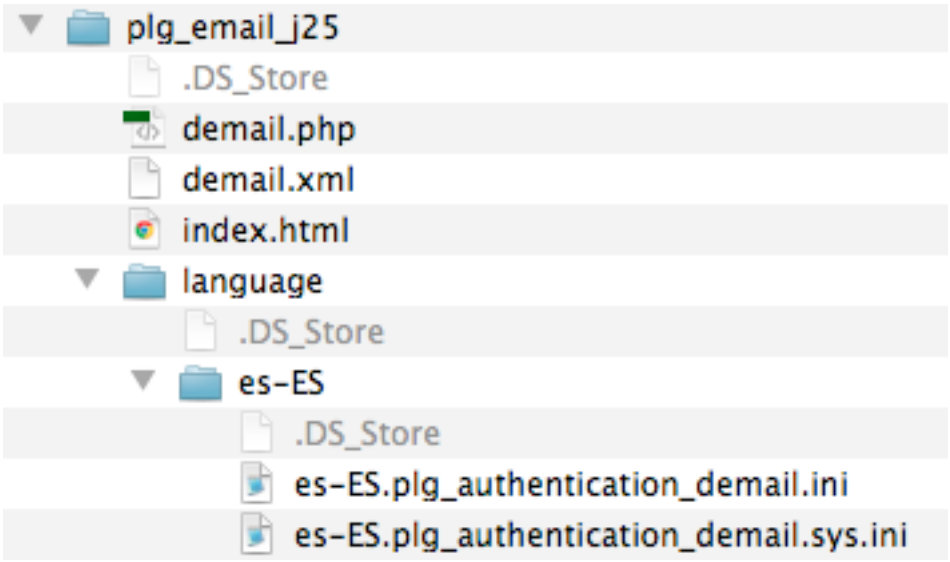


Figura 10. Estructura archivos del plugin

7.2 Descripción de Funciones.

El siguiente archivo demail.xml contiene la información del plugin y la manera de instalarlo, como en todas las extensiones Joomla.

El archivo demail.php es el núcleo de nuestro plugin:

```
function onUserAuthenticate($credentials, $options, &$response)
{
	jimport('joomla.user.helper');

	$response->type = 'Joomla';
	// Joomla does not like blank passwords
	if (empty($credentials['password'])) {
		$response->status = JAUTHENTICATE_STATUS_FAILURE;
		$response->error_message =
JText::_('JGLOBAL_AUTH_EMPTY_PASS_NOT_ALLOWED');
		return false;
	}
```

```
		$conditions='';

		// Lee base de datos
		$db		= JFactory::getDbo();
		$query	= $db->getQuery(true);

		$query->select('id, password');
		$query->from('#__users');
		$query->where('email=' . $db->Quote($credentials['username']));

		$db->setQuery($query);
		$result = $db->loadObject();

			$dbb = JFactory::getDbo();
			$queryy	= $dbb->getQuery(true);

			$queryy->select('id, password');
			$queryy->from('#__users');
			$queryy->where('username=' . $dbb->Quote($credentials['username']));

			$dbb->setQuery($queryy);
			$resultu = $dbb->loadObject();




		if ($result) {
			$parts	= explode(':', $result->password);
			$crypt	= $parts[0];
			$salt= @$parts[1];
			$testcrypt = JUserHelper::getCryptedPassword($credentials['password'], $salt);

			if ($crypt == $testcrypt) {
				$user = JUser::getInstance($result->id); //Recoge el usuario
                                $response->username = $user->username;
				$response->email = $user->email;
				$response->fullname = $user->name;
				if (JFactory::getApplication()->isAdmin()) {
					$response->language = $user->getParam('admin_language');
				}
				else {
					$response->language = $user->getParam('language');
				}
				$response->status = JAUTHENTICATE_STATUS_SUCCESS;
				$response->error_message = '';
			} else {
				$response->status = JAUTHENTICATE_STATUS_FAILURE;
				$response->error_message = JText::_('JGLOBAL_AUTH_INVALID_PASS');
			}

		} elseif($resultu) {
```

```
                    $parts    = explode(':', $result->password);
                    $crypt    = $parts[0];
                    $salt= @$parts[1];
                    $testcrypt =
JUserHelper::getCryptedPassword($credentials['password'], $salt);

                    if ($crypt == $testcrypt) {
                    $user = JUser::getInstance($result->id); //Recoge el
usuario
                              $response->username = $user->username;
                    $response->email = $user->email;
                    $response->fullname = $user->name;
                    if (JFactory::getApplication()->isAdmin()) {
                         $response->language = $user-
>getParam('admin_language');
                         }
                         else {
                              $response->language = $user-
>getParam('language');
                         }
                    $response->status = JAUTHENTICATE_STATUS_SUCCESS;
                    $response->error_message = '';
                    } else {
                         $response->status = JAUTHENTICATE_STATUS_FAILURE;
                         $response->error_message =
JText::_('JGLOBAL_AUTH_INVALID_PASS');
                    }


          }else{
                    $response->status = JAUTHENTICATE_STATUS_FAILURE;
                    $response->error_message =
JText::_('JGLOBAL_AUTH_NO_USER');
          }

     }
}
```

El funcionamiento del plugin es bastante intuitivo, como cualquier archivo php de Joomla incluye:

```
defined('_JEXEC') or die;
```

Esta sentencia evita que el código sea utilizado fuera de Joomla.

```
jimport('joomla.plugin.plugin');
class plgAuthenticationDemail extends JPlugin
```

Con estas 2 sentencias, Joomla importa la librería de plugins y define una clase que deriva del objeto genérico JPlugin.

```
function onUserAuthenticate($credentials, $options, &$response)
```

Ahora definimos la función heredada de JPlugin que es llamada cuando se va a autenticar un usuario.

Después de esto miramos que el password no esté en blanco, importamos una librería con funciones de autentificación, leemos la base de datos y

comprobamos lo que el usuario a introducido en el formulario, si es su usuario o un e-mail válido.

Una vez hecho esto, en cada caso comprobamos que le password coincida. Si no coincide, devuelve Error, y si coincide, devuelve que es válido. Nuestro plugin sólo consiste en esta lógica, Joomla ya conoce a través de definir la función "onUserAuthenticate" definida dentro de JPlugin cuando llamará a nuestra función para autentificar al usuario.

Los otros archivos son los relativos al idioma. En este caso nuestro plugin no requiere una respuesta especifica, por eso, estos archivos sólo contienen la repuesta del componente a la instalación por parte de Joomla:

```
; $Id: es-ES.plg_authentication_demail.ini
; Copyright (C) 2012 danibram. All rights reserved.
PLG_AUTH_EMAIL_XML_DESCRIPTION="Usa el email en vez del username"
PLG_AUTHENTICATION_EMAIL="Authentication - Danibram - email"
```

8. DESARROLLO DEL COMPONENTE.

En este capítulo se describe tanto la estructura física como la lógica con el contenido y las funcionalidades implementadas en cada uno de los diferentes ficheros que comprende el componente.

8.1 Estructura del Componente.

El componente principal lo hemos llamado *ecalendar* y comprende estos archivos y carpetas. Lo separaremos en dos por mejor organización: a la izquierda, los correspondientes al Back-End y a la derecha, los correspondientes al Front-End (Figura 11):

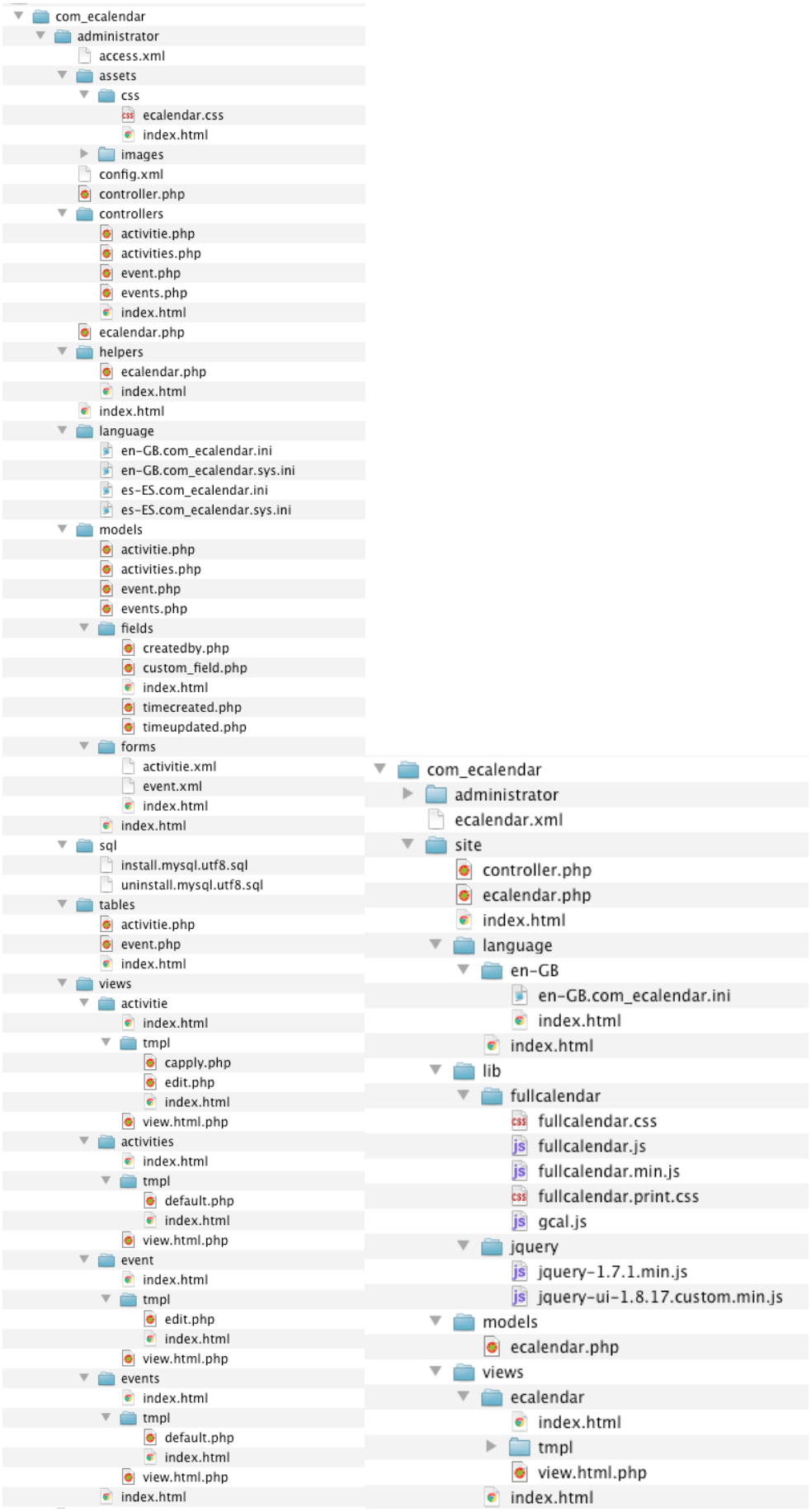


Figura 11. Estructura de archivos de un componente.

Separaremos a partir de ahora las dos partes para tener más claros los conceptos.

8.2 Descripción de Funciones del Front-End.

El primer módulo que nos encontramos dentro del Front-End es "ecalendar.php" que es el punto de entrada del componente:

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

// import joomla controller library
jimport('joomla.application.component.controller');

// Get an instance of the controller prefixed by ecalendar
$controller = JController::getInstance('ecalendar');

// Perform the Request task
$controller->execute(JRequest::getCmd('task'));

// Redirect if set by the controller
$controller->redirect();
?>
```

En el punto de entrada definimos el controlador. En este caso, si no especificamos el archivo, éste será "controller.php" y lamará a todas las funciones de la parte del Front-End de nuestro componente.

8.2.1 Controladores.

En el caso específico de nuestro componente y en la parte del Front-End el archivo "controller.php" contiene:

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

// import Joomla controller library
jimport('joomla.application.component.controller');

/**
 * ecalendar Component Controller
 */
class ecalendarController extends JController
{
}
```

```
?>
```

Cuando no definimos una o varias tareas en el controlador, Joomla ejecuta la tarea por defecto, que es mostrar la vista por defecto para nuestro componente "ecalendar".

8.2.2 Modelos.

El modelo ecalendar.php incorpora dos funciones:

- **getWeekstart**: Esta función es la encargada de acceder a la configuración del componente para devolver el día que empieza la semana.

```
public function getWeekStart()
{
	if (!isset($this->weekstart))
	{
		$this->weekstart =JRequest::getInt('weekstart');

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

- **getEvents**: Esta función accede a la base de datos para saber que eventos están definidos y a qué actividad se refieren estos eventos. Devuelve una lista con toda la información recogida.

```
public function getEvents()
{
	/*$dias = array(L,M,X,J,V,S,D);
	$fecha = $dias[date(N, strtotime(2008-02-25))];*/



	$db = JFactory::getDBO();
	$query = $db->getQuery(true);
	$query = "
	SELECT *
	FROM ".$db->nameQuote('#__ecalendar_events')."
	";
	$db->setQuery((string)$query);
	$event_list = $db->loadObjectList();

	$list = array();
	if ($event_list)
	{
		$ultimoel=count($event_list);
		$i=1;
		$list="[";
		foreach($event_list as $event_one)
```

```
                {
                        $db = JFactory::getDBO();
                        $query = $db->getQuery(true);
                        $query = "
                                SELECT ".$db-
>nameQuote('#__ecalendar_activities.name')."
                        FROM ".$db-
>nameQuote('#__ecalendar_activ_event')."
                        INNER JOIN ".$db-
>nameQuote('#__ecalendar_activities')."
                        ON ".$db-
>nameQuote('#__ecalendar_activ_event.id_activitie')." = ".$db-
>nameQuote('#__ecalendar_activities.id')."
                        WHERE ".$db-
>nameQuote('#__ecalendar_activ_event.id_event')." = ".$event_one-
>id.";
                                ";

                        $db->setQuery($query);
                        $title = $db->loadAssoc();

                        //echo date(U, strtotime($event_one->start));
                        //echo date(U, strtotime($event_one-
>end));die;

                        $list .=    "{ \n";
                        $list .=    "id : ".$event_one->id.",\n";
                        $list .=    "title : '".$title[name]."',\n";
                        $list .=    "start : '".date(U,
strtotime($event_one->start))."',\n";
                        if ($event->end){
                        $list .=    "end : '".date(U,
strtotime($event_one->end))."',\n";
                        }
                        if ($event->allday==1){
                        $list .=    "allDay: true \n";
                        }else{
                        $list .=    "allDay: false \n";
                        }

                        if($i==$ultimoel)
                        {
                                $list .=    "}\n";
                        }else{
                                $list .=    "},\n";
                        }
                        $i++;
                }
                $list.="]";
        }
        return $list;
    }
}
?>
```

8.2.3 Vistas.

La vista del ecalendar (en el Front-End) se encarga de mostrar el calendario, puesto que el calendario requiere de la librería jQuery para funcionar. Al cargar estas librerías necesitamos utilizar esta función:

```
$document->addScriptDeclaration('jQuery.noConflict();');
```

Con esta función hacemos que jQuery no interfiera con moontools (librería similar utilizada en el código fuente de Joomla). Se cargan las librerías jQuery y el código Javascript del calendario. Una vez hecho esto, se le pide al modelo que le pase los datos con la siguiente instrucción:

```
$this->get('Events')
$this->get('WeekStart')
```

A partir de aquí se carga el código del calendario para que aparezca en la pantalla.

8.3 Descripción de Funciones del Back-End.

Como en el caso anterior, lo primero que nos encontramos es el punto del entrada al Back-End del componente. Lo único que cambia con respecto al Front-End, es que se añaden unas líneas para restringir el acceso a nuestra aplicación a ciertos usuarios.

```
// Access check
if (!JFactory::getUser()->authorise('core.manage', 'com_ecalendar')) {
     return JError::raiseWarning(404, JText::_('JERROR_ALERTNOAUTHOR'));
}
```

Luego carga las librerías de Joomla y da pasa al controlador.

8.3.1 Controladores.

En el controlador tenemos que tener en cuenta que tendremos varias vistas definidas y debemos de elegir cual es la que por defecto cargará. Además, hemos añadido un menú para que Joomla lo tenga en cuenta. Esto lo realizamos creando un archivo en la carpeta “helpers” donde irá el código de creación del menú, de tal manera que el controlador queda más limpio y estructurado.

```
     public function display($cachable = false)
     {
          require_once JPATH_COMPONENT.'/helpers/ecalendar.php';

          // Load the submenu.
          EcalendarHelper::addSubmenu();
```

```
        // set default view if not set
        JRequest::setVar('view', JRequest::getCmd('view', 'activities'));

        // call parent behavior
        parent::display($cachable);

    }
```

El archivo *helper* contiene como hemos apuntado anteriormente el código para la creación del submenú:

```
public static function addSubmenu($vName = '')
    {
        JSubMenuHelper::addEntry(
            JText::_('COM_ECALENDAR_TITLE_ACTIVITIES'),
            'index.php?option=com_ecalendar&view=activities',
            $vName == 'activities'
        );
        JSubMenuHelper::addEntry(
            JText::_('COM_ECALENDAR_TITLE_EVENTS'),
            'index.php?option=com_ecalendar&view=events',
            $vName == 'events'
        );

    }
```

8.3.2 Modelos.

A partir de aquí tenemos que tener en cuenta que hemos implementado la gestión de actividades y luego la posibilidad de que se pueda modificar los eventos uno por uno. Por tanto, tenemos 4 controladores asociados a 4 vistas, separadas de 2 en 2, y cada pareja maneja y muestra los datos tanto de todas las actividades/eventos definidos como individualmente los datos de una en concreto. Es por esto que tenemos 2 vistas para actividades y 2 para los eventos.

Empecemos con las actividades. Tenemos 2 modelos: activities y activitie. En activities el modelo accede a la base de datos para obtener la lista de actividades definidas.

Al integrar la gestión de permisos de Joomla en la aplicación a la hora de la lectura de los datos, se tiene en cuenta el usuario que está pidiendo esos datos, y restringe el acceso. También hay que tener en cuenta si la actividad está publicada o no para mostrarla u ocultarla. Además, Joomla cuenta con un sistema de búsqueda y también hemos añadido el código para que funcione.

```
// Select the required fields from the table.
        $query->select(
            $this->getState(
```

```
                    'list.select',
                    'a.*'
                )
            );
            $query->from('`#__ecalendar_activities` AS a');


             // Join over the users for the checked out user.
              $query->select('uc.name AS editor');
              $query->join('LEFT', '#__users AS uc ON uc.id=a.checked_out');
             // Filter by published state
              $published = $this->getState('filter.state');
              if (is_numeric($published)) {
                  $query->where('a.state = '.(int) $published);
              } else if ($published === '') {
                  $query->where('(a.state IN (0, 1))');
              }
            // Filter by search in title
            $search = $this->getState('filter.search');
            if (!empty($search)) {
                 if (stripos($search, 'id:') === 0) {
                      $query->where('a.id = '.(int) substr($search, 3));
                 } else {
                      $search = $db->Quote('%'.$db->getEscaped($search,
true).'%');
                  $query->where('( a.name LIKE '.$search.' )');
                 }
            }
              // Add the list ordering clause.
          $orderCol    = $this->state->get('list.ordering');
          $orderDirn   = $this->state->get('list.direction');
          if ($orderCol && $orderDirn) {
              $query->order($db->getEscaped($orderCol.' '.$orderDirn));
          }

            return $query;
```

Por último, devolvemos la lista de datos.

Para el modelo de activities , la parte de lectura y guardado de los datos de la actividad la realiza Joomla automáticamente si tienes los campos bien definidos en la carpeta forms. La parte compleja es la creación de los eventos, dada una actividad definida en unos periodos de tiempo y en unos días de la semana en concreto, la función “create” nos creará los eventos para cada día en el que está definida la actividad dentro del periodo.

También se ha implementado una función para el cálculo del último día del mes, y para que borre los eventos de esa actividad. Añadimos a todo esto la gestión de los errores para saber si ha fallado.

```
      /**
       * Method to create de events.
       *
```

```
	 * @param   array  &$pks  The ids of the activities to create the
events.
	 *
	 * @return  boolean  True on success.
	 *
	 * @since   1.6
	 */
	public function create(&$pks)
	{

		$dw['Sunday'] = "D";
		$dw['Monday'] = "L";
		$dw['Tuesday'] = "M";
		$dw['Wednesday'] = "X";
		$dw['Thursday'] = "J";
		$dw['Friday'] = "V";
		$dw['Saturday'] = "S";

		$pks = (array) $pks;

		foreach ($pks as $pk)
		{


EcalendarModelactivitie::Borra_prev($pk);


// Get the DB object
				$db = $this->getDbo();
				$query = $db->getQuery(true);

				$query->select('*')
					->from($db->quoteName('#__ecalendar_activities'))
					->where($db->quoteName('id')." = ".(int) $pk);
				$db->setQuery($query);

				$event = $db->loadAssoc();


				$end=explode(" ", trim($event['end_date'], " "));
				//$dateend=explode("-", trim($end[0], "-"));
				//$ud_ultimomes=UltimoDia($dateend[0],$dateend[1]);

				$start=explode(" ", trim($event['start_date'], " "));
				//$datestart=explode("-", trim($start[0], "-"));
				//$ud_primermes=UltimoDia($datestart[0],$datestart[1]);
				$daysofweek=explode(",", trim($event['days'], ","));
				$fechaInicio=strtotime($start[0]);
				$fechaFin=strtotime($end[0]);
				foreach ($daysofweek as $dayofweek){
					for($i=$fechaInicio; $i<=$fechaFin; $i+=86400){
						if($dw[date('l', $i)]==$dayofweek){

							$day_week=date("Y-m-d",$i);

							$fecha_s=array($day_week,$start[1]);
							$fecha_e=array($day_week,$end[1]);

//echo implode(" ",$fecha_s)." ".implode(" ",$fecha_e)."<br/>";
```

```
                                $dbb = $this->getDbo();
                                $queryy = $dbb->getQuery(true);
                                $queryy='INSERT INTO #__ecalendar_events
(start,end,allday)' .
                                        ' VALUES ("'.implode("
",$fecha_s).'","'.implode(" ",$fecha_e).'","0")';
                                $dbb->setQuery($queryy);
// Check for database errors.
                                if (!$dbb->query())
                                     {
                                          $this->setErrbor($dbb-
>getErrorMsg());
                                          return false;
                                     }
                                $ultim_id=$db->insertid();
                                $dbb->clear();
//Relationships
                                $dbb = $this->getDbo();
                                $queryy = $dbb->getQuery(true);
                                $queryy='INSERT INTO #__ecalendar_activ_event
(id,id_activitie,id_event)' .
                                             ' VALUES
(0,"'.$pk.'","'.$ultim_id.'")';

                                $dbb->setQuery($queryy);
// Check for database errors.
                                if (!$dbb->query())
                                     {
                                          $this->setError($dbb-
>getErrorMsg());
                                          return false;
                                     }
                                $dbb->clear();
//     Fecha mod events update
                                $dbb = $this->getDbo();
                                $queryy = $dbb->getQuery(true);
                                $queryy='UPDATE `#__ecalendar_activities`'.
                               ' SET `mod_date_event` = "'.date("Y-m-d H:i:s").
                               '" WHERE (id='.$pk.')';
                                $dbb->setQuery($queryy);

                                if (!$dbb->query())
                                     {
                                          $this->setError($dbb-
>getErrorMsg());
                                          return false;
                                     }
                           }
                     }
                     }
               }
                     return true;
          }

}
```

Esta función es núcleo de la aplicación, puesto que los modelos para gestionar los eventos son más sencillos de implementar, ya que únicamente muestran o modifican los datos en la base de datos, no operan con ellos.

El modelo events es similar al de activities, pues da como resultado la lista de eventos. En este modelo no se ha tenido que implementar una función compleja para guardar los datos, ya que Joomla, como hemos explicado anteriormente, ya tiene en cuenta los campos que tiene leer y que ha de guardar. Estos campos para las actividades y eventos están definidos dentro de la carpeta forms en unos archivos xml.

```
<field name="name" type="text"
      label="COM_ECALENDAR_FORM_LBL_ACTIVITIE_NAME"
      description="COM_ECALENDAR_FORM_DESC_ACTIVITIE_NAME"
      required="true"
      filter="raw"
      size="40"
      maxlength="40" />
```

Éste es un ejemplo de cómo está definido el campo “name” para una actividad.

8.3.3 Vistas.

Para las vistas se ha de destacar que existen 4 vistas en función de la tarea asignada.

La vista activities contiene la lista de actividades definidas. La barra de arriba contiene las acciones que puedes hacer con la o las actividades seleccionadas. Además, se ha añadido el botón para crear los eventos a partir de la actividad seleccionada. En la lista de actividades hemos añadido una columna más que se llama actualizar, cuando modificas una actividad esa columna te avisa de que debes volver a crear los eventos. Cuando pulsas sobre la actividad seleccionada cambiamos a la vista activitie.

En esta vista tenemos los campos para crear una nueva actividad. Hemos añadido un campo deshabilitado que avisa cuándo se han creado los eventos y la actividad, para poder avisar al usuario de que debe crear los eventos de nuevo. Por otro lado, para la selección de la fecha aparece un pequeño calendario para facilitar la selección.

La vista events muestra una lista de los eventos definidos, similar a la vista activities. Al editar un evento podemos mover una clase de día si lo necesitamos. Faltaría implementar que pudieras crear una clase.

En la vista event puedes hacer la edición del evento y guardarlo.

8.4 Instalación.

La aplicación utiliza 3 tablas:

- #__ecalendar_activities:Contiene la información de la actividad, utiliza también variable auxiliares de joomla.
- #__ecalendar_events: Contiene la información de la actividad, utiliza tambien variables auxiliares de joomla.
- #__ecalendar_activ_event: tabla auxiliar que relacióna los eventos con la actividad definida, no utiliza variables del sistema de Joomla.

Las variables auxiliares de Joomla son: orden, estado de publicación, si está checkeada y la fecha de creación. Éstas las utiliza Joomla por defecto y luego nosotros hemos añadido los campos necesarios.

```
CREATE TABLE IF NOT EXISTS `#__ecalendar_activities` (
`id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,
`ordering` INT(11)  NOT NULL ,
`state` TINYINT(1)  NOT NULL DEFAULT '1',
`checked_out` INT(11)  NOT NULL ,
`checked_out_time` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
`name` VARCHAR(255)  NOT NULL ,
`start_date` DATETIME NOT NULL ,
`end_date` DATETIME NOT NULL ,
`days` VARCHAR(255)  NOT NULL ,
`mod_date_activ` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
`mod_date_event` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
PRIMARY KEY (`id`)
) DEFAULT COLLATE=utf8_general_ci;

CREATE TABLE IF NOT EXISTS `#__ecalendar_events` (
`id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,
`ordering` INT(11)  NOT NULL ,
`state` TINYINT(1)  NOT NULL DEFAULT '1',
`checked_out` INT(11)  NOT NULL ,
`checked_out_time` DATETIME NOT NULL DEFAULT '0000-00-00 00:00:00',
`start` DATETIME  NOT NULL DEFAULT '0000-00-00 00:00:00',
`end` DATETIME  NOT NULL DEFAULT '0000-00-00 00:00:00',
`allday` BOOLEAN NOT NULL ,
PRIMARY KEY (`id`)
) DEFAULT COLLATE=utf8_general_ci;

CREATE TABLE IF NOT EXISTS `#__ecalendar_activ_event` (
`id` int(11) UNSIGNED NOT NULL AUTO_INCREMENT,
`id_activitie` INT(11)  NOT NULL ,
`id_event` INT(11)  NOT NULL ,
PRIMARY KEY (`id`)
) DEFAULT COLLATE=utf8_general_ci;
```

Estos son los ficheros de instalación MySQL que se almacenan en la carpeta “sql”. Se utilizan para que Joomla cree las tablas a la hora de instalar y a la hora de desinstalar que no se vea afectado el sistema, ni se queden restos en la base de datos.

9. Migración.

9.1 Consideraciones.

Hay que considerar que la web lleva varios años en activo, por lo tanto, suponemos que contiene bastante información tanto en artículos como en imágenes.

9.2 Migración Joomla 1.5 a 2.5.

El proceso de migración de Joomla 1.5 a Joomla 2.5 puede realizarse empleando componentes de terceros o de manera manual. Para el primer caso, emplearemos dos extensiones: J2XML y Jupgrade, cada una de las cuales precisa de un procedimiento específico para realizar la migración.

En el caso de llevar a cabo la migración de manera manual, debemos tener en cuenta que los esquemas de las tablas de Joomla 1.5 son diferentes a los de Joomla 2.5, por lo que habrá que adaptarlos para que la importación de datos se efectúe de forma correcta. Este proceso no es sencillo y se puede complicar si tenemos que migrar componentes, plugins o módulos de terceros.

Además, tenemos que tener presente que la migración es un procedimiento de alto riesgo y puede fallar o provocar errores, por lo que es imprescindible hacer una copia de seguridad de nuestro sitio antes de lanzarnos a pasar a la nueva versión.

Una de las extensiones que podemos emplear para llevar a cabo el proceso de migración es J2XML. Este componente nos permitirá exportar nuestros artículos, secciones, categorías, y usuarios, e importarlas posteriormente en la versión 2.5 de Joomla.

Podemos descargar la extensión desde su página oficial e instalarla siguiendo los pasos habituales. Una vez instalada, iremos al gestor de plugins y activaremos el plugin System – J2XML (Figura 12).

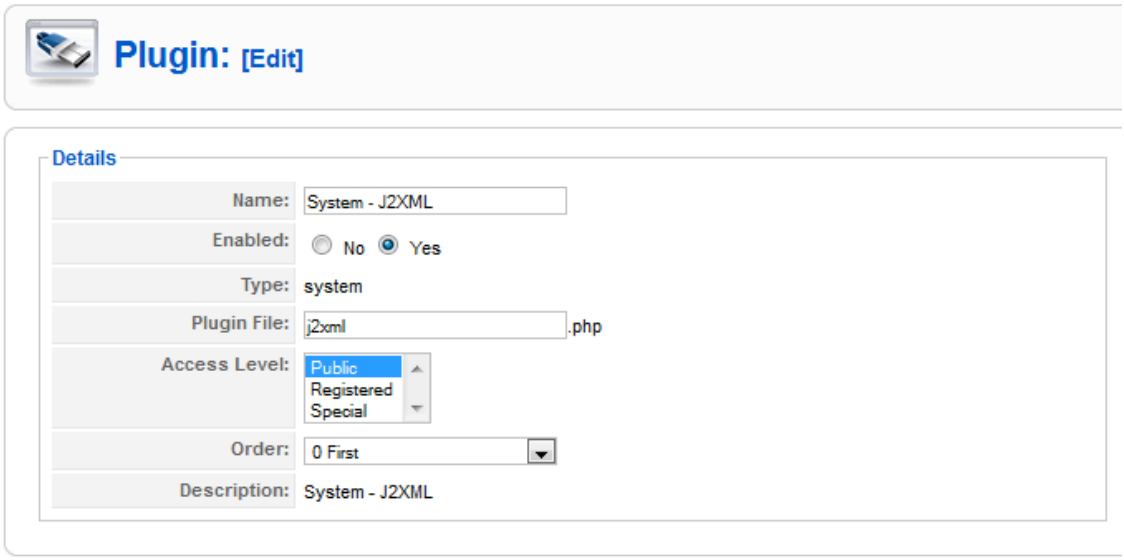


Figura 12. *Activar el plugin System – J2XML*

Ahora nos dirigiremos a "Componentes > J2XML" donde podremos establecer algunas opciones para la exportación. Decidiremos si se exportarán las imágenes de los artículos, los usuarios o si se comprimirán los archivos (Figura 13).

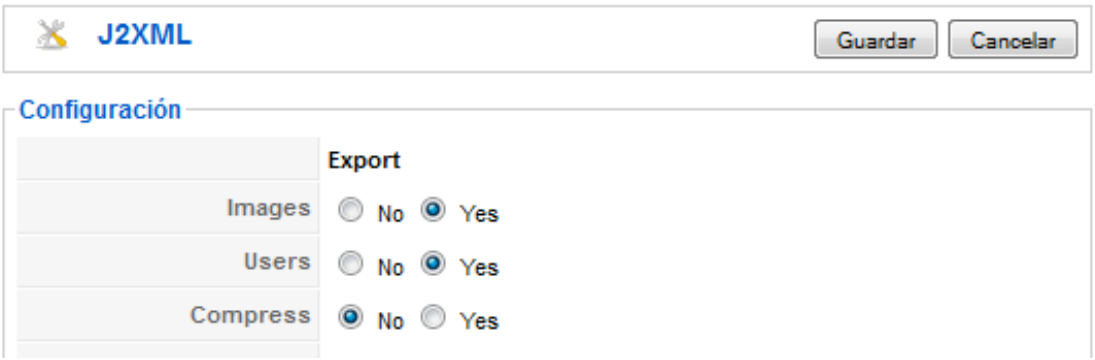


Figura 13. C*onfiguración de J2XML*

Figura 14. Para dar comienzo al proceso, pulsaremos en "Article Manager".

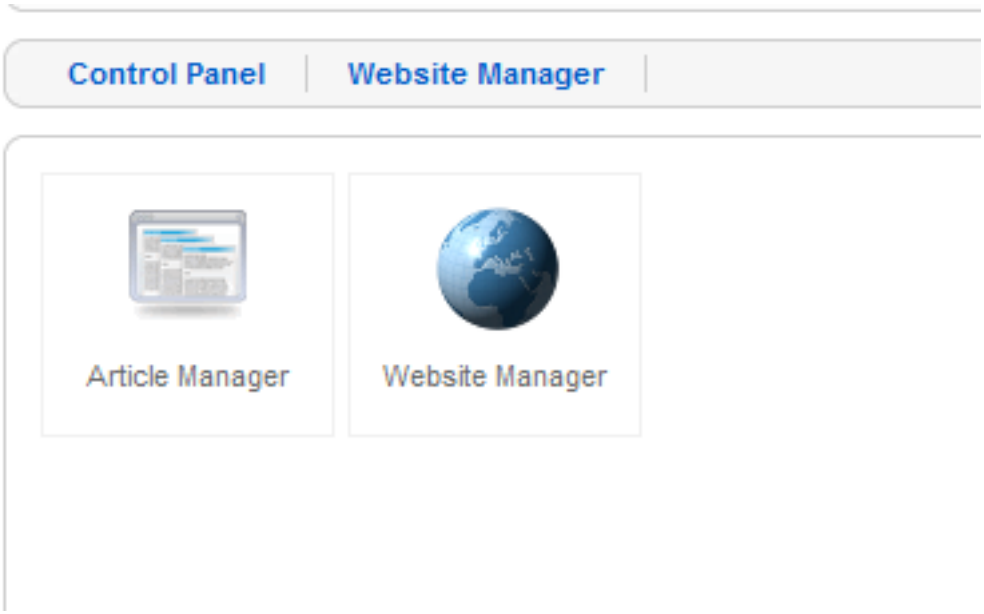


Figura 14. *Panel de control de J2XML*

Accederemos al gestor de artículos de Joomla y veremos que tiene dos nuevos iconos: uno para exportar los artículos y otro para enviarlos directamente a otro sitio web (Figura 15).

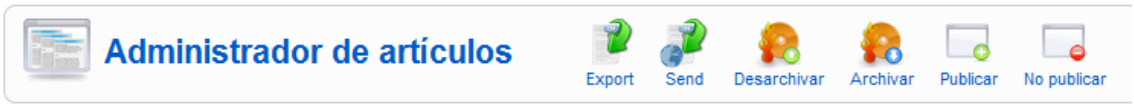


Figura 15. Botones "Export" y "Send"

Para realizar la migración tendremos que marcar todos los artículos que queramos añadir a nuestro nuevo Joomla y pulsar en exportar (Figura 16).

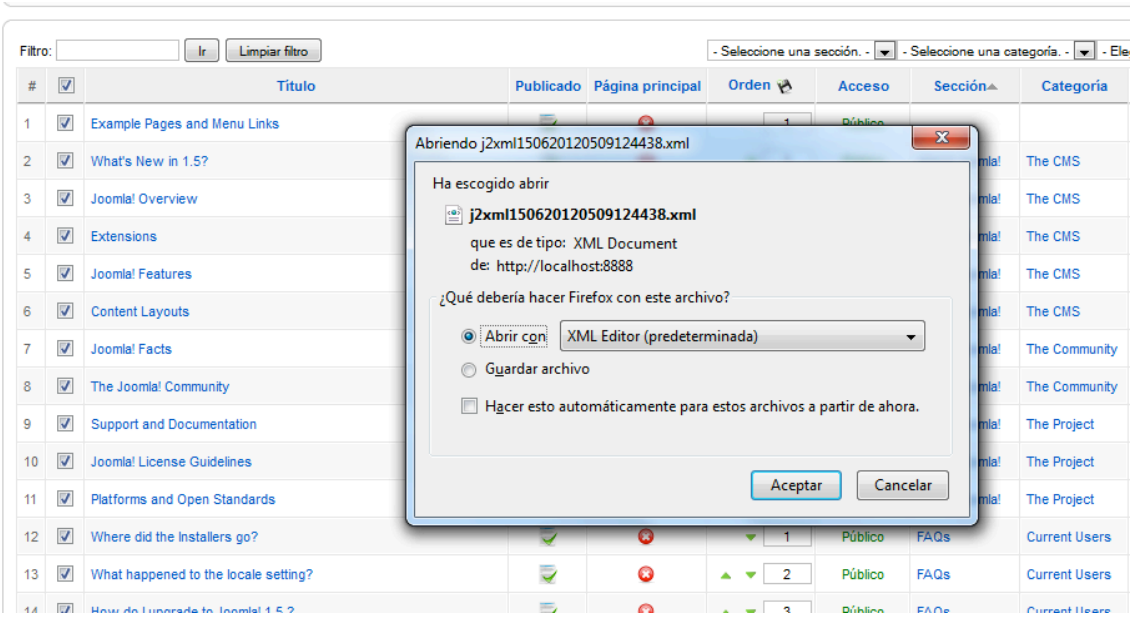


Figura 16. Exportación de los datos

De esta forma, se generará un archivo Xml que contendrá toda la información de nuestro sitio y que podremos importar más tarde (Figura 17).

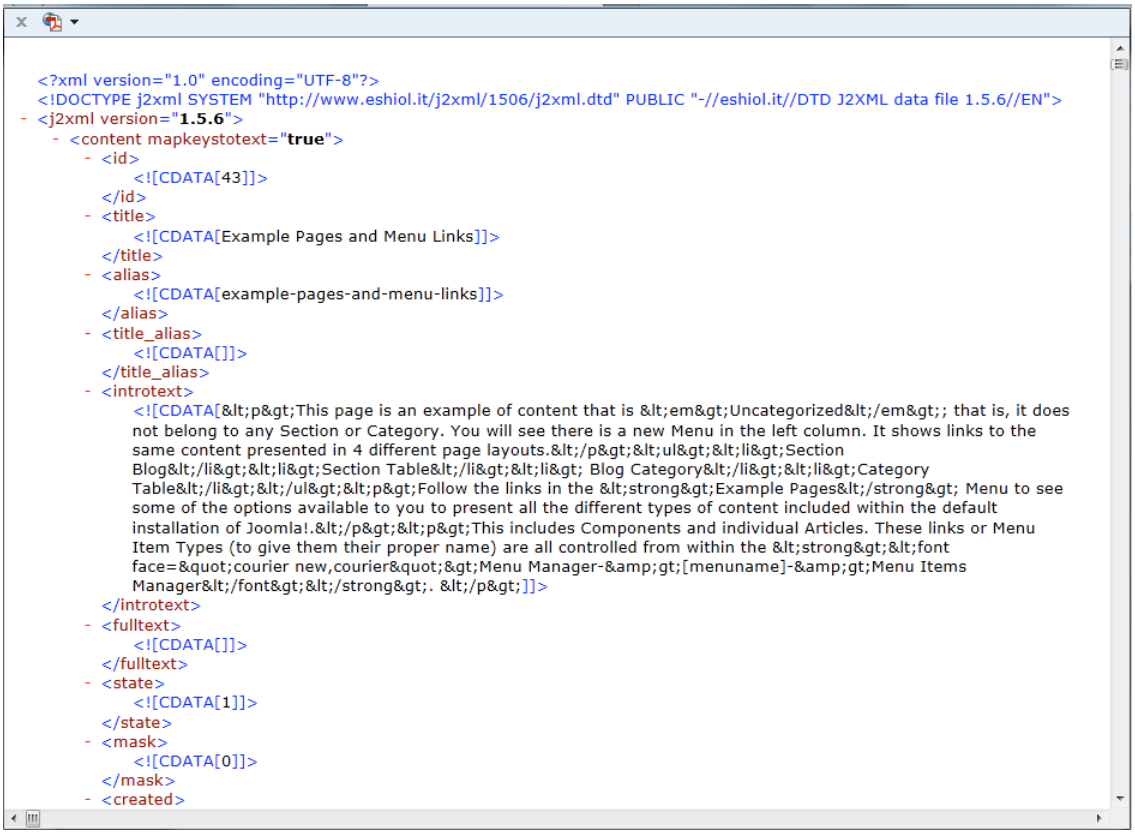


Figura 17. Contenido del archivo Xml generado

Este archivo contendrá, además de los artículos, las secciones y categorías de nuestro sitio. Si hay algún otro contenido que queremos exportar (usuarios o enlaces web), podemos acceder al gestor correspondiente para realizar la exportación (Figura 18).

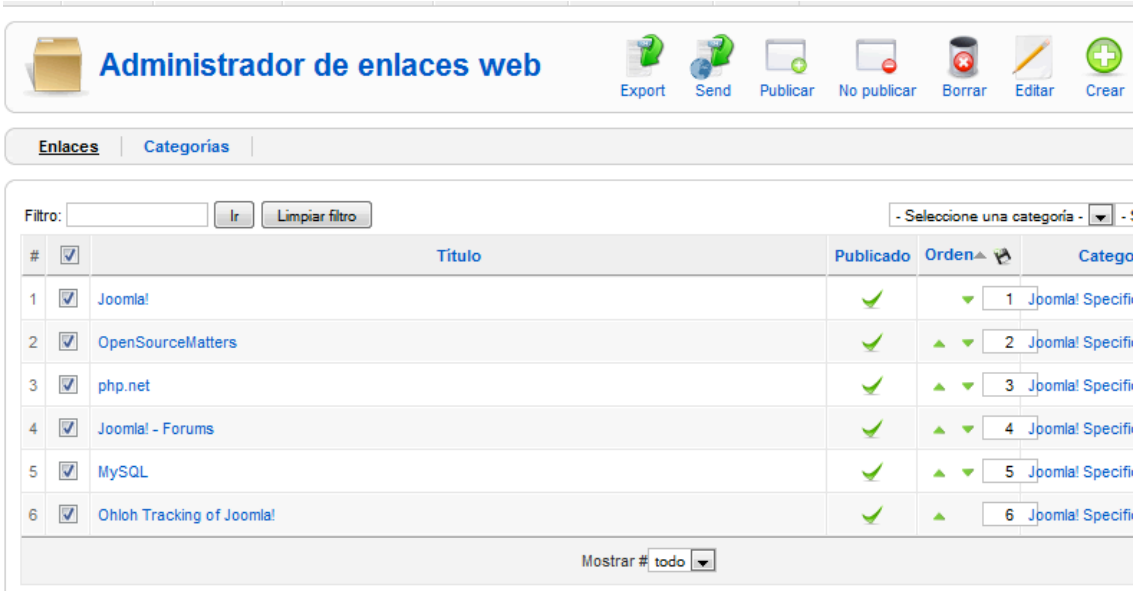


Figura 18. Botones "Export" y "Send" en el administador de enlaces web

Después de exportar todos los contenidos al formato Xml, tendremos que descargar e instalar, en la nueva versión de Joomla, el componente J2XML importer, que nos permitirá importar los archivos generados.

Tras la instalación, nos dirigiremos a "Componentes > J2XML Importer", donde podremos configurar las opciones de importación pulsando en "Opciones". Decidiremos qué contenidos se importarán y si se sobrescribirán, si se mantendrán los niveles de acceso originales, el autor, la categoría, etc. (Figura 19)

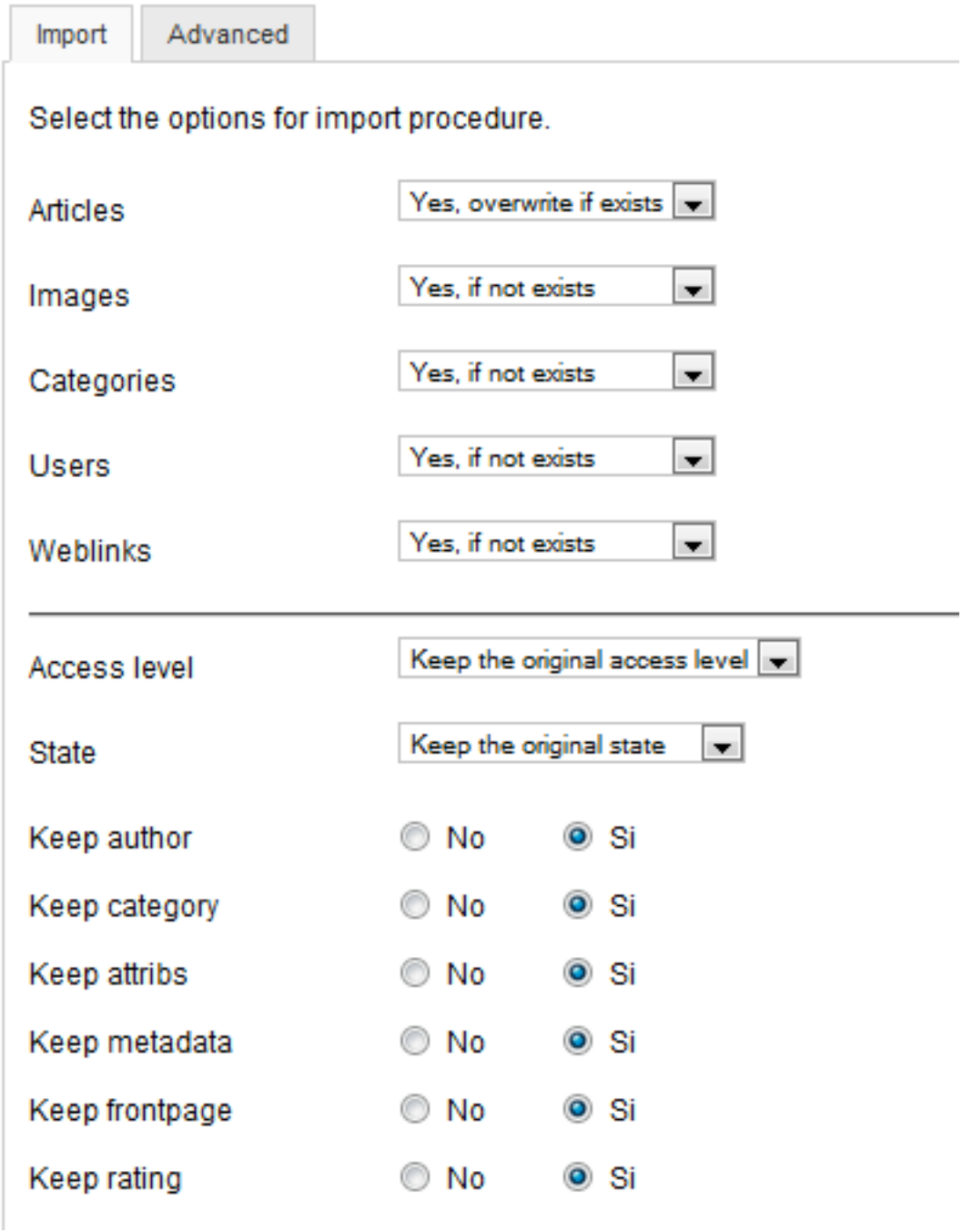


Figura 19. Configuración de J2XML Importer

Una vez establecidas las opciones, pulsaremos en examinar y, tras seleccionar el archivo adecuado, pulsaremos en "Importar." (Figura 20)

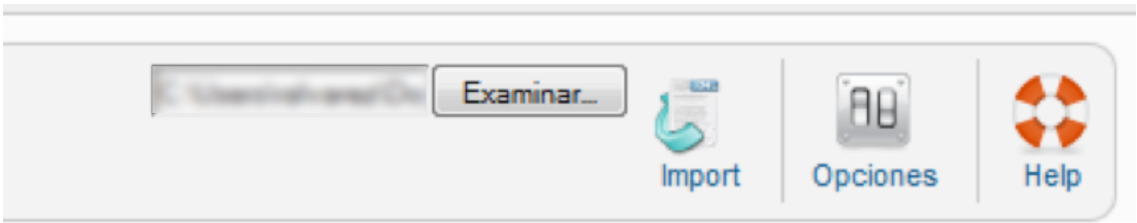


Figura 20. *Importación de los datos desde el archivo Xml*

Si todo ha ido bien, veremos una lista con los elementos que se han importado (Figura 21).

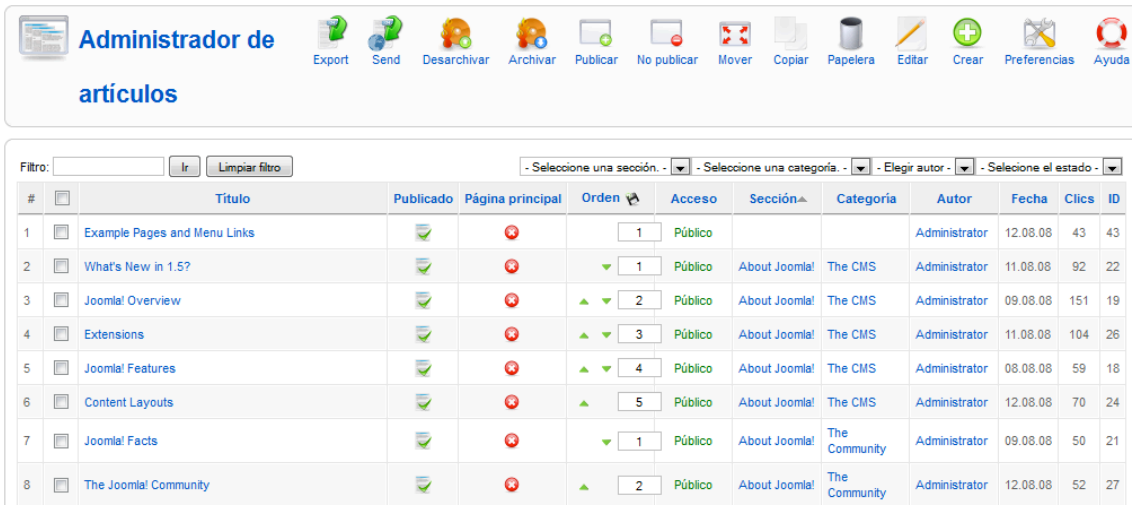
Figura 21. Lista de elementos importados

Si accedemos ahora al gestor de categorías veremos que ya tenemos las secciones y categorías ordenadas en nuestro nuevo sitio (Figura 22).

Sin categoría (Alias: uncategorised)	✓	1	Publico	Todo	2
About Joomla! (Alias: about-joomla)	✓	2	Publico	Todo	8
The CMS (Alias: the-cms)	✓	1	Publico	Todo	11
The Community (Alias: the-community)	✓	2	Publico	Todo	12
The Project (Alias: the-project)	✓	3	Publico	Todo	13
FAQs (Alias: faqs)	✓	3	Publico	Todo	9
Current Users (Alias: current-users)	✓	1	Publico	Todo	14
General (Alias: general)	✓	2	Publico	Todo	15
Languages (Alias: languages)	✓	3	Publico	Todo	16
New to Joomla! (Alias: new-to-joomla)	✓	4	Publico	Todo	17

Figura 22. *Categorías importadas a la nueva instalación de Joomla*

Realizaremos el mismo procedimiento en el gestor de artículos (Figura 23).



#	Título	Publicado	Página principal	Orden	Acceso	Sección	Categoría	Autor	Fecha	Clics	ID
1	Example Pages and Menu Links	✓	✗	1	Público			Administrator	12.08.08	43	43
2	What's New in 1.5?	✓	✗	1	Público	About Joomla!	The CMS	Administrator	11.08.08	92	22
3	Joomla! Overview	✓	✗	2	Público	About Joomla!	The CMS	Administrator	09.08.08	151	19
4	Extensions	✓	✗	3	Público	About Joomla!	The CMS	Administrator	11.08.08	104	26
5	Joomla! Features	✓	✗	4	Público	About Joomla!	The CMS	Administrator	08.08.08	59	18
6	Content Layouts	✓	✗	5	Público	About Joomla!	The CMS	Administrator	12.08.08	70	24
7	Joomla! Facts	✓	✗	1	Público	About Joomla!	The Community	Administrator	09.08.08	50	21
8	The Joomla! Community	✓	✗	2	Público	About Joomla!	The Community	Administrator	12.08.08	52	27

Figura 23. Artículos importados a la nueva versión

Este procedimiento fue sencillo en una instalación de prueba que realizamos, pero para este caso en concreto tuvimos que hacer una migración manual para asegurarnos que todo estaba en su sitio correcto.

Realizamos una instalación nueva de Joomla, pero a la hora de instalar nos aseguramos que el prefijo de las tablas de la base de datos era el que tenia Joomla 1.5, para poder comparar las tablas mas fácilmente.

A través de phpMyAdmin comparamos las dos bases de datos, con el exportador de phpMyAdmin exportamos las tablas que necesitamos y adaptamos los parámetros para que coincidieran con los de la antigua instalación. Luego, para cambiar los prefijos de la base de datos, hicimos una copia con un componente

que se llama akeeba backup y al restaurarlo cambiamos los prefijos. Debemos que resaltar que las instalaciones de Joomla 1.5 utilizan siempre el mismo prefijo “jan_”.

Una vez terminada la instalación procedemos a la modificación de la plantilla creada para adaptarla al diseño pedido.

9.3 Migración Virtuemart estándar

Para realizar esta migración, primero empeczamos probando con un componente llamado jUpgrade, con el cual migramos de un click la instalación de Joomla 1.5 a 2.5. El único inconveniente que tuvimos es que las 2 versiones de Joomla se instalan bajo la misma base de datos y esto fue un foco de problemas, aunque instalamos Virtuemart 2 y al hacer la migración que viene por defecto, reconoció la base de datos de Joomla 1.5 y extrajo todos los productos y categorías. Con esto ya había conseguido una parte de la migración.

Los productos los importamos en Joomla 2.5 manualmente a través del phpMyAdmin Ordenamos y corregimos las categorías, puesto que perdieron la jerarquía.

9.4 Migración avanzada

Al hacer la migración nos dimos cuenta de que no migraba las tallas ni atributos especiales, ya que Virtuemart 2 ha mejorado en ese aspecto separando en una tabla esos atributos (antes los tenia en cada producto). Hemos realizado un script que copia esos atributos a todas las categorías de un producto, además el script copia los métodos de envío (Otra de las cosas que no se importan).

Se compone de diferentes archivos:

```
<?php
//datos de conexion
$GLOBALS['DB_IP'] = 'localhost';
$GLOBALS['DB_USER'] = 'infoanet_usluce';
$GLOBALS['DB_PASS'] = 'lkJHGH6sagFAr';
$GLOBALS['DB_NAME'] = 'infoanet_bdluces';
// Funcion que vamos a usar para realizar la conexion
function get_db_conn() {
if (!$link) {
$link=mysql_connect($GLOBALS['DB_IP'], $GLOBALS['DB_USER'],
$GLOBALS['DB_PASS']);
}
mysql_select_db($GLOBALS['DB_NAME'], $link);
if (!$link) {
echo "Fallo config.php: " . mysql_error();
exit;
}
```

```
return $link;
}
```

Este archivo se llama “config.php” y sólo contiene los datos de acceso a la base de datos y una función para establecer la conexión.

El archivo “shippment.php” tiene dos entradas de texto para los prefijos de la base de datos de una instalación de Joomla y otra, y una vez definido, tiene 2 acciones: Leer y Migrar. Una función lee la base de datos y muestra los métodos de envío. Una vez checkeados que están bien, cambiamos a Migrar y pulsamos el botón. Así escribirá en las tablas de Virtuemart 2 los métodos de envió.

De Virtuemart 1 a 2 cambia la estructura de tablas de los métodos de pago: en Virtuemart 2 han separado la tabla de métodos de envío en dos, así facilita la característica multi-idioma de Virtuemart 2. Otro cambio también aparece a la hora de leer los países: en Virtuemart 1 los países usan un código formado con 3 letras, por ejemplo, “España” es “ESP”, y en Virtuemart 2 utilizan códigos numéricos de 3 dígitos también, por ejemplo, “180” “ESP” “SPAIN”.

Para la migración de los campos personalizados el script lee la base de datos y carga los selectores con las categorías y también busca los atributos especiales (Figura 24).

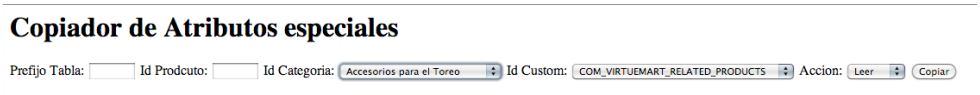


Figura 24 Copiador de atributos especiales.

Una vez rellenado el prefijo de la base de datos y leído los atributos que queremos copiar, rellenaremos la categoría destino y el producto a clonar los atributos.

9.5 Instalación de nuevos módulos y plugins.

Una vez Realizada la migración abrimos la administración de Joomla con /administrator/ vamos a la pestaña Extensiones/instalar (Figura 25).

Figura 25 Seleccionamos el plugin y pulsamos el botón de instalar.

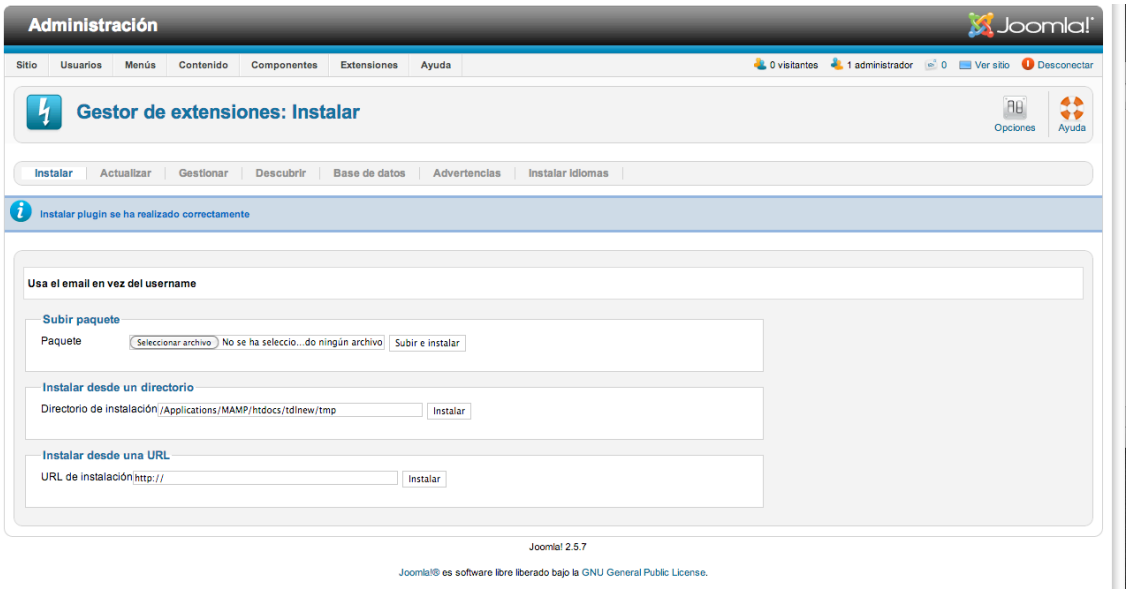


Figura 26. Una vez instalado buscamos en la pestaña Extensiones/plugin y lo habilitamos.

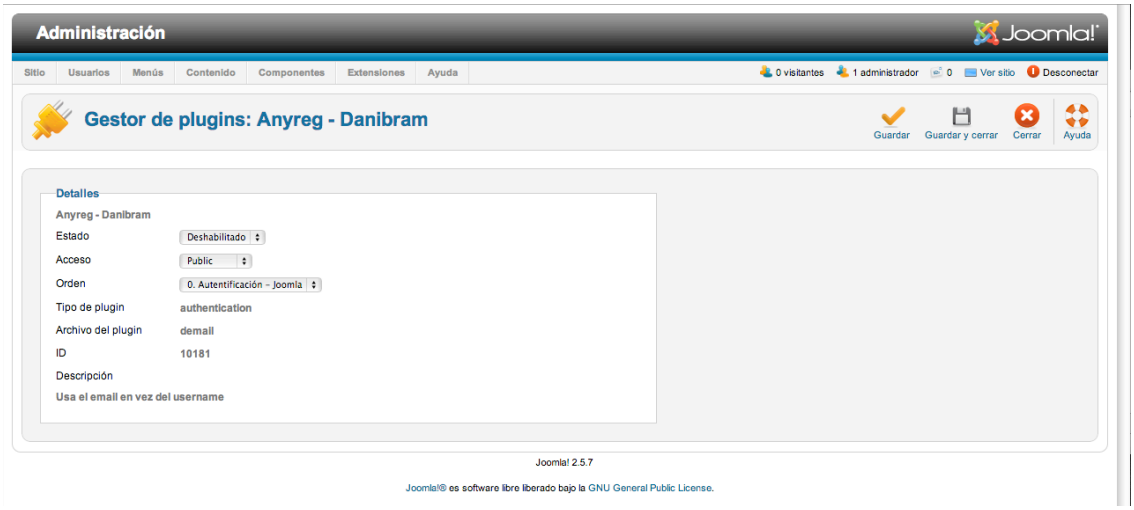


Figura 27. Ahora comprobamos que funciona.

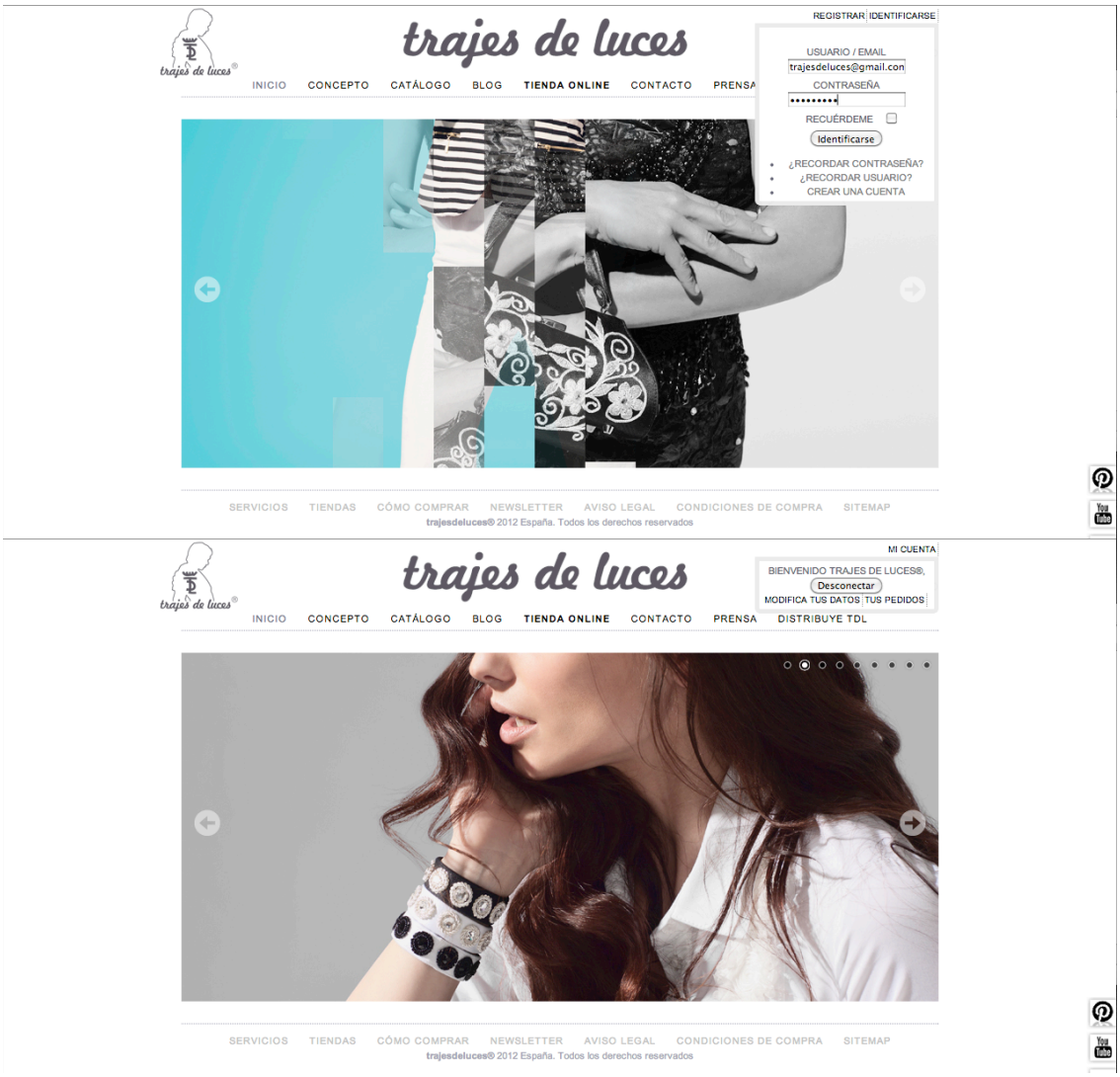


Figura 28 Vemos el funcionamiento

Una vez registrados ya vemos que funciona perfectamente. Seguiremos este proceso con los demás plugins para completar las características de la pagina, en total quedaría así (Figura 29):

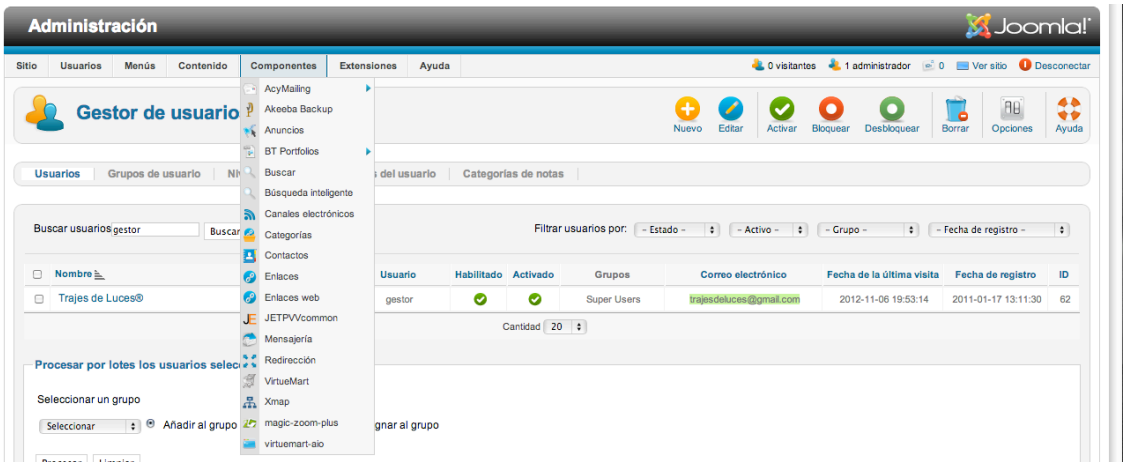


Figura 29. Lista de componentes instalados en Joomla

9.6 Funcionamiento y posibles fallos.

Tras la migración detectamos que existe un problema con las imágenes de los artículos. A algunas imágenes se les ha cambiado el nombre al migrar la base de datos de productos con jUpgrade. La solución pasa por corregir a mano los nombres tanto en la base de datos como el archivo original.

Para evitar que Google redirija mal a los usuarios que buscan sus artículos, hemos realizado una redirección de algunos de los artículos mas visitados, el análisis lo hemos realizado con Google Analytics, ejemplo de redirección 301:

```
Redirect 301 /es http://www.trajesdeluces.com
```

Al acabar este proceso la pagina web ya está lista para subir al servidor la web y realizar el cambio.

10.Instalacion y Funcionamiento de “ecalendar”

Accedemos a la administración de nuestro sitio Joomla (Back-End) vamos a la pestaña “Extensiones/Gestor” de Extensiones (Figura 30).

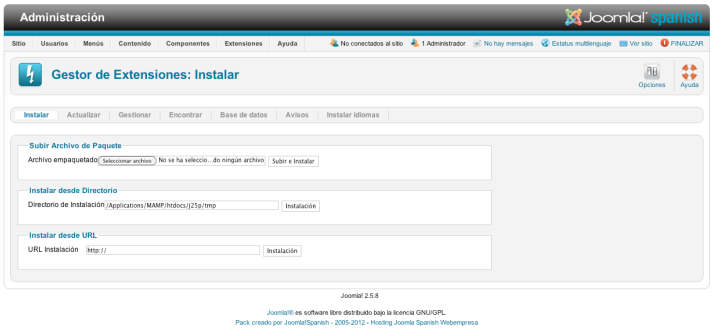


Figura 30. Instalacion del componente

Una vez aquí pulsamos sobre “Subir e instalar”, seleccionamos nuestra aplicación y puslsamos sobre “Instalar” (Figuro 31).

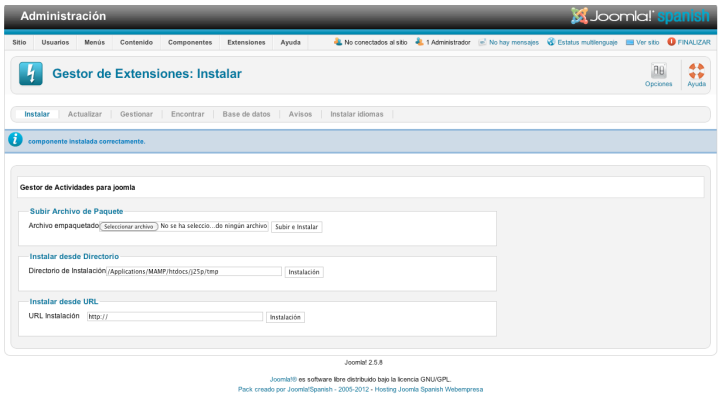


Figura 31. Instalacion correcta

Una vez instalada vamos a componentes, y se nos abre un desplegable y seleccionamos nuestro componente (Figura 32).

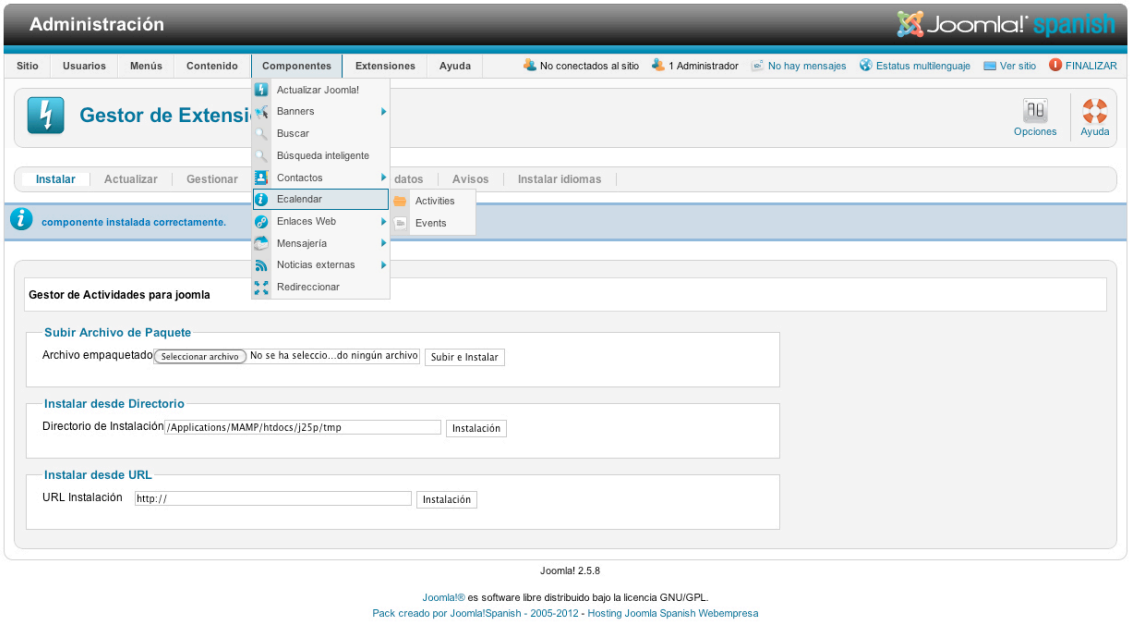


Figura 32. Menu de Componentes de Joomla

Esta (Figura 33), será la pantalla de nuestra aplicación, donde estarán definidas todas las actividades.

Figura 33. Pantalla principal del componente

Si clickamos en “Nueva” podremos definir la nueva actividad, rellenamos nombre, fechas (importante modificar después de la fecha la hora a la que empieza y a la que acaba la actividad), y días de la semana que se repiten. Una vez terminado pulsamos sobre “Guardar & Cerrar” (Figura 34)

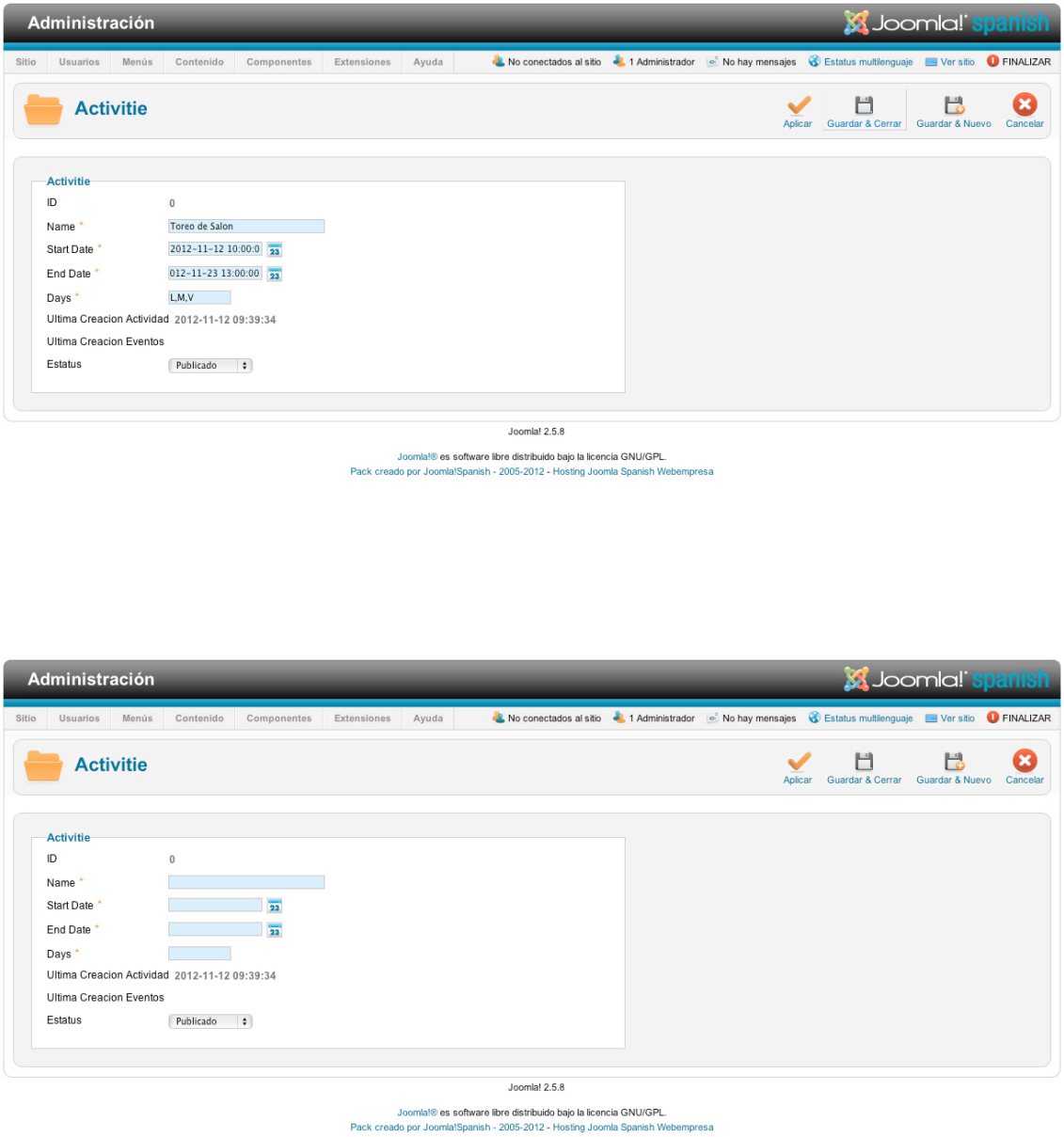


Figura 34. Creacion de un evento

Una vez que se ha guardado ya la vemos en la lista de actividades (figura 35), ahora tendremos que crear los eventos pulsando en el menú de arriba “Crear eventos”

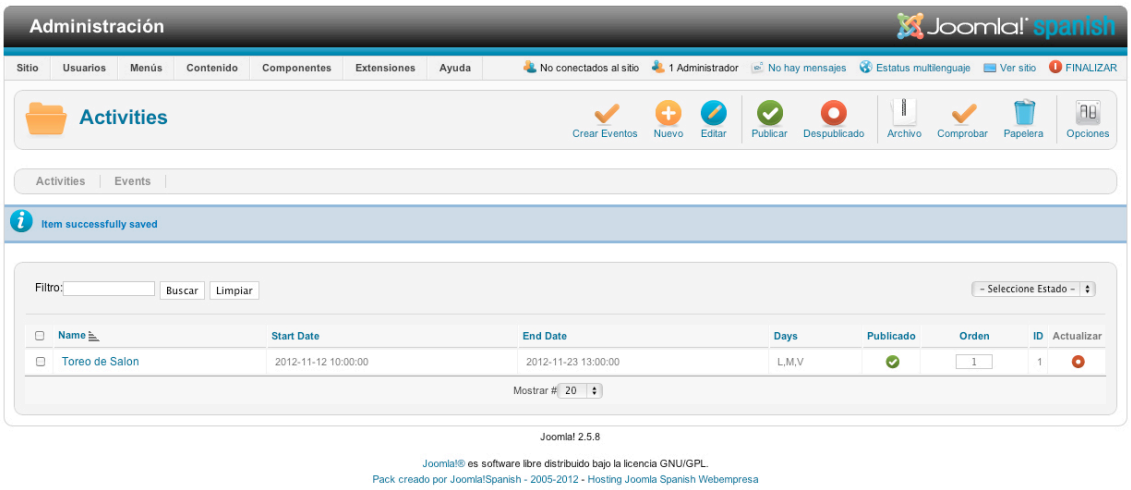


Figura 35. Actividad definida

Una vez que hemos pulsado en “Crear Eventos”, así el icono que nos avisa de la actualización de los eventos se vuelve verde y así nos fijamos que hemos actualizado los eventos. Para comprobarlo clickamos en eventos y veremos la lista (Figura 36).

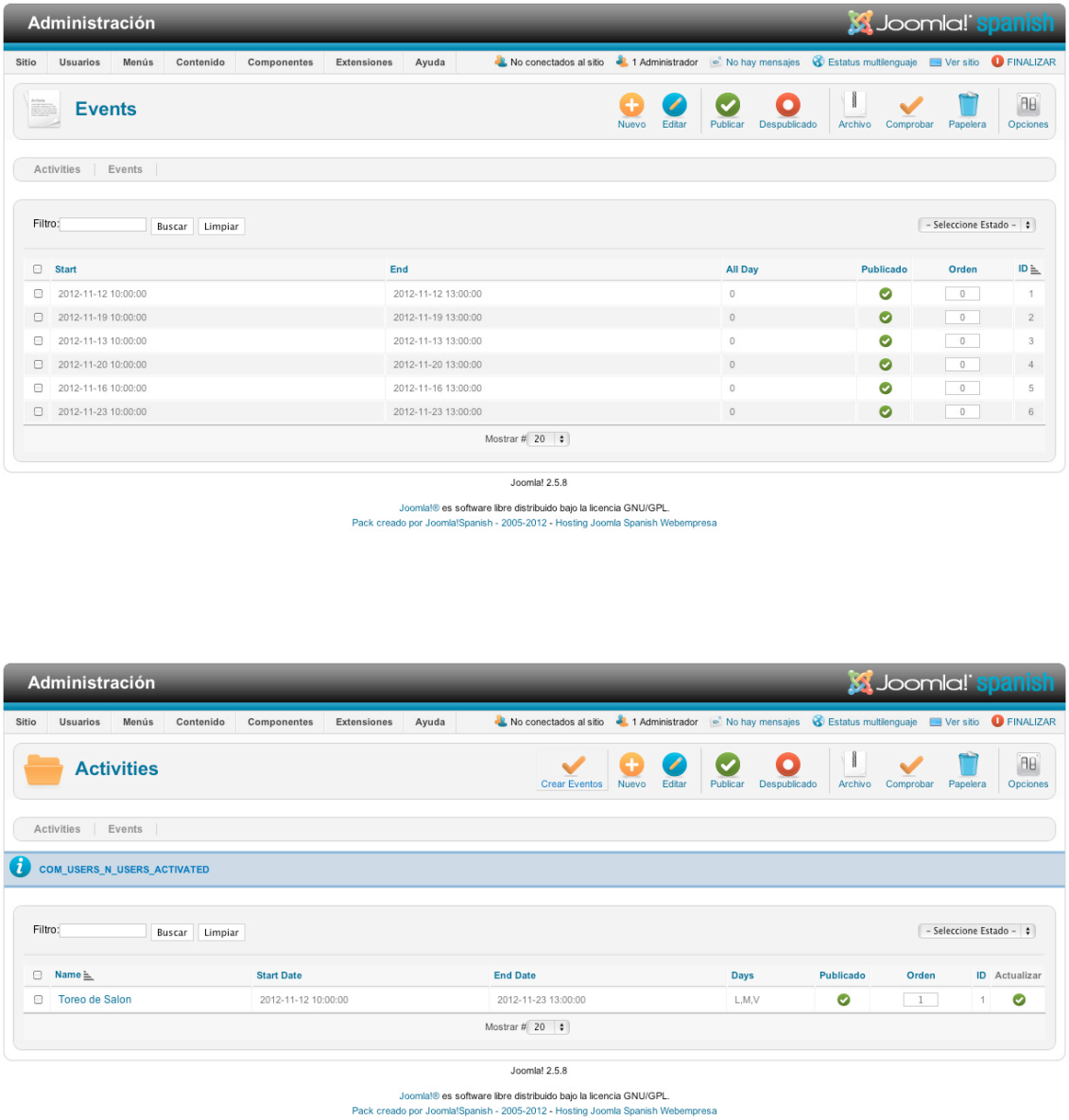


Figura 36. Creacion de los Eventos.

Podemos modificar un evento en concreto seleccionándolo y pulsando en el menú de la parte superior “Editar”, y así lo modificaremos a nuestro gusto (Figura 37).

Figura 37. Modificación de un evento

Ahora una vez configurados los eventos, añadiremos en el menú principal del Front-End una pestaña con el calendario (Figura 38).

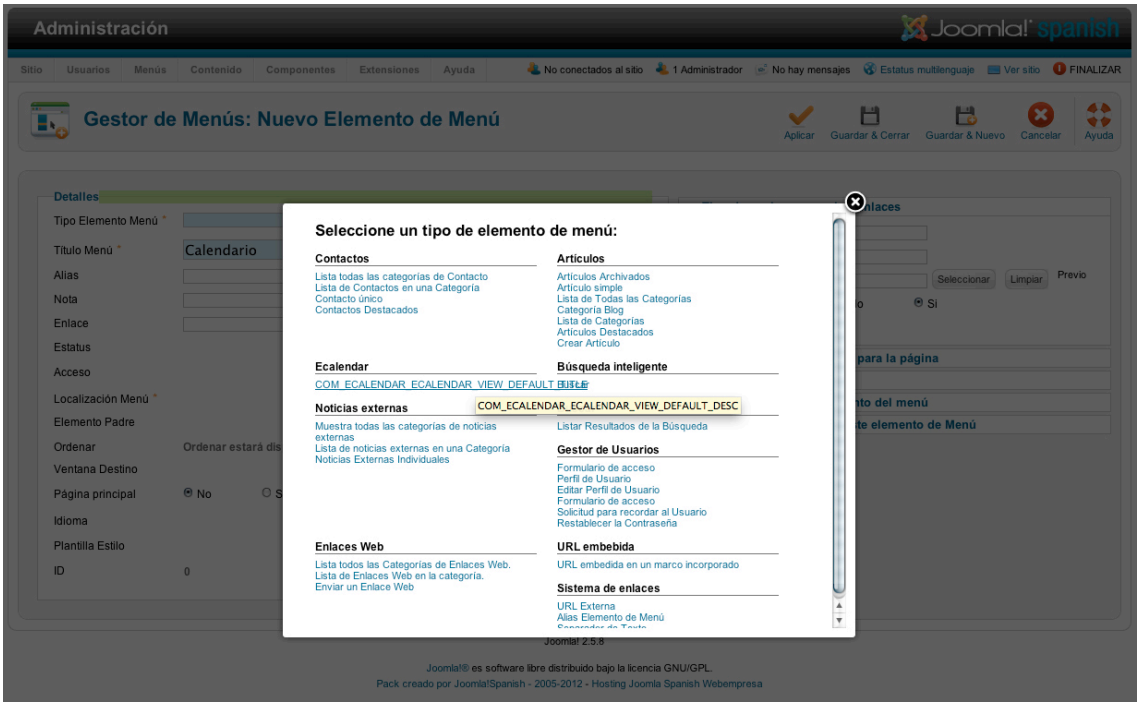


Figura 38. Asignación de nuestro componente

En las opciones de configuración podemos ver que se puede cambiar el dia de comienzo de la semana (Figura 39).

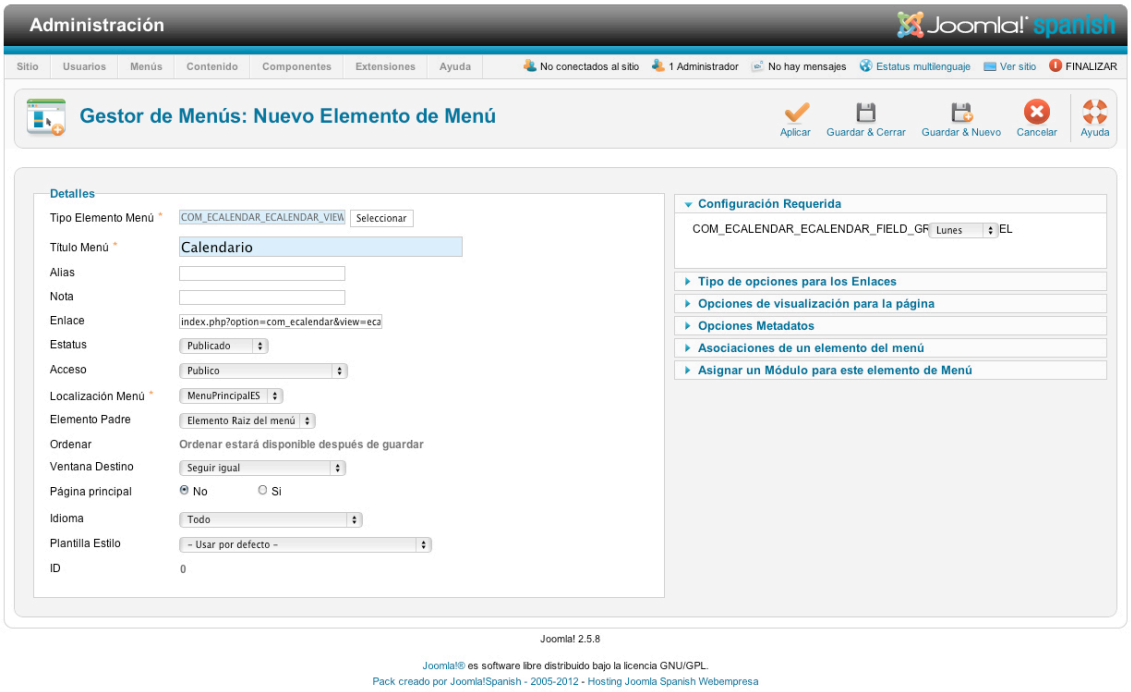


Figura 39. Opciones del Front-End de nuestro componente.

Ahora salimos del Back-End y vamos al Front-End, donde podemos ver en el menú que esta nuestro calendario, si pulsamos sobre el aparecerá (Figura 40)

Figura 40. Nuestro componente en acción.

Podemos comprobar las 2 semanas que hemos definido la actividad.

12.Conclusiones

El desarrollo de una aplicación Web exige el conocimiento de varias tecnologías (CMS, PHP, HTML, CSS, JAVASCRIPT, JSON, JQUERY, AJAX, MySQL, APACHE) que forman un conjunto muy diverso en las que unas en el lado del cliente y otras en el lado del servidor y combinadas de manera adecuada convergen y permiten un amplio abanico de posibilidades en el desarrollo Web.

El hecho de tener que utilizar tantas tecnologías nos permite adquirir un amplio conocimiento de las nuevas tecnologías y herramientas que se utilizan para el desarrollo de webs dinámicas, conocimiento que acompañado del desarrollo continuado, nos permite alcanzar un alto grado de competitividad en la implantación y uso de las nuevas tecnologías y servicios para la Web.

La mejora constante en el conocimiento de la herramienta, el API de desarrollo y las diferentes tecnologías que se utilizan, mejoran las posibilidades de elaborar un buen producto y un aprovechamiento adecuado de las mismas.

En este Proyecto Final de Carrera, se ha desarrollado una solución global compuesta por:

- Desarrollo de plantilla para Joomla con estas características:
  - Amigable para los buscadores
  - Aplicando CSS3 y HTML
  - Estilo Claro y minimalista
  - Modulo oculto para el Log-In
  - Compatible con Virtuemart 2
  - Aspecto de la tienda acorde a la web
  - Botón de añadir al carro sin ver la descripción del producto
  - Vista de productos relacionados en tres columnas
  - Espacios para banners
- Plugin para Joomla
- Componente para Joomla
  - Gestión de clases
  - Gestión de eventos individuales
  - Calendario semanal mensual y diario
- Migración Joomla y Virtuemart:
  - Estabilidad del sistema migrado
  - Scripts propios para una migración completa

Tras el desarrollo del trabajo se ha conseguido que la empresa Trajes de Luces tenga de una manera sencilla un sistema de gestión para sus actividades futuras, y, a la vez, el cambio de imagen en la web hacia su nueva etapa como firma de moda, además que la web pueda seguir vendiendo como hasta ahora sin haber perdido nada ganando la estabilidad y seguridad que ofrece la versión 2.5 de Joomla.

A través de Google Analytics podemos observar un crecimiento del numero de visitas tras el lanzamiento (Figura 27).

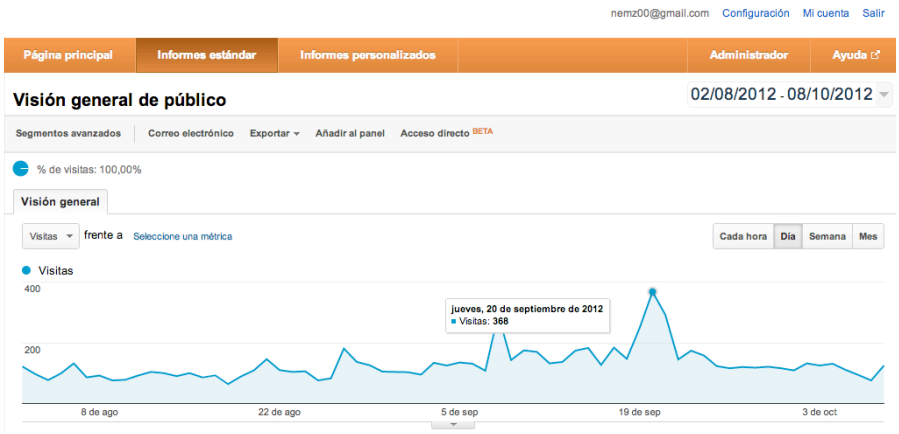


Figura 27

Con ello, logramos cumplir con las especificaciones demandadas por el cliente y hacer un buen trabajo.

13.Trabajo futuro

Para el trabajo futuro, en la parte de la web, sería la adaptación del sistema para añadir el soporte multi-idioma, versión móvil (aunque el ancho de la web se ha fijado en 900 para que los dispositivos móviles puedan abrirla sin problemas).

Para la parte del componente, faltaría añadir el manejo de salas, de usuarios y la gestión económica del centro, pudiendo tener profesores y alumnos y saber el flujo de dinero que pasa por la escuela.

Para el manejo de salas tendríamos en cuenta que las salas tendrían su horario independiente puesto que se puede dar actividades simultáneas dependiendo del número de salas.

En la actualidad no se tiene en cuenta el número de alumnos que están apuntados, se tendría que desarrollar un plugin para añadir campos adicionales al formulario de cada alumno y tener en cuenta a que clases está apuntado y dependiendo de eso saber las horas de clase que cursa.

Para la gestión económica se está pensando en utilizar la librería de gráficos de Google para la generación simple de gráficos. También en un futuro habría que pensar en la posibilidad de compra online de las clases.

Además de esto, falta una mejora de la traducción ya que algunos mensajes no están definidos de manera correcta, también se tendría que poder diferenciar en los eventos cuales son de cada actividad definida.

14. Bibliografía

- http://www.javascriptya.com.ar/jquery/.
  Tutorial de Introducción al uso de JQuery.
- http://jquery.com/
  Página oficial de JQuery.
- http://www.JSON.org
  Página oficial de JSON.
- http://tecnologiasjava.blogspot.com/2010/02/json-javascript-object-notation.html.
  Cómo funciona JSON.
- http://docs.joomla.org/How_to_use_the_JTable_class.
  Como utilizar las clases JTable en Joomla.
- http://www.w3c.es/divulgacion/guiasbreves/ServiciosWeb
  Guía Breve de Servicios Web.
- http://www.javascriptya.com.ar/jquery/.
  Tutorial de Introducción al uso de JQuery.
- http://www.w3c.es/
  Pagina Web para conocer los estándares Web actuales
- http://www.google.es /
  Buscador de Google para solucionar problemas de programación.
- http://es.wikipedia.org//
  Aquí podemos buscar información básica sobre muchas tecnologías.
- http://api.joomla.org/
  Extensiones de Joomla, pagina web oficial.