

**Universidad
Rey Juan Carlos**

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Curso Académico 2009/2010

Proyecto Fin de Carrera

**Creación de sitios web con sistemas de gestión de
contenidos (CMS) con un aspecto personalizado**

Autor: Cristina Huerta García

Tutor: Micael Gallego Carrillo

Copyright

Esta memoria forma parte del Proyecto de Fin de Carrera de la titulación de Ingeniería Técnica de Informática de Gestión en la Escuela Técnica Superior de Ingeniería Informática de la Universidad Rey Juan Carlos de Madrid, realizado por Cristina Huerta García en Junio de 2010.

©Copyright 2010, CHG. Se otorga permiso para copiar, distribuir y/o modificar este documento bajo los términos de la licencia “Reconocimiento-Compartir bajo la misma licencia 3.0 España”, proporcionada por CREATIVE COMMONS.

Agradecimientos

Me gustaría dar las gracias en primer lugar a mi tutor Mica por guiarme en todo el proceso de desarrollo, por su dedicación e intuición.

En segundo lugar, a mis padres por su gran apoyo durante toda la carrera, por no dejarme tirar la toalla en los malos momentos y por haberme mostrado la importancia de estudiar, dándome la oportunidad de tener una carrera universitaria.

A Miguel, por valorarme y darme su apoyo incondicional en la decisión de hacer una pausa y dar el último empujón para terminar la carrera.

Y por último a mis amigos, que me han animado en todo momento a terminar, por preocuparse y ofrecerme su ayuda.

Resumen

En la actualidad el desarrollo continuo de Internet y el aumento de las páginas web ha llevado a la creación (desde 1995) de sistemas CMS (*Content Management System*, Sistemas de Administración de Contenidos), a través de los cuales podremos desarrollar nuevos sitios web, sin necesidad de tener conocimientos de programación.

La mayoría de los CMS son desarrollados por una comunidad de programadores y se ponen al servicio de todo el mundo, software libre. Existen también CMS de pago, que nos proporcionan además del software, soporte, garantía y escalabilidad. Por lo que la utilización de CMS facilita enormemente el desarrollo de sitios web.

Los CMS suelen tener un apartado de administrador y uno para los usuarios, ofreciendo gran interactividad con el portal web incluyendo cambios relativos del diseño, estructura e inclusión de contenidos que en parte son automatizados. Se trata de interfaces conectadas con diversas bases de datos que permiten administrar el contenido de un sitio web. Pero no sólo es la actualización de contenidos, lo más importante es que separan el contenido del diseño. Esto permite que en cualquier momento puedas cambiar el diseño de la web sin modificar el contenido.

Su mayor desventaja es que no permite la personalización completa del diseño del sitio, esto provoca que la mayoría de los sitios web desarrollados mediante CMS presenten un aspecto similar. Para poder diferenciar unos sitios web de otros es necesario utilizar *skin* diferentes, que al aplicarse sobre un determinado software, modifican su apariencia externa, la disposición de los elementos y el estilo visual del sitio web. No podemos aplicar el *skin* directamente a nuestros sitios web desarrollados mediante gestores de contenidos, por este motivo, el objetivo de este proyecto se centra en el estudio de las técnicas que permiten aplicar un *skin* a un sitio web construido con CMS.

Para ello lo primero que haremos será buscar sitios web desde donde podamos descargarnos *skin* gratuitos, después analizaremos el código del *skin* para separarlo del diseño, y por último buscaremos la forma de adaptar ese *skin* a nuestro sitio web en CMS. Existen *skin* gratuitos que podemos descargarnos a través de la web, más adelante proporcionamos una listas con varios de estos sitios.

En definitiva lo que vamos a obtener del proyecto son los mecanismos necesarios para separar el código del *skin* y así poder aplicar plantillas HTML libres sobre gestores de contenidos.

Índice general

1. Introducción	1
1.1. Creación estática de sitios Web	1
1.1.1. HTML.....	2
1.1.2. CSS	2
1.2. CMS	3
1.2.1. XWiki	4
1.2.2. Drupal	5
1.2.3. Joomla.....	6
1.2.4. Wordpress.....	6
2. Objetivos	8
3. Marco de trabajo y Tecnologías	10
3.1. Proceso de Desarrollo	10
3.1.1. Proceso unificado de desarrollo.....	10
3.1.2. Programación Extrema (XP)	12
3.2. Tecnologías	14
3.2.1. HTML.....	14
3.2.2. CSS	15
3.2.3. PHP	17
3.2.4. Velocity	18
3.2.5. <i>Skin</i>	19
3.2.6. XWiki Enterprise.....	20
3.2.7. Drupal	23
3.3. Herramientas	31
3.3.1. Mozilla.....	31
3.3.2. Firebug.....	32
3.3.3. FCKEditor	33
4. Descripción Informática	34
4.1. Especificación de requisitos.....	34
4.1.1. Requisitos Funcionales.....	34
4.1.2. Requisitos no funcionales.....	35
4.2. XWiki.....	36

4.2.1.	Diseño	36
4.2.2.	Implementación	37
4.2.3.	Pruebas	45
4.3.	Drupal	46
4.3.1.	Diseño	46
4.3.2.	Implementación	47
4.3.3.	Pruebas	53
5.	Conclusiones	55
6.	Bibliografía	56

1. Introducción

En las últimas décadas todo gira en torno a Internet, por lo que resulta casi imposible calcular los millones de sitios web que existen en la actualidad. A medida que se incrementa el desarrollo tecnológico se va perfeccionando el diseño de las páginas que utilizan el nuevo software para el logro de sus objetivos. El aspecto estético es fundamental en una página web. No realizar un diseño adecuado hace ver las páginas como obsoletas, poco interesantes y primitivas.

Sin embargo la creación de un diseño web (*webpage design*), también conocido como *skin*, *template* o plantilla entre otros, no es una tarea fácil. En la actualidad existen herramientas que nos facilitan la creación y administración de los sitios web y sus contenidos, son los llamados CMS. Uno de los problemas que presenta la utilización de estos CMS es la dificultad de aplicar de *skin* propios sobre ellos.

Para intentar resolver este problema hemos investigado cómo podríamos aplicar *skin* independientes sobre algunos de los gestores de contenidos más conocidos. A continuación definiremos algunos de los conceptos usados en el desarrollo del proyecto. Seguidamente realizaremos una breve introducción a los CMS más conocidos.

1.1. Creación estática de sitios Web

El contenido de la página puede ser predeterminado (página web estática) o generado al momento de visualizarla o solicitarla a un servidor web (página web dinámica). Las páginas dinámicas que se generan al momento de la visualización, se hacen a través de lenguajes de programación como PHP, Java, Ruby, Phyton, etc., y la aplicación encargada de visualizar el contenido es la que debe generarlo. Las páginas dinámicas que se generan al ser solicitadas son creadas por una aplicación en el servidor web que alberga las mismas.

Una página Web estática presenta las siguientes características:

- No podemos utilizar buscadores.
- Realizadas en XHTML o HTML.
- Para cambiar los contenidos de la página, es imprescindible acceder al servidor donde está alojada la página.
- El proceso de actualización es lento, tedioso y esencialmente manual.
- No se pueden utilizar funcionalidades tales como bases de datos, foros

Como podemos ver las páginas estáticas presentan bastantes limitaciones en cuanto a funcionalidad e interacción con el usuario.

Veamos algunas tecnologías básicas relacionadas con la creación de sitios web estáticos.

1.1.1. HTML

Por el momento, el lenguaje HTML (*HyperText Markup Language, Lenguaje de marcas de hipertexto*) es el lenguaje estándar para la distribución de documentos en la Web. El lenguaje HTML se usa para especificar la presentación de un documento y sus hipervínculos a otros documentos a través del uso de etiquetas de formato. El W3C (*World Wide Web Consortium*), el consorcio internacional que se encarga de estandarizar las reglas de Internet, especificó la versión HTML 4.01 a principios de 2001.

Las especificaciones oficiales de HTML describen las "instrucciones" del lenguaje, pero no cómo seguirlas, es decir, cómo las interpretan los programas informáticos. Esto permite visualizar páginas Web independientemente del sistema operativo o la arquitectura del equipo del usuario.

Las páginas web pueden ser vistas por el usuario mediante un tipo de aplicación llamada navegador, un programa que envía solicitudes a los servidores web, procesa los datos resultantes y muestra la información como se requiere, en base a las instrucciones de la página HTML. Podemos decir por lo tanto que el HTML es el lenguaje usado por los navegadores para mostrar las páginas webs al usuario, siendo hoy en día la interfaz más extendida en la red.

Sin embargo, aunque las especificaciones del W3C son muy detalladas, hay cierto margen para la interpretación por parte del navegador y esta es la razón por la cual la misma página puede aparecer de modo diferente en un navegador u otro.

El lenguaje HTML nos permite aglutinar textos, sonidos e imágenes y combinarlos a nuestro gusto. Además, y es aquí donde reside su ventaja con respecto a libros o revistas, el HTML nos permite la introducción de referencias a otras páginas por medio de los enlaces hipertexto. [1]

1.1.2. CSS

CSS (*Cascading Style Sheets*) es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Separar la definición de los contenidos y la definición de su aspecto presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y

con significado completo (también llamados "documentos semánticos"). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el aspecto de cada elemento: color, tamaño y tipo de letra del texto, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.

1.2. CMS

CMS (*Content Management System*), o sistema de gestión de contenidos, es un software que permite la creación y administración de contenidos (o información), principalmente en la web.

Consiste en una interfaz (unos menús y botones para manejar el sistema) que controla una o varias bases de datos donde se almacena el contenido la información o datos del sitio.

El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio sin tener que darle formato al contenido de nuevo, además de permitir fácilmente y de manera controlada la publicación en el sitio a varios editores o autores.

Existen multitud de CMS diferentes, tanto libres como de pago. Nosotros vamos a agruparlos según el tipo de sitio que permiten gestionar, siendo los más representativos los que listamos a continuación:

- **Genéricos:** Ofrecen la plataforma necesaria para desarrollar e implementar aplicaciones que den solución a necesidades específicas. Pueden servir para construir soluciones de gestión de contenidos, para soluciones de comercio electrónico, blogs, portales,...

Ejemplos: Zope, OpenCMS, Typo3, Apache lenya.

- **Foros:** sitio que permite la discusión en línea donde los usuarios pueden reunirse y discutir temas en los que están interesados.

Ejemplos: phpBB, MyBB.

- **Blogs:** Publicación de noticias o artículos en orden cronológico con espacio para comentarios y discusión.

Ejemplos: Wordpress, Typo.

- **Wikis:** Sitio web dónde todos los usuarios pueden colaborar en los artículos, aportando información o reescribiéndola. También permite espacio para discusiones. Indicado para material que irá evolucionando con el tiempo.

Ejemplos: Mediawiki, Tikiwiki, XWiki.

- **eCommerce:** Son Sitios web para comercio electrónico.
- **Portal:** Sitio web con contenido y funcionalidad diversa que sirve como fuente de información o como soporte a una comunidad.

Ejemplos: Drupal, PHPNuke, Postnuke, Plone.

- **Galería:** Permite administrar y generar automáticamente un portal o sitio web que muestra contenido audiovisual, normalmente imágenes.

Ejemplo: Gallery.

- **e-Learning:** Sirve para la enseñanza de conocimientos. Los usuarios son los profesores y estudiantes, tienen aulas virtuales donde se ponen a disposición el material del curso,... La publicación de un contenido por un profesor es la puesta a disposición de los estudiantes, en un aula virtual, de ese contenido.

Ejemplo: Moodle.

- **Publicaciones digitales:** son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc.

Ejemplo: ePrints.

A continuación pasamos a describir de forma breve algunos de los CMS más relevantes y más utilizados en la actualidad.

1.2.1. XWiki

XWiki, es un wiki de plataforma genérica, un sitio web cuyas páginas pueden ser editadas por múltiples usuarios de colaboración a través del navegador web. En la Figura 1.1 podemos ver el logo de XWiki.

XWiki es un motor wiki implementado en Java bajo la licencia de código abierto LGPL, que utiliza un motor de base de datos y un lenguaje de programación, Velocity, que permite la creación rápida de módulos y la representación del sitio web. Uno de los wiki más conocido es la enciclopedia libre de colaboración Wikipedia.

Permite el desarrollo rápido de aplicaciones ya que se compone de múltiples herramientas que facilitan su mantenimiento y configuración, como controladores de versiones y cambios, anexo de ficheros, paneles de control de objetos, gestión de privilegios, la renderización de páginas, el almacenamiento transparente, manipulación de las propias páginas (editar, eliminar, renombrar, ver), autenticación, macros, secuencias de comandos (Velocity, Groovy) y más.

Estos servicios están disponibles como las API en Java, XML-RPC / y GWT. Las API XWiki son métodos que el equipo de desarrollo XWiki considera seguros para su utilización, garantizando su compatibilidad. Se pueden llamar desde Java o directamente desde sus páginas wiki usando un lenguaje de script (ya sea Velocity o Groovy). [2]



Figura 1.1: Logo Xwiki

1.2.2. Drupal

Drupal es uno de los CMS más utilizado. Es un marco de administración de contenidos, sistema de administración de contenidos y motor de *blogging* basado en PHP (*PHP Hypertext Pre-processor*), fue originalmente escrito por Dries Buytaert y es el software usado para impulsar los sitios web Debian Planet, Terminus1525, Spread Firefox y Kernel Trap, entre otros. Drupal es la ortografía (deletreo) inglesa para la palabra neerlandesa “*druppel*” que significa “gota”. Vemos su logotipo en la Figura 1.2.

El texto y los enlaces entre el contenido son almacenados en una base de datos, y las páginas se construyen dinámicamente para ser presentadas al usuario en respuesta a una "petición web" mediante un navegador.

Está orientado hacia la capacidad de configuración y personalización de los contenidos y módulos de nuestra aplicación. Es una herramienta de creación de sitios web muy flexible, con la que personas con o sin conocimientos de programación pueden crear, configurar y administrar su sitio web. [3]



Figura 1.2: Logo Drupal

1.2.3. Joomla

Es otro de los CMS de código abierto, desarrollado con el lenguaje de programación PHP (*PHP Hypertext Pre-processor*) bajo una licencia GPL. Joomla! está diseñado para realizar sitios web tanto para Internet como para intranets. Para su funcionamiento necesita un servidor web con soporte para PHP y una base de datos MySQL Server.

Debido a la gran difusión de este proyecto, se han desarrollado multitud de módulos para Joomla!: noticias, blogs, foros, encuestas, calendarios, galerías de imágenes, descargas, etc.

Su nombre es una pronunciación fonética para anglófonos de la palabra swahili jumla que significa "todos juntos" o "como un todo", podemos ver su logo en la Figura 1.3. Se escogió como una reflexión del compromiso del grupo de desarrolladores y la comunidad del proyecto.

La primera versión de Joomla! (Joomla! 1.0.0) fue publicada el 16 de septiembre de 2005. Se trataba de una versión mejorada de Mambo 4.5.2.3 combinada con otras modificaciones de seguridad y "anti-bugs". Actualmente los programadores han publicado Joomla! 1.5 estable bajo un código completamente reescrito y construido bajo PHP 5. [4]



Figura 1.3: Logo de Joomla!

1.2.4. Wordpress

Es un sistema de gestión de contenidos enfocado a la creación de blogs (sitios web periódicamente actualizados). Por medio de este software e interfaces, sus usuarios pueden crear sus propios blogs de una manera sencilla y personalizada.

Como sistema de gestión de contenidos (CMS), WordPress consiste en un programa que permite una estructura de soporte para que usuarios de Internet puedan crear y administrar contenidos. El CMS controla bases de datos que permiten manejar el diseño y el contenido de una página web. Es por eso que con WordPress, los usuarios pueden cambiar, cuantas veces quieran, el diseño de su página o blog, sin necesidad de cambiar el formato. Además, permite la publicación de material de una manera sencilla para luego ser vista por otros usuarios.

Para usar WordPress el usuario sólo necesita de un correo electrónico como requisito y luego puede bajar el programa o "script", que es totalmente gratuito. Con WordPress, se puede disponer de un corrector ortográfico para escribir en el blog o comentar en otras páginas. También se pueden subir fotos, insertar videos y todo de manera sencilla. Además, tiene un servicio de bloqueo de *spam* (correo basura) que puede provenir de los comentarios de otros usuarios y también los que provienen de blogs que tratan de escabullirse en WordPress (es importante señalar que son frecuentes los ataques a estos sistemas masivos de blogs; los *hackers* permanentemente buscan nuevas formas de vulnerar estos sistemas).

Desarrollado en PHP y MySQL, bajo licencia GPL, su facilidad de uso, su enfoque hacia la elegancia y la estética, y todas sus atractivas características lo han convertido en una de las plataformas de publicación personal más populares de la blogosfera.

WordPress es el sucesor de *b2\cafelog*, conocido como *b2* o *cafelog*, creado con el deseo de tener un sistema de publicación elegante y bien diseñada, WordPress apareció por primera vez en el año 2003. [5]

Las causas de su enorme crecimiento son, entre otras, su licencia, su facilidad de uso y sus características como gestor de contenidos. En la Figura 1.4 podemos ver el logotipo de este CMS.



Figura 1.4: Wordpress

2. Objetivos

El uso de estos CMS facilita enormemente al usuario tanto la creación como la administración y posterior mantenimiento de su sitio web.

A continuación enumeramos algunas de las ventajas y desventajas de usar un CMS:

- Gestión descentralizada
- Mayor rapidez en la gestión de nuevas páginas y cambios en las mismas
- Mayor consistencia de contenidos
- Navegación mejorada del sitio web
- Flexibilidad creciente del sitio web
- Ayuda a una descentralización del sitio web
- Mayor seguridad del sitio web
- Reducción duplicidad de información
- Mayor capacidad de el crecimiento
- Reducción de los costes de mantenimiento
- Su mayor desventaja es que no permite la personalización completa de funcionamiento y diseño del sitio.

La modificación de un *skin* es una tarea complicada, tiene la ventaja de que podemos modificar el aspecto de nuestro software para que sea más atractivo, pero a su vez el cambio de apariencia puede complicar el apoyo técnico y la información. Una interfaz de usuario que ha sido fuertemente personalizada por un usuario, puede parecer completamente desconocida a otro que conoce el software bajo una apariencia diferente, ya sea la original de éste u otro *skin*.

Los profesionales de software a menudo describen que la flexibilidad se puede ver afectada por la aplicación de un *skin* y que se requiere que el usuario sea un diseñador experto para adaptar el software y conseguir un mejor uso.

La ventaja de los CMS es que nos permiten personalizar nuestro sitio web ya que ofrecen *skin* que hacen principalmente cambios estéticos. [6]

A pesar de que existen multitud de *skin*, los nuevos sitios web requieren de una personalización para poder hacerlos únicos. Es como el logotipo de una empresa, debe ser único para poder diferenciarla del resto.

Otro de los problemas que podemos encontrarnos es si queremos migrar nuestro sitio web a uno de estos CMS y queremos mantener el aspecto o apariencia que teníamos hasta entonces.

Hemos realizado un estudio sobre cómo aplicar *skins* libres sobre estos CMS, para así poder utilizar cualquier plantilla HTML sobre nuestro sitio Web desarrollado mediante un gestor de contenidos y poder tener el diseño deseado.

Se plantearon dos alternativas a la hora de abordar del problema: La primera consistía en ir modificando el *skin* que estuviéramos utilizando en nuestro CMS, por ejemplo el que viene por defecto, e ir adaptándolo hasta que tuviera el aspecto de nuestra plantilla nueva.

La segunda alternativa consistía en aplicar directamente el nuevo *skin* e ir arreglando los problemas que fuéramos encontrando por el camino, como modificar el código fuente, los CSS,...etc.

Comenzamos aplicando la primera de las opciones, pero debido a la cantidad de problemas encontrados, sobre todo a nivel de hojas de estilo, finalmente se optó por la segunda alternativa para los dos CMS que nos ocupan, Drupal y XWiki.

Hay que tener en cuenta que los *skin* libres que hemos utilizado son en lenguaje HTML, sin embargo los CMS utilizan otro tipo de lenguajes adicionales para la generación de contenido dinámico como: Velocity, PHP,... por lo que hemos tenido que analizar en profundidad cada código fuente encontrado para adaptar el *skin* a estos CMS.

3. Marco de trabajo y Tecnologías

3.1. Proceso de Desarrollo

Cada proyecto software tiene características diferentes al resto y por ello es necesario adaptar la metodología, también llamado proceso de desarrollo software, de manera que se realicen una serie de tareas entre la idea inicial y el resultado final.

Para la realización de este proyecto no ha sido necesario elegir una metodología, ya que no se trata de un proyecto de desarrollo como tal, sino de un proyecto de investigación de aplicaciones, gestores de contenidos, lenguajes de programación y *skin* para su integración.

Aún así, comentaremos las diferencias entre dos de las metodologías más empleadas en la actualidad y opuestas entre sí. También haremos una pequeña introducción al Proceso unificado de Desarrollo para entender de forma más clara las diferencias entre ambas metodologías.

3.1.1. Proceso unificado de desarrollo

El Proceso Unificado de Desarrollo (PUD) reúne las tres metodologías de desarrollo basadas en el paradigma de la orientación a objetos:

- OOSE: Object Oriented Software Engineering, Ivar Jacobson (Casos de Uso).
- OMT: Object Modeling Technique, James Rumbaugh, (Análisis).
- Booch, Grady Booch (Diseño).

Está basado en componentes, lo que quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidos. Utiliza el lenguaje UML para expresar gráficamente todos los diagramas que definen el sistema del software.

Los aspectos que caracterizan al Proceso Unificado son tres:

- **Dirigido por Casos de Uso:** Una interacción con el usuario es un caso de uso, que captura los requisitos funcionales. Conducen el proceso de desarrollo mediante modelos de diseño e implementación, creados por los desarrolladores, mediante pruebas. Además estos modelos evolucionan junto a la arquitectura del sistema y se validan para que sean conformes a los casos de uso.

- **Centrado en la arquitectura:** El concepto de arquitectura software incluye los aspectos estáticos y dinámicos más significativos del sistema, tales como: plataforma (BBDD, SO, protocolo de comunicación,...), aspectos legales, componentes reutilizables disponibles, requisitos no funcionales, etc. Es una vista del diseño completo que hace visibles las características principales.
- **Iterativo e incremental:** Todo sistema informático es complejo y suele durar desde meses hasta años. Por lo que lo más eficiente es dividir el proyecto en pequeñas fases. Al terminar cada fase hay que revisarla y probarla. Las fases se dividen en iteraciones, y cada iteración se basa en la anterior, por lo que se produce un incremento, que no siempre es aditivo.

El ciclo de vida del software se divide en 4 fases, cada una de las cuales tiene varias iteraciones. Una iteración representa un ciclo de desarrollo completo que consta de: requisitos, análisis, diseño, implementación y pruebas. El énfasis en cada flujo de trabajo es diferente dependiendo de la fase en que se encuentre. Las fases son: concepción, elaboración, construcción y transición.

La concepción es definir el alcance del proyecto, definir los casos de uso y analizar los riesgos. La elaboración es proyectar un plan, definir las características y cimentar la arquitectura. La construcción es crear el producto y la transición es transferir el producto a sus usuarios.

En la Figura 3.1 se puede ver gráficamente la estructura y la distribución del flujo de trabajo.

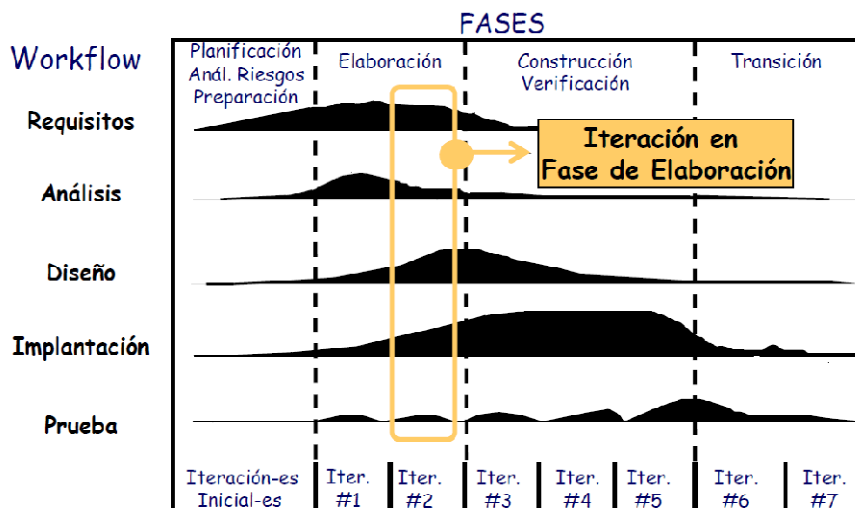


Figura 3.1: Estructura del PUD.

3.1.2. Programación Extrema (XP)

La programación extrema, Extreme Programming (XP), es una de las llamadas "Metodologías Ágiles" de desarrollo de software más populares actualmente. La programación extrema se basa en unos valores, unos principios fundamentales y unas prácticas.

Los principales valores de la programación extrema son los siguientes:

- Comunicación: La comunicación entre los usuarios y desarrolladores.
- Simplicidad: La simplicidad al desarrollar y codificar los módulos del sistema.
- Realimentación.
- Coraje: El coraje tiene el lema "si funciona mejóralo".
- Humildad: Compartir código con los usuarios.

Los principios fundamentales de esta metodología son:

- Realimentación veloz.
- Modificaciones incrementales.
- Trabajo de calidad.
- Asunción de simplicidad.

Y por último, las prácticas de esta metodología se podrían resumir en las siguientes:

- Equipo completo: Pertenecen al equipo todos los desarrolladores del proyecto y el cliente.
- Planificación: Se hacen las historias de usuario y se planifican las mini-versiones. La planificación se revisa continuamente.
- Test del cliente: El cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las mini-versiones.
- Versiones pequeñas: Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no trozos de código que no pueda ver funcionando.
- Diseño simple: Hacer siempre lo mínimo imprescindible de la forma más sencilla posible.
- Importante mantener siempre sencillo el código.
- Pareja de programadores: Los programadores trabajan por parejas.
- Desarrollo guiado por las pruebas automáticas: Se deben realizar programas de pruebas automáticas. Deben realizarse con frecuencia.

- Mejora del diseño: Mientras se codifica, debe mejorarse el código ya hecho. Como por ejemplo eliminar líneas de código innecesarias, añadir nuevas funciones, no repetir código.
- Integración continua: Debe tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recompilarse y probarse.
- El código es de todos: Cualquiera puede y debe tocar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
- Normas de codificación: Debe haber un estilo común de codificación de forma que parezca que ha sido realizado por una única persona.
- Metáforas: Poner nombres o frases que definan correctamente la funcionalidad de las distintas partes del programa, sin que haya lugar a equivocación.
- Ritmo sostenible: El ritmo de trabajo debe mantenerse durante el desarrollo del proyecto. Nunca se deben dejar unos días muertos. Hay que llevar un ritmo continuo.

En nuestro caso ninguna de estas metodologías se adapta a nuestro trabajo, ya que al no tratarse de un proyecto de desarrollo, no hemos seguido ninguna de las metodologías de trabajo típicas.

Hemos decidido seguir una metodología con las siguientes fases:

- Buscar un *skin* libre
- Estudio de la tecnología de *skins* del CMS utilizado en cada caso.
- Análisis de los lenguajes script encontrados en cada CMS.
- Integración del *skin* en el CMS correspondiente.
- Pruebas funcionales

3.2. Tecnologías

Dentro de este apartado se expondrán las diferentes tecnologías software utilizadas para la realización del proyecto. También se describirá el entorno en el que se ha trabajado y el software que ha intervenido en la gestión del proyecto.

3.2.1. HTML

HTML es lo que se conoce como "lenguaje de marcado", cuya función es preparar documentos escritos aplicando etiquetas de formato. Las etiquetas indican cómo se presenta el documento y cómo se vincula a otros documentos. Junto con el código HTML se enlazan otros recursos como imágenes y sonidos, que se incluyen en archivos separados. HTML se usa también para la lectura de documentos en Internet desde diferentes equipos gracias al protocolo HTTP, que permite a los usuarios acceder, de forma remota, a documentos almacenados en una dirección específica de la red, denominada dirección URL.

La funcionalidad del HTML es tan sencilla que puede ser creado y editado en cualquier editor de textos básicos. También puede editarse en procesadores de textos, software de diseño web o aplicaciones web directamente, como lo más convencionales programas de administración de contenido.

En la Figura 3.2 mostramos un pequeño ejemplo de programación en código HTML, con la que pintaremos un formulario. Seguidamente en la Figura 3.3 vemos su representación en un navegador.

```
<html>
<head><title>Formulario de registro</title></head>
<body>
<center><h3>Cambridge English Center</h3>
      <h4>Formulario de Registro</h4>
</center>
<hr>
Sea tan amable de rellenar el siguiente formulario.
<br>
<FORM ACTION="mailto:dirección_de_email" METHOD="POST"
ENCTYPE="TEXT/PLAIN" >
Nombre: <BR>
<INPUT TYPE="text" NAME="Apellido"><br>
Dirección: <BR> <INPUT TYPE="text" NAME="Dirección"><br>
Teléfono: <BR> <INPUT TYPE="text" NAME="Telefono"><br>
Comentarios:<br> <TEXTAREA NAME="comentario" ROWS="7" COLS="40">
</TEXTAREA>
<INPUT TYPE="submit" VALUE="Enviar datos">
<INPUT TYPE="reset" VALUE="Borrar datos">
</FORM>
</body>
</html>
```

Figura 3.2: Ejemplo de código fuente de un formulario HTML.

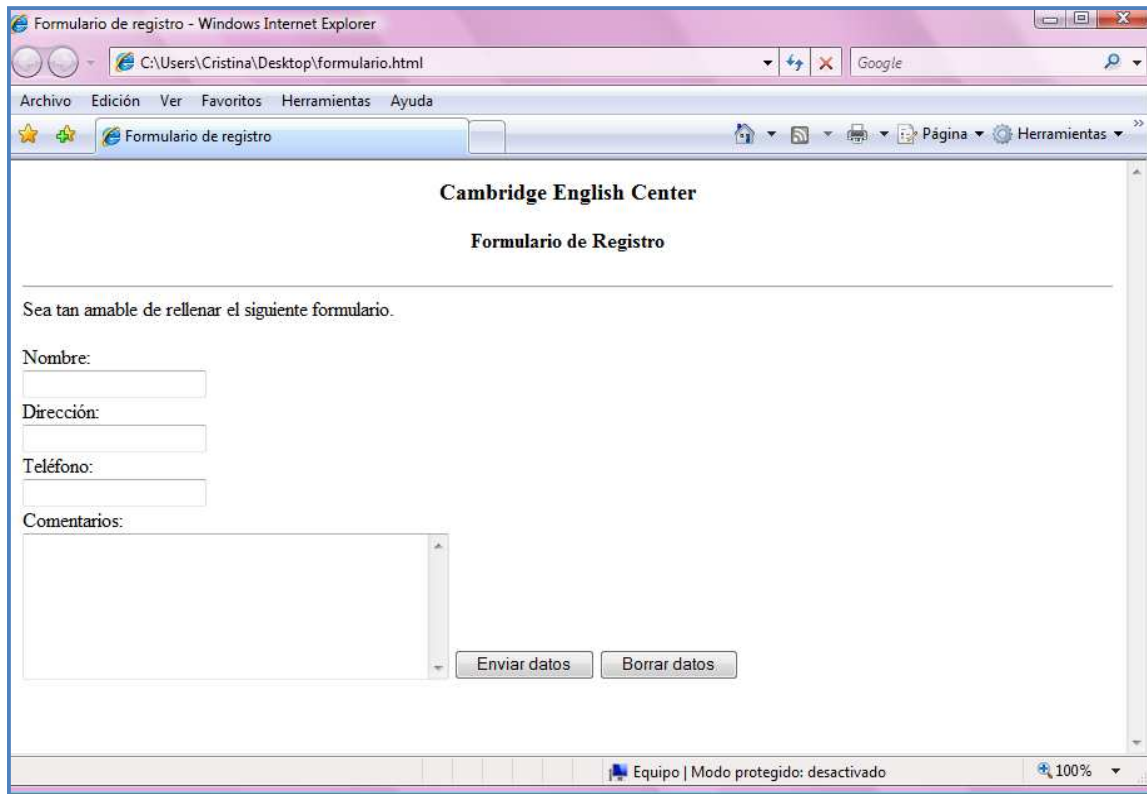


Figura 3.3: Representación en navegador del código fuente HTML

3.2.2. CSS

CSS (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento. [7]

El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial, la forma de presentación que le aplicaremos a:

- Un web entero, de modo que se puede definir la forma de todo el web de una sola vez.
- Un documento HTML o página, se puede definir la forma, en un pequeño trozo de código en la cabecera, a toda la página.
- Una porción del documento, aplicando estilos visibles en un trozo de la página.

En la Figura 3.4 vemos un ejemplo de código HTML al que aplicamos una CSS.

```

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <title>Mi primera página con estilo</title>
  <style type="text/CSS">
    body {
      color: purple;
      background-color: #Aaffaa}
    </style>
</head>
<body>
<!-- Menú de navegación del sitio -->
<ul class="navbar">
  <li><a href="indice.html">Página principal</a>
  <li><a href="enlaces.html">Enlaces</a>
</ul>
<!-- Contenido principal -->
<h1>Mi primera página con estilo</h1>
<p>¡Bienvenido a mi primera página con estilo!
<address>Creada el 8 de Junio de 2010.</address>
</body>
</html>

```

Figura 3.4: Ejemplo código HTML y CSS

Seguidamente en la Figura 3.5 mostramos cómo afecta el código CSS que hemos insertado en el HTML mediante la representación a través de un navegador web.

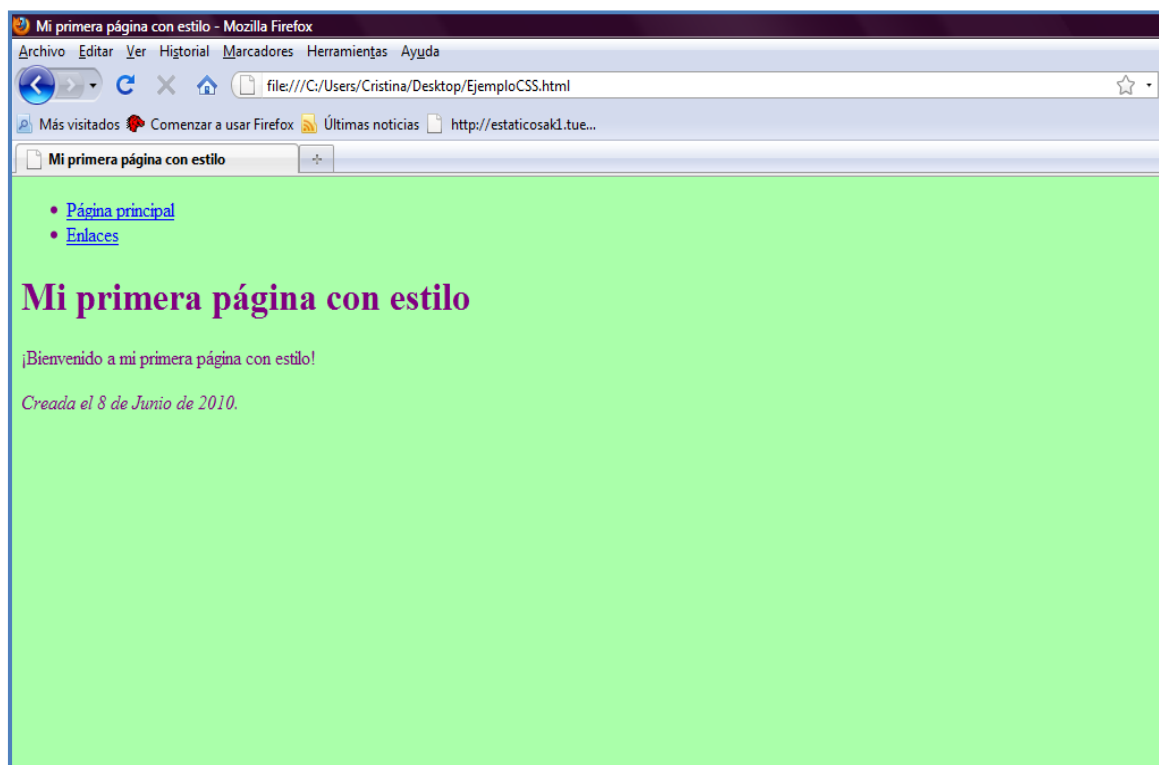


Figura 3.5: Ejemplo página HTML con CSS

3.2.3. PHP

PHP es un lenguaje interpretado de propósito general ampliamente usado, diseñado especialmente para desarrollo web y que puede ser incrustado dentro de código HTML. Se trata de un lenguaje dinámico, no necesita compilación, se ejecuta en un servidor web tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. La versión más reciente de PHP es la 5.3.2 (para Windows) del 04 de marzo de 2010.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta. También les permite involucrarse con aplicaciones de contenido dinámico sin tener que aprender todo un nuevo grupo de funciones.

Aunque todo en su diseño está orientado a facilitar la creación de página web, es posible crear aplicaciones con una interfaz gráfica para el usuario, utilizando la extensión PHP-QT o PHP-GTK.

También puede ser usado desde la línea de órdenes, de la misma manera como Perl o Python pueden hacerlo, a esta versión de PHP se la llama PHP CLI (Command Line Interface).

Cuando el cliente hace una petición al servidor para que le envíe una página web, el servidor ejecuta el intérprete de PHP. Éste procesa el script solicitado que generará el contenido de manera dinámica (por ejemplo obteniendo información de una base de datos). El resultado es enviado por el intérprete al servidor, quien a su vez se lo envía al cliente. Mediante extensiones es también posible la generación de archivos PDF, Flash, así como imágenes en diferentes formatos.

Permite la conexión a diferentes tipos de servidores de bases de datos tales como MySQL, Postgres, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite.

XAMPP es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GPL y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP está disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS X.

PHP también tiene la capacidad de ser ejecutado en la mayoría de los sistemas operativos, tales como UNIX (y de ese tipo, como Linux o Mac OS X) y Windows, y

puede interactuar con los servidores de web más populares ya que existe en versión CGI, módulo para Apache, e ISAPI. [8]

3.2.4. Velocity

Hay una gran variedad de posibles opciones cuando se trata de producir contenido web dinámico. Normalmente, la mayoría de los proyectos caen en lo ya aceptado como estándar, es decir si la plataforma elegida para tu proyecto es Java, tendrás que utilizar JSP. Sin embargo, algunas veces utilizar el estándar no es la mejor elección. Quizás necesites o quieras un control más directo sobre el código que produce tu contenido. [9]

Velocity es un motor de plantillas basado en Java. Permite a los diseñadores de páginas hacer referencia a métodos definidos dentro del código Java. Los diseñadores Web pueden trabajar en paralelo con los programadores Java para desarrollar sitios de acuerdo al modelo de Modelo-Vista-Controlador (MVC), permitiendo que los diseñadores se concentren únicamente en crear un sitio bien diseñado y que los programadores se encarguen solamente de escribir código de primera calidad.

Velocity separa el código Java de las páginas Web, haciendo el sitio más fácil de mantener a largo plazo y presentando una alternativa viable a *Java Server Pages* (JSP) o PHP.

El Lenguaje de Plantillas de Velocity (VTL) fue creado para probar la manera más fácil, simple y limpia de incorporar contenido dinámico dentro de una página web. Incluso un desarrollador de páginas web con poca o ninguna experiencia puede rápidamente ser capaz de utilizar VTL para incluir contenido dinámico en un sitio web. VTL usa “referencias” para incluir contenido dinámico dentro de un sitio web. Una variable es un tipo de referencia que puede referirse a algo definido dentro del código Java u obtener su valor de un enunciado VTL en la página misma. [10]

En Velocity, por cada página (o por cada grupo de páginas), escribe un *servlet*. Su *servlet* extrae todos los datos necesarios para esa página, los sitúa en un contexto, y finalmente, le dice a Velocity que dibuje ese contexto utilizando una plantilla. La plantilla contiene etiquetas HTML y directivas Velocity (no código Java).

Por esto es tan efectivo; forzándonos a limitar severamente el código de la plantilla, simplificamos las páginas, permitiendo de forma efectiva que se puedan entregar a un diseñador web, como no se puede hacer con JSP.

Se puede utilizar para crear páginas web, SQL, PostScript y cualquier otro tipo de salida de plantillas. Se puede utilizar como una aplicación independiente para generar código fuente y reportes, o como un componente integrado en otros sistemas.

3.2.5. Skin

Un *skin* o piel, también llamado *template*, plantilla, *theme* o tema, consiste en una serie de elementos gráficos que, al aplicarse sobre un determinado software, modifican su apariencia externa, la disposición de los elementos y el estilo visual de los sitios web. Los *skins* son independientes de la aplicación, con lo que esta puede tener entre sus opciones varios de estos *skin* o ninguno, sin embargo hasta ahora, cada *skin* se podía aplicar exclusivamente sobre un software determinado, no pudiendo enviarse a otros programas.

Hemos estudiado la forma en la que los CMS seleccionan y pintan sus diseños para poder aplicar *skin* libres sobre ellos. Drupal los denomina *theme* o temas, al igual que Wordpress y Gallery entre otros; los Wikis (Mediawiki, Tikiwiki, XWiki) los llaman *skin*, para Joomla son simplemente *templates* o plantillas y en varios sitios web se los conoce como *webdesign*.

A continuación mostramos en la Figura 3.6 algunos ejemplos de *skins* libres disponibles en la web:

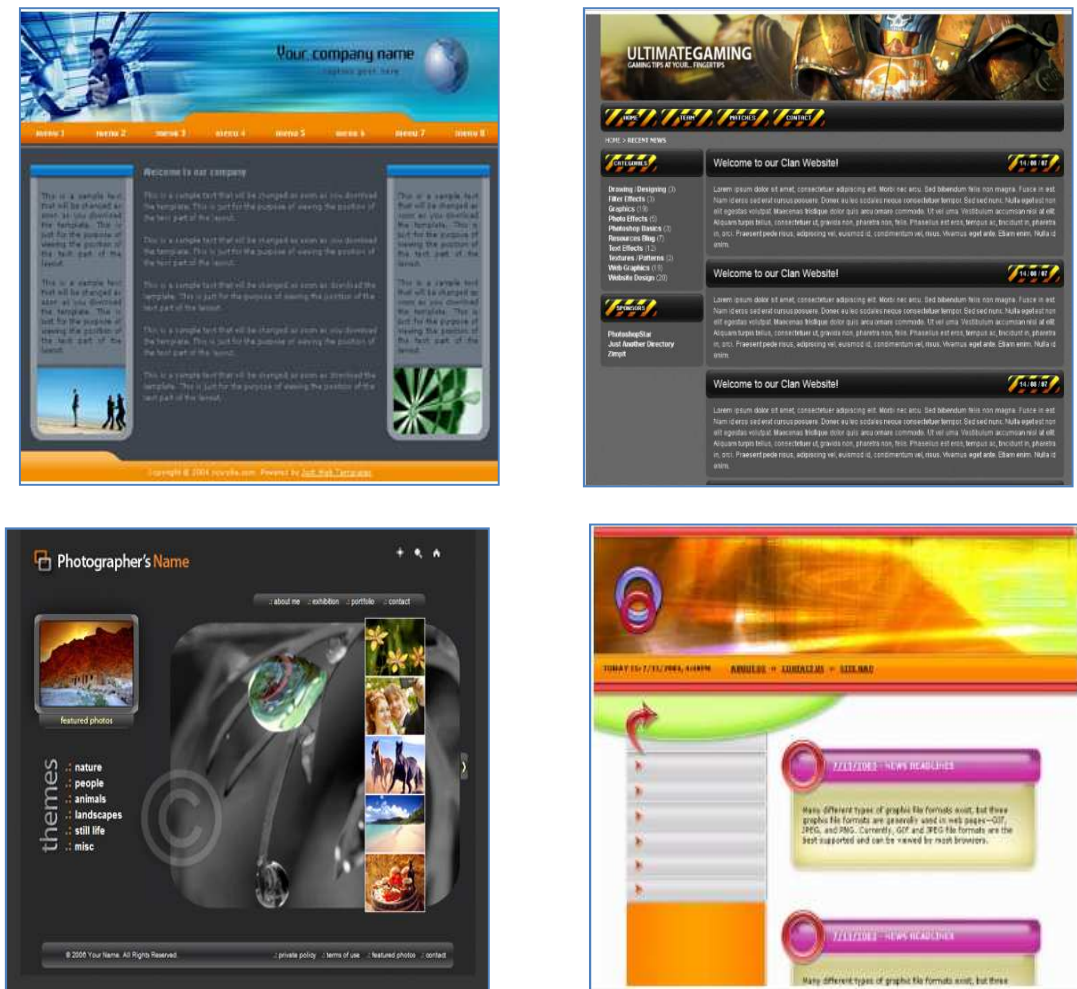


Figura 3.6: Diseños de skins

Existen varios sitios desde los que podemos descargar *templates* libres, a continuación listamos algunos de ellos.

- <http://www.eplantillas.com/plantillas-gratis/>
- <http://www.templatemonster.com/free-templates.php>
- <http://freeskins.blogspot.com/>
- <http://www.themelab.com/free-wordpress-theme/>
- <http://www.programaswarez.com/descargar-templates-gratis/>
- <http://mediawiki2u.com/>
- <http://skins.b2evolution.net/>
- <http://www.1234.info/webtemplates/>
- <http://www.freeCSStemplates.org/>

Todos ellos son skins planos en HTML, para poder aplicarlos a los CMS es necesario un proceso de adaptación. El objetivo de este proyecto va a consistir en estudiar los pasos necesarios que debemos seguir para poder adaptar un *skin* a un CMS.

3.2.6. XWiki Enterprise

Además de la Plataforma XWiki, el proyecto XWiki ofrece varias aplicaciones construidas sobre él, una de ellas es XWiki Enterprise. Se trata de un wiki con características empresariales como blog, gestión privilegios, autenticación mediante LDAP, exportar a PDF, etc.

También es un wiki de segunda generación que ofrece la capacidad de instalar y desarrollar pequeñas aplicaciones dentro de las páginas wiki. Tiene características de extensibilidad de gran alcance, tales como secuencias de comandos en las páginas, *plugins* y una arquitectura altamente modular.

XWiki Enterprise es perfecto para los siguientes usos:

- Intranets
- Sitios web públicos
- Administración de documentación
- Proyectos de colaboración
- Portales basados en enlace de datos de fuentes externas...

XWiki es fácil de instalar ya que en el propio paquete tenemos el *servlet*, la base de datos y el apache.

Para comenzar a utilizar XWiki es conveniente seguir la guía del usuario. [11]

Xwiki Enterprise está dividido en *Spaces*, *Pages* y *Panles*. Un Panel es un menú que permite diferentes acciones en el wiki de navegación, búsquedas, creación de páginas, estadísticas de visualización. Los paneles contienen generalmente enlaces, estos enlaces se escriben de forma manual o se generan automáticamente mediante secuencias de comandos. Un espacio es un contenedor de páginas (*pages*), en otras palabras es un conjunto de páginas agrupadas bajo un mismo nombre.

3.2.6.1. Space

Como hemos dicho un *Space* es una colección de wiki *pages*. Agrupar las páginas facilita:

- Localizar páginas incluso visualizarlas juntas o formar estructuras en árbol.
- El software permite indizar el contenido del *Space* fácilmente.
- Se pueden ajustar los niveles dentro de los *Spaces* en conjunto con *skins* y los paneles laterales.
- Los permisos de usuario pueden establecerse a nivel de *Space*.
- Limitar las búsquedas dentro del alcance del *Space*. También desde el modo WebDAV.

La página de inicio suele crearse en la URL “.../*SpaceName*/WebHome”. Por ejemplo: “http://localhost:8080/xwiki/Nombre_del_Space/WebHome”

3.2.6.2. Page

Es una unidad básica de contenido, normalmente dividida en 4 áreas:

- Barra de acciones (arriba): Permite interactuar con la página en cuestión.
- Contenido.
- Adjuntos que han sido subidos a dicha página.
- Historial de la Página (cada versión).
- Información del Documento (superiores e inferiores en la jerarquía del *Space*, etiquetas,...)

Las siguientes acciones pueden ser realizadas en una página, en función de los derechos de los usuarios:

- Editar
 - Usando el editor WYSIWYG, que es un editor de texto enriquecido.
 - Usando el editor wiki, para usuarios que manejan la sintaxis XWiki.
- Imprimir y Exportar los contenidos
 - Usando el menú para impresión.
 - Cada Page puede ser exportable a: HTML, PDF, RTF (para MS Office) y XAR (XWiki Archive).
- Visualización
 - El usuario puede añadirla a su *watchlist*.
 - Siempre que haya modificaciones será notificado.

Además los administradores podrán:

- Cambiar los derechos de acceso.
- Editar los objetos y clases de las páginas.
- Renombrar, copiar y borrar las páginas.

3.2.6.3. Paneles

Un Panel es básicamente un marco con enlaces útiles que son colocados formando columnas.

Dos formas de controlarlos:

- Elegir columnas que se muestran (derecha, izquierda, ambas o ninguna).
- Elegir qué paneles están presentes.

Además si eres Administrador del XWiki podrás realizar las siguientes operaciones sobre los paneles:

- Cambiar el logo de XWiki
- Modificar el conjunto de colores.
- Seleccionar el conjunto de paneles que queremos mostrar.
- Crear nuevos usuarios, añadirlos a grupos, darles privilegios.
- Configurar el idioma.
- Configurar la política de acceso (pública, privada).

3.2.7. Drupal

Drupal es un sistema de gestión de contenido modular para sitios Web, dinámico y muy configurable. Es un programa de código abierto, 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.

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. El sitio principal de desarrollo y coordinación de Drupal es <http://www.drupal.org>, en el que participan activamente varios miles de usuarios de todo el mundo.

Antes de instalar Drupal es necesario descargarnos un módulo de Apache [12] para interpretar Php [13] y una base de datos MySQL. [14]. En nuestro caso ya que la aplicación va sobre Windows hemos utilizado la herramienta XAMPP [15], que contiene todos los paquetes necesarios.

Drupal es una plataforma muy flexible que permite multitud de opciones para cambiar el aspecto de su sitio, cómo los usuarios interactúan con él, o los tipos de contenido que se pueden mostrar.

Una vez instalado no requiere mucha configuración inicial para comenzar a trabajar. El administrador puede activar o desactivar diferentes características y establecer configuraciones que cambian el aspecto y funcionalidad del sitio en función de las necesidades.

El objetivo de usar un gestor de contenidos es la abstracción de conceptos de niveles inferiores a los conceptuales de los tipos de datos. Es decir, no es trascendente como funcionen internamente esos contenidos, debe ser transparente para la asimilación del funcionamiento.

3.2.7.1. Gestión de usuarios

Gracias al sistema de privilegios de Drupal es posible crear diferentes tipos de usuarios, cada uno de ellos con un rol.

Los usuarios se pueden registrar e iniciar sesión de forma local o utilizando un sistema de autenticación externo como Jabber, Blogger, LiveJournal u otro sitio Drupal. Para su uso en una intranet, Drupal se puede integrar con un servidor LDAP.

Los administradores de Drupal no tienen que establecer permisos para cada usuario. En lugar de eso, pueden asignar permisos a un 'rol' y agrupar los usuarios por roles.

3.2.7.2. Componentes y Contenidos

Se define componente como una agrupación de objetos físicos o lógicos relacionados con la arquitectura Drupal, de forma que dichos componentes se puedan organizar y validar.

Así mismo podemos incluir en este conjunto los siguientes ítems:

- Componentes físicos (ficheros o estructuras de ficheros):
 - Módulos, temas, *templates*, logos, iconos.
- Componentes lógicos (definidos por la arquitectura Drupal):
 - Módulos (un módulo puede definir “n” submódulos en la jerarquía Drupal).
 - Temas (un tema puede definir varios temas)
- Otros componentes que detallaremos a continuación son :
 - Tipos de usuario , Taxonomías, Bloques, Vistas, Paneles, Menús

En la parte lógica de componente, un contenido sería un componente con información. Más allá de los tipos de datos de usuario, que también pueden ser componentes, disponemos de objetos más complejos con cierta funcionalidad pre-establecida y también la posibilidad de definir nuevos tipos de usuario que serán nuevos componentes reutilizables disponibles en la arquitectura Drupal. Resumiendo, tendríamos estos tipos de componentes:

- Componentes aportados por el *core* de Drupal
- Componentes aportados por módulos de Drupal
- Componentes o tipos de Usuario definidos para la Arquitectura

Hay varias formas de enfocar la clasificación de contenidos/componentes. Se pueden clasificar por usabilidad y se pueden clasificar por origen. Clasificación por usabilidad:

- Tipos de Usuario: Tipos de datos (componentes) definidos por el usuario.
- Taxonomías: Tipo de datos (categoría) definido por el usuario.
- Vistas: Modelo de presentación de datos.
- Paneles: Modelo de estructuración de la presentación de datos.
- *Templates*: Modelo de presentación de contenidos
- Etc.

Clasificación por origen:

- Contenidos auto creados: Creados automáticamente al instalar un módulo en el sistema, bien haya sido del *core* o bien una nueva contribución.
- Contenidos definidos por el usuario: Estos contenidos los define el usuario ajustando los campos que necesita para la funcionalidad establecida.

El *core* de Drupal es muy pobre en la definición de nuevos tipos de datos (tipos de usuario). Una primera visión de los componentes que nos ofrece el producto sería:

- Tipos de usuario: Como los tipos de datos de cualquier lenguaje pero con mayor complejidad. Muy cercano al concepto de “objeto”.
- Categorías: Tipo de datos enfocado a clasificación de contenidos.
- Foros: Aportado por el módulo fórum. La versión que aporta este módulo es muy pobre comparada con otras que tenemos en el mercado como phpBB.
- Apuestas: Aportado por el módulo *poll*.
- Bloques: Son una forma de presentación rápida y escueta de información. Lo aporta el módulo *block*.
- Comentarios: Funcionalidad de comentar contenidos expuestos. Aportado por el módulo *comment*.

Sin embargo si permite una rica definición de categorías. Las categorías forman parte de la estructura jerárquica de Drupal y son una de las claves para la localización y estructuración de contenidos.

- Vistas → Aportado por el módulo *views*
- FAQ → Aportado por el módulo *faq*
- Paneles → Derivado del módulo *panels*
- Servicios → Ofrecido por el módulo *services*
- Etc.

3.2.7.3. Módulos Oficiales / Contribuciones

Para personalizar y extender nuestro sitio web podemos ir añadiendo módulos y temas a nuestro sitio. Los módulos serían el equivalente a las clases java, es decir el código, que proporciona funcionalidad.

Un módulo es una funcionalidad nueva que aporta comportamiento nuevo, pero ese comportamiento puede afectar a muchas partes de la arquitectura de Drupal. Algunos módulos vienen con la instalación del producto, son los llamados del núcleo, *core* u oficiales. Además existen módulos “contribuidos” que son desarrollados, probados y

certificados por la comunidad Drupal, los cuales pueden ser descargados del sitio Web. [16]

Este gran abanico de posibilidades que nos ofrece el producto varía según la versión del producto que estemos utilizando. Actualmente disponemos de 4 versiones principales:

4.x, muy antigua con muy poca usabilidad.

5.x, la más extendida. La que dispone de más módulos y ha sido la base de la mayoría de las funcionalidades aportadas por Drupal.

6.x, la versión que se usa en la actualidad debido a que incorpora muchos módulos nuevos que usan Flash o que mejoran la visibilidad de los contenidos de Drupal. El problema es que hay muchos módulos que no están aún en esta versión.

7.x, última versión de Drupal. Está en desarrollo pero ya dispone de módulos que se podrían certificar. Está orientada a objetos, sólo funciona con php 5 y mejora mucho el rendimiento del producto.

Podemos actualizar los módulos dentro de una misma versión. A esta sección accedemos con: “admin/logs/updates”. En ella podemos ver el estado de los módulos utilizados. Podemos ver además si hay nuevas versiones disponibles y a su vez si esta nueva versión es un parche de seguridad. En la Figura 3.7 vemos un ejemplo de actualización de módulos:

Available updates		
<input type="button" value="Lista"/> <input type="button" value="Opciones"/> <input type="button" value="Upgrade status"/>		
Here you can find information about available updates for your installed modules. Note that each module is part of a "project", which may have the same name as the module or may have a different name.		
Last checked: 50 mins 3 secs ago (Check manually)		
Drupal 5.19 Security update required! ❌		
Security update:	5.20 (2009-Sep-16)	Download · Release notes
Includes: <i>Comment, User, Help, Menu, Node, Book, Block, Watchdog, Poll, Color, Taxonomy, Drupal, Ping, Profile, Locale, Aggregator, Statistics, Upload, Forum, Filter, Contact, Search, Tracker, Path, System</i>		
Advanced Poll 5.x-1.0-beta6		Actualizado ✓
Includes: <i>Advanced Poll, Advanced Poll Converter</i>		
Automatic Nodetitles 5.x-1.2		Update available ❌
Recommended version:	5.x-1.3 (2009-Jul-31)	Download · Release notes
Includes: <i>Automatic Nodetitles</i>		
CAPTCHA 5.x-3.2		Actualizado ✓
Includes: <i>CAPTCHA, Image CAPTCHA, Text CAPTCHA</i>		
Content Construction Kit (CCK) 5.x-1.10		Actualizado ✓

Figura 3.7: Administración de Módulos de Drupal

3.2.7.4. Configuración de los temas

Cuando hablamos de temas en Drupal nos referimos al conjunto de configuraciones y ejecuciones que determinan la presentación general del portal. Los temas reflejan la apariencia del portal. En muchos otros productos se llaman también plantillas, pero en Drupal ese término ya está reservado a las “*contemplates*”, es por ello que en este producto “los temas no son plantillas”, se les conoce con *Themes*.

En la Figura 3.8 vemos un ejemplo de página de administración de temas de Drupal:

The screenshot shows the Drupal theme administration interface. At the top, there are tabs for 'Temas', 'Lista', and 'Configurar'. A yellow warning box contains the text: "No hay información disponible sobre nuevas versiones de los módulos y temas gráficos instalados. Para comprobar si las hay, debe ejecutar cron o puede comprobarlo manualmente. Tenga en cuenta que comprobar la existencia de actualizaciones puede llevar tomar mucho tiempo. Sea paciente, por favor." Below the warning, there is instructional text: "Seleccione qué temas gráficos están disponibles para sus usuarios y especifique el tema gráfico predeterminado. Para configurar opciones de despliegue a nivel de sitio, haga clic arriba en la tarea «configurar». Alternativamente, para sobrescribir estas opciones en un tema gráfico específico, haga clic en el enlace «configurar» para ese tema gráfico. Note que temas gráficos distintos pueden tener disponibles distintas regiones para desplegar contenido; para ofrecer consistencia en la presentación, quizás desee activar un solo tema gráfico." and "Para cambiar la apariencia de su sitio, tiene a su disposición numerosos temas gráficos de terceros." and "Vea la página de actualizaciones disponibles para información sobre los módulos y temas gráficos instalados que tienen nuevas versiones." Below this is a table of themes:

Captura de pantalla	Nombre	Versión	Activo	Predeterminado	Operaciones
	Bluemarine Tema gráfico multicolumnas, basado en tablas, con un esquema de colores marinos y ceniza.	6.12	<input type="checkbox"/>	<input type="radio"/>	
	Chameleon Tema minimalista con tablas y colores claros.	6.12	<input type="checkbox"/>	<input type="radio"/>	
	ficod ficod.	6.x-1.0	<input checked="" type="checkbox"/>	<input type="radio"/>	configurar

Figura 3.8: Administración de Themes de Drupal

Al igual que en los módulos, existen temas oficiales y temas contribuidos. Drupal por defecto te aporta 5 *themes*:

- Garland (y Minnelli): Temas óptimos para la administración del portal. No son recomendables para presentación.
- Blue Marine, Chamaleon, Marvin, Pushbutton son los primeros temas que podemos abordar para la presentación de portales.

Otra opción de presentar un tema es o bien crearlo desde cero, partiendo de plantillas estándar como la de Zen, o bien modificando uno de los existentes. El caso que nos ocupa es la aplicación de *skins* independientes HTML a Drupal.

Construir un tema desde cero tiene una complejidad elevada. Cuando hacemos referencia a “desde cero” queremos decir, desde un archivo de texto en blanco.

Drupal posee cientos de reglas y mecanismos que son difíciles de comprender en su totalidad, y que son necesarios para crear un tema desde cero. Algunas personas decidieron construir temas de propósito general. La mayoría de ellos se caracterizan por, ser poco atractivos visualmente, y ser muy fáciles de entender por los diseñadores.

Las plantillas en Drupal son matrices que definen como estará organizado el contenido del sitio. Por ejemplo, la plantilla *page.tpl.php* se encarga de definir la estructura HTML de cualquier página del sitio.

En los primeros años de Internet, las páginas webs se caracterizaban por estar constituidas por una inmensa cantidad de archivos .html que definían la estructura de la web como podemos ver en la Figura 3.9.

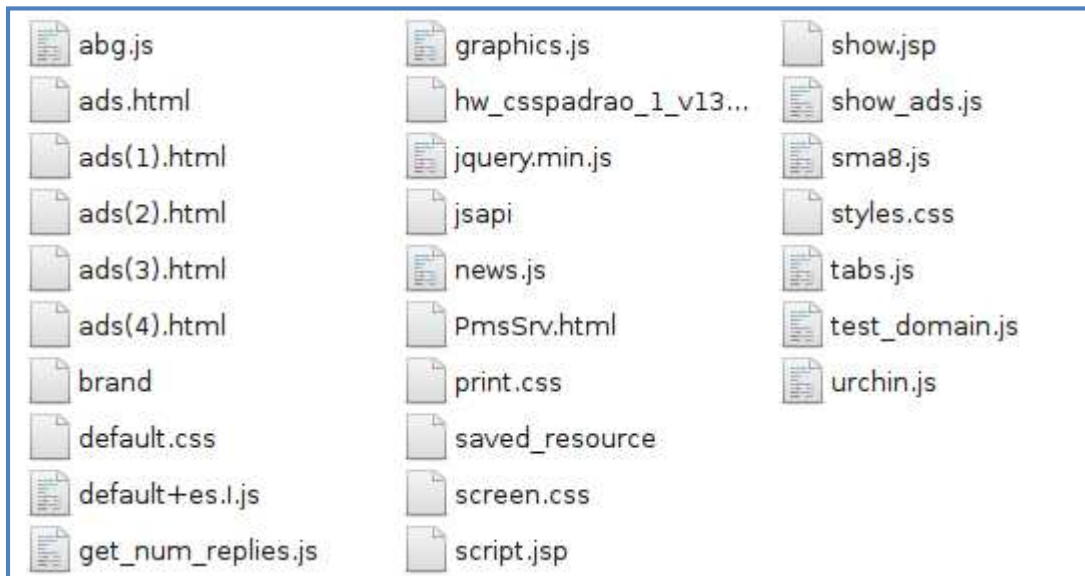


Figura 3.9: Estructura originaria de archivos de un sitio web en HTML.

Todos esos archivos HTML son los que definían las páginas de Inicio, Contáctenos, Quiénes somos, etc. Con la adopción de los administradores de contenidos todas estas páginas ahora son guardadas en una base de datos, Drupal busca el contenido dependiendo de la página que deseamos visitar y la muestra, está claro que no puede mostrar todos esos textos como algo plano y sin formato, para organizar la estructura del contenido hace uso de las plantillas.

Gracias a las plantillas con un único archivo **.tpl.php* se le puede dar estilo a muchas páginas HTML. Volviendo a la plantilla *page.tpl.php*, en ella se definen todos los *divs* que se estructurarán con **.css* para darle el formato al encabezado, al pie de página, a la barra lateral izquierda, derecha, etc.

Por otro lado los archivos CSS son los que definen que estilo tomará cada valor de la plantilla. El lenguaje CSS (*Cascade Style Sheets* - hojas de estilo en cascada) es uno de los métodos más utilizados en los sitios webs para definir los aspectos visuales. También mencionamos el archivo **.info*, este archivo se encarga de definir una serie de aspectos entre los que se destacan las regiones del tema, y los archivos de CSS.

Las regiones son la forma en que Drupal define dónde se podrá colocar el contenido. Por ejemplo, si alguien quisiera colocar en el pie de página el contenido “Este sitio está construido con Drupal” elegiría la región *Footer*, que se dibuja generalmente en la parte inferior del sitio. ¿Y cómo se posiciona en el inferior del sitio? Porque así lo indica el HTML y la CSS del sitio.

Las regiones se pueden observar a la hora de posicionar los bloques en el sitio, en la página de administración de los bloques las resalta de la forma en la que mostramos en la Figura 3.10.



Figura 3.10: Manejo de los Bloques en Drupal.

Las regiones generalmente se dibujan en las plantillas *page.tpl.php* pero otras plantillas como *node.tpl.php* pueden también incluirlas. El archivo *template.php* define una serie de funciones que se encargan de procesar y dejar disponibles para su uso en las plantillas las variables que mencionamos antes.

3.2.7.5. Descarga de Themes independientes en Drupal

El primer sitio para buscar *theme* o plantillas para Drupal es el propio proyecto oficial, en su sección de *themes*. Aquí podemos encontrar cientos de temas categorizadas por compatibilidad con la versión de Drupal, 4.x, 5.x, 6.x. La mayoría tienen una imagen previa para ver el aspecto que tiene el *skin* antes de descargarlos y algunos podemos incluso probarlos en una demo.

Para instalarlos basta con descomprimirlos y ubicarlos en el directorio */theme* del servidor. Si la que hemos elegido presenta algún problema, siempre podemos solicitar soporte al dueño del proyecto o incluso que incluya alguna característica nueva.

A continuación presentamos una lista de recursos donde obtener *themes* gratuitos o comerciales.

Podemos descargarnos *themes* gratuitos desde los siguientes sitios:

- Proyecto Oficial: el primer sitio donde mirar, gran variedad, *screenshots*, demos y soporte.
- *Themebot*: cientos de *themes*, algunos están en el proyecto oficial, otros no, la mayoría para Drupal 5.
- DrupalLab: ofrecen de forma gratuita el *theme* de su página, para Drupal 6.
- FreeCMSTemplates: ofrecen cinco *themes* diferentes, cada una con cinco variaciones de color.
- Drupal2u: más de 200 *themes*, probablemente podáis encontrar la mayoría en el proyecto oficial.
- DrupalSix: ofrecen descargas para Drupal 6.x.
- Drupal 6 *Theme*: Ofrecen *themes* para la versión 6.x

Podemos obtener *themes* comerciales en las siguientes web:

- *Themenap*: ofrecen un *theme* "eficiente para blogs y páginas de noticias", para Drupal 5.x y 6.x.
- *Alldrupaltheme*: *themes* gratuitas para Drupal 5. Tienen una específica para amantes del WoW.
- GoDrupal: más de 100 *themes*, aunque en realidad son variaciones de colores de unas 15 diferentes.
- *TopNotchTheme*: sitio especializado en Drupal que ofrece en torno a 10 *themes* de altísima calidad.
- *Templatemonster*: acaban de incorporar Drupal a su catálogo de *themes* en el que ofrecen también para Mambo, Joomla, Wordpress...
- *Deonixdesign*: en torno a 30 *themes* para Drupal con descarga solo para miembros.

3.3. Herramientas

3.3.1. Mozilla

El proyecto Mozilla, cuyo logotipo original podemos ver en la Figura 3.11, nació en el año 1998, fruto de la liberación del código fuente de la serie 4.x de Netscape. Actualmente Mozilla es una organización sin ánimo de lucro para permitir la continuidad del proyecto más allá de la participación de voluntarios individuales. [17]



Figura 3.11: Logo Mozilla

Originalmente el producto más importante del proyecto era la suite de aplicaciones Mozilla, ahora el futuro del proyecto Mozilla se encuentra en componentes separados: Mozilla Firefox (Navegador), Mozilla Thunderbird (Cliente de Correo y Lector de Noticias), Mozilla Sunbird (Calendario) o Mozilla Nvu (Editor Web) entre otros.

Lejos de ser sólo un navegador, Mozilla es una plataforma de desarrollo multiplataforma sobre la que se pueden construir otras aplicaciones. Las aplicaciones basadas en Mozilla se pueden ejecutar en cualquier ordenador que tenga instalado Mozilla independientemente del sistema operativo que se usa. Esto es posible porque Mozilla sirve como "intermediario" entre el sistema operativo y la aplicación.

El navegador Mozilla Firefox (Figura 3.12) es un navegador de Internet libre y de código abierto, desarrollado por la Corporación Mozilla, la Fundación Mozilla y un gran número de voluntarios externos.



Figura 3.12: Logo Firefox

Firefox es un navegador multiplataforma (Figura 3.13) y está disponible en varias versiones de Microsoft Windows, Mac OS X, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre, publicado bajo una triple licencia GPL/LGPL/MPL. Es el segundo navegador más popular en todo el mundo, después de Internet Explorer.

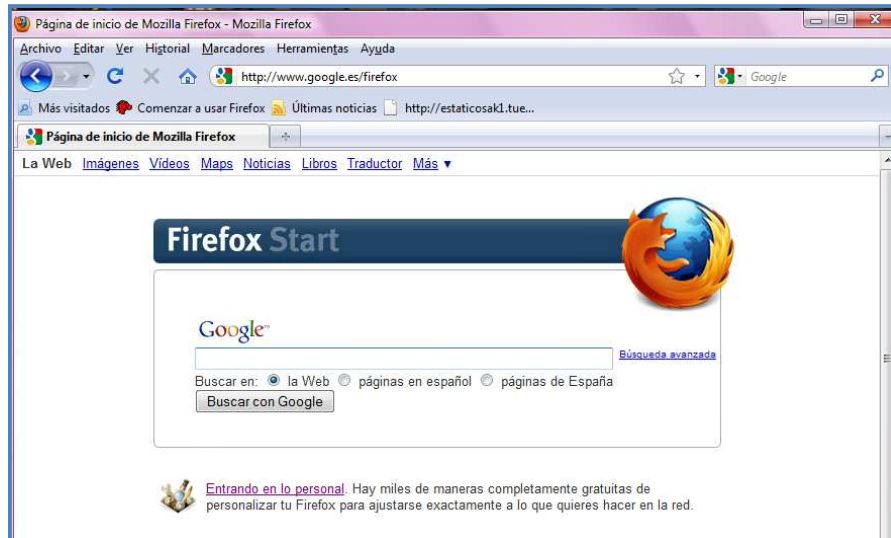


Figura 3.13: Navegador Firefox

3.3.2. Firebug

Firebug (Figura 3.14) es una extensión para Firefox bastante útil sobre todo si eres desarrollador de sitios web, esta extensión nos permite ver el funcionamiento interno de los sitios webs, inspeccionar todo el código y ver en tiempo real las llamadas "xml http request", además puedes encontrar y depurar los errores en CSS, html, javascript y XMLHttpRequest, podemos ver un ejemplo de su funcionamiento en la Figura 3.15.[18]



Figura 3.14: Logo Firebug

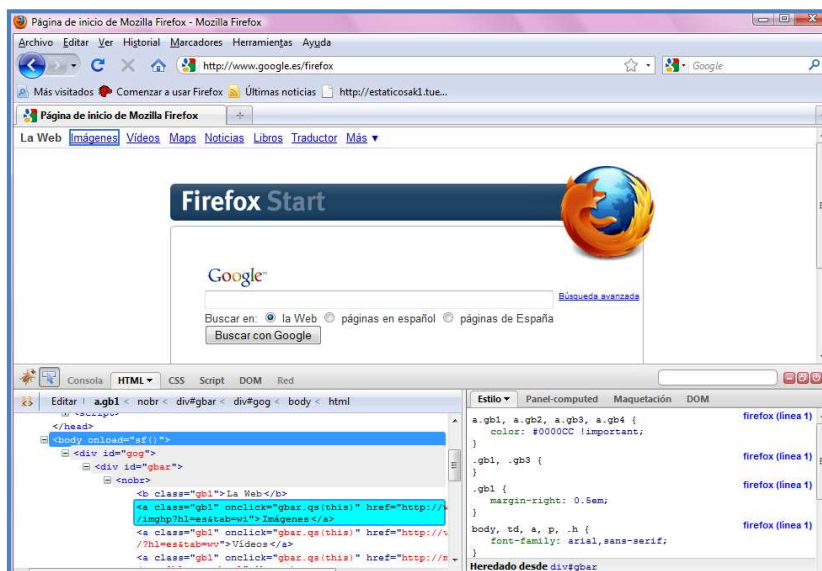


Figura 3.15: Ejemplo de utilización de Firebug

3.3.3. FCKEditor

FCK Editor es un editor de uso gratuito que puede ser embebido en aplicaciones web o sitios web para proveer a los usuarios la posibilidad de editar texto enriquecido y obtener internamente el resultado como HTML. Vemos en la figura 3.16 la consola de utilización de esta herramienta. Como sus autores lo llaman: “El Editor de Texto para Internet”. FCKEditor no requiere de ninguna instalación especial. [19]

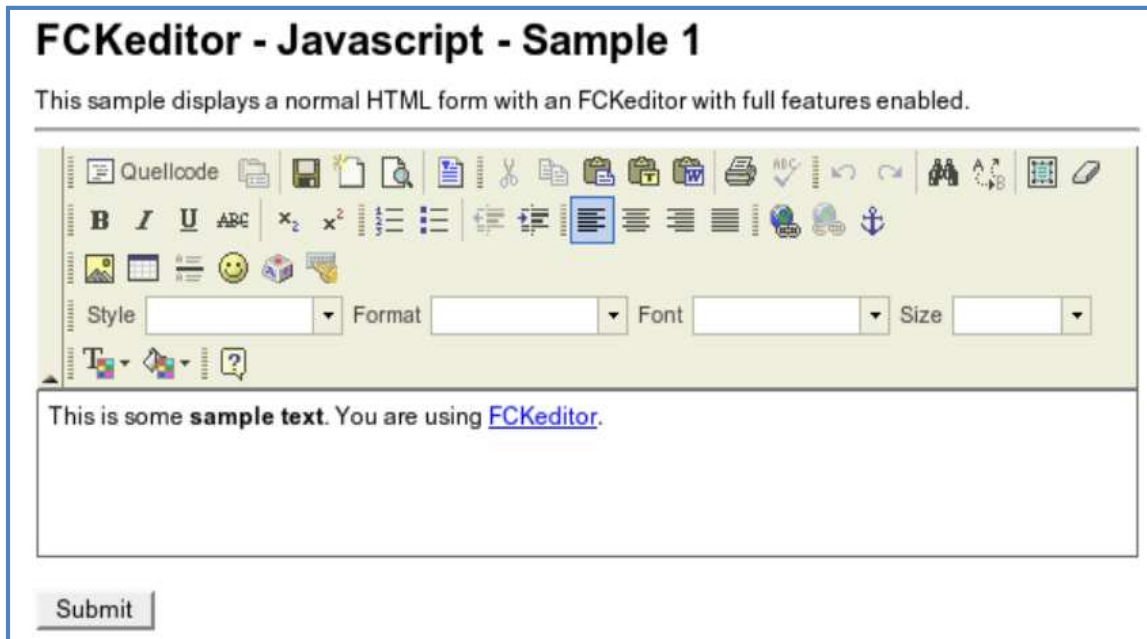


Figura 3.16: Panel FCKEditor

Se trata de un editor para HTML que se incrusta en páginas web o aplicaciones web, y que permite brindar al usuario (sin necesidad de conocimientos de HTML) una interfaz fácil de utilizar con la cual se pueden crear páginas o contenido HTML.

Hemos utilizado FCKEditor para pasar páginas HTML a Drupal sin necesidad de modificar el código fuente. Podríamos utilizar cualquier otro editor de texto como Notepad (Windows), TextEdit (Mac) o HTML Kit.

4. Descripción Informática

4.1. Especificación de requisitos

El objeto de la especificación de requisitos es definir de manera clara y precisa todas las funcionalidades y restricciones del software que se plantea utilizar. El documento va dirigido tanto a los desarrolladores, como a los responsables de evaluación del mismo y a los posibles usuarios finales.

El tratamiento de los requisitos es el proceso mediante el cual se especifican y validan los servicios que debe proporcionar el sistema así como las restricciones en las que debe operar.

En este proyecto hemos utilizado software distintos, sin embargo los requisitos funcionales serán los mismos, ya que el objetivo que queremos cumplir y las restricciones son las mismas en ambos casos.

4.1.1. Requisitos Funcionales

A continuación se nombran los requisitos específicos funcionales para la aplicación de un *skin* sobre un CMS:

- **REQF.1.** El aspecto de nuestro sitio web debe ser igual a la plantilla que hemos utilizado.
- **REQF.2.** Los menús deben tener las mismas funciones que presentaban antes de aplicarles la plantilla.
- **REQF.3.** Los paneles laterales deben presentar la misma estructura que la propuesta en la plantilla.
- **REQF.4.** La funcionalidad del sitio web debe permanecer intacta.
- **REQF.5.** El diseño de letra debe ser el mismo que en la plantilla.
- **REQF.6.** La plantilla debe poder aplicarse de forma independiente sobre las distintas páginas del portal.
- **REQF.7.** Las cabeceras y pie de páginas deben presentar en mismo diseño que en la plantilla.
- **REQF.8.** La plantilla adaptada al CMS podrá ser exportada y utilizada en cualquier otro sitio web desarrollado con el mismo gestor de contenidos.

4.1.2. Requisitos no funcionales

Los requisitos no funcionales, son aquellos que describen las facilidades que debe proporcionar el sistema en cuanto a:

- **REQNF1: Facilidad de uso**

La plantilla podrá ser adaptada de forma sencilla a cualquier gestor de contenidos siguiendo los pasos detallados en esta memoria.

- **REQNF2: Mantenibilidad**

El mantenimiento del producto está garantizado mientras el gestor de contenidos para el que ha sido adaptado permanezca activo.

- **REQNF3: Fiabilidad**

No existen atributos del sistema para este apartado.

- **REQNF4: Seguridad**

No existen atributos del sistema para este apartado.

- **REQNF5: Portabilidad**

Gracias a que se han usado gestores de contenidos de uso libre, el software puede ser portado a cualquier otro sitio web desarrollado con el mismo gestor.

- **REQNF6: Tiempo de respuesta**

Dependerá del gestor de contenido utilizado y de la capacidad de la máquina sobre la que trabajemos.

4.2. XWiki

4.2.1. Diseño

Los *skin* de XWiki son utilizados para personalizar la interfaz de una instancia de XWiki Enterprise. Es posible personalizar el aspecto de nuestro XWiki, bien editando los ficheros de configuración que por defecto trae el XWiki, o bien creando un nuevo *skin* y meterlo en nuestro XWiki.

Un *skin* es un conjunto de imágenes, ficheros CSS y plantillas que permiten cambiar el diseño de la wiki. Suele ser desarrollado por los administradores y diseñadores gráficos y se puede utilizar para cambiar el aspecto de una instalación de XWiki para:

- una página
- un usuario
- un espacio
- todo el wiki.

El *skin* puede tener varias hojas de estilo que son conmutables por los usuarios. Los componentes que definen el *Skin* de XWiki son:

- Plantillas de Velocity.
- Hojas de estilo CSS.
- Ficheros Javascript.
- Imágenes.

Las principales partes en las que se divide un *skin* en XWiki son:

- **header.vm:** Se encarga de pintar la cabecera del XWiki.
- **footer.vm:** Genera el pie de página.
- **view.vm:** `contenview.vm → shutcuts.vm → viever.vm → docextra.vm → endpage.vm`
- **style.css:** Se encarga de importar los estilos.
Ejemplo: `@import "hoja_de_estilos.CSS";`
- **stylesheets.vm:** Genera los estilos.
- **contentview.vm:** Es el fichero que carga el contenido de la página del wiki.
- **hierarchy.vm:** Recoge en la cabecera del xwiki las páginas por las que hemos pasado hasta llegar a la principal, es decir, las páginas padre asociadas a la actual.
Ejemplo: `Principal → Menu Inicio → submenú_1 → submenú_2`

En el siguiente ejemplo (Figura 4.1) mostramos un fragmento de código de una de estas páginas de XWiki, donde podemos ver cómo recupera los campos que luego mostrará por pantalla.

```
#set($parents = $util.arrayList)
#set($discard = $parents.add($doc.fullName))
#macro(computeHierarchy $doc $result $level)
  #set($parent = $doc.parent)
  #if(($parent != '') && ($level < 6) &&
    (!$parents.contains($parent)))
    #set($discard = $parents.add($parent))
    #set($pdoc = $xwiki.getDocument($parent).getTranslatedDocument())
    #set($pdocurl = $pdoc.getURL("view"))
    #set($nstring = "<a
href='$pdocurl'>$xwiki.getXMLEncoded(${pdoc.displayTitle})</a> <span
class='separator'>&raquo;</span> $result")
    #set($level = $level + 1)
    #computeHierarchy($pdoc $nstring $level)
  #else
    $result
  #end
#end
#if($context.getMode()==0) ## Visible only in a page
  <div id="hierarchy">
    <span class="currentSpace"><a
href="$SpaceViewUrl">$xwiki.getXMLEncoded(${doc.Space})</a>: </span>
#computeHierarchy($doc "<a
href='${doc.getURL('view')}'>$xwiki.getXMLEncoded($doc.displayTitle)
</a>" 0)
    </div>
#end
```

Figura 4.1: Código Velocity de una página XWiki

4.2.2. Implementación

Hemos creado varios contenidos con las páginas del grupo de investigación GAVAB de la Universidad Rey Juan Carlos, para utilizarlos como ejemplo tanto en el CMS XWiki como en el CMS Drupal.

En la Figura 4.2 vemos algunos de los ejemplos de las páginas del GAVAB pasadas a XWiki con el *skin* “colibri” que es el que se carga por defecto en la instalación:

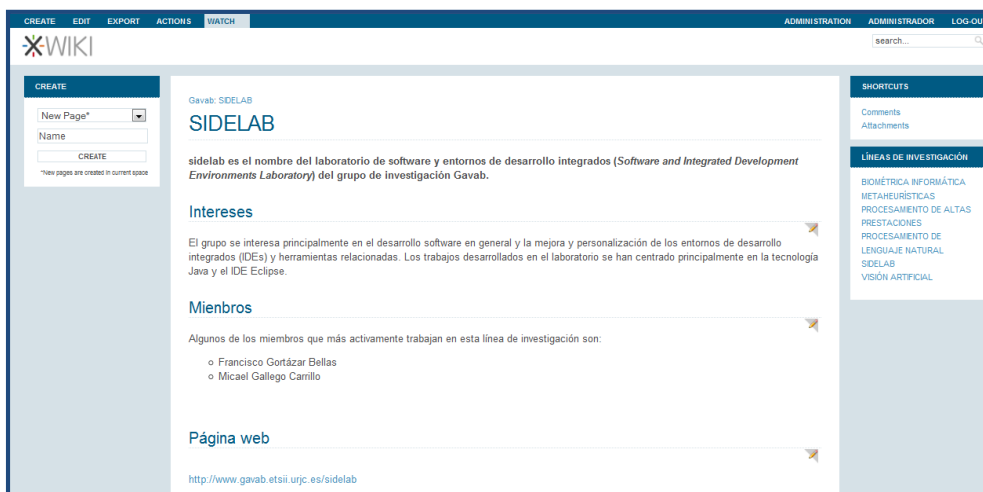


Figura 4.2: Páginas de Gavab creadas con Xwiki

4.2.2.1. Aplicar un *skin* a una página XWiki

Para modificar un *skin* en XWiki es necesario editar los ficheros de configuración encargados de dar el aspecto al XWiki. Existen unos ficheros que por defecto se crean al instalar XWiki, se encuentran en la ruta: “XWiki_version/webapps/xwiki/skins”

Los *skins* que vienen por defecto en XWiki son tres, y se encuentran dentro de los siguientes directorios:

- colibri
- tucan
- albatross

Para comenzar a trabajar, lo mejor es copiar y pegar el contenido original del *skin*, dejando una copia de seguridad y hacer pequeñas modificaciones sobre el mismo. Para poder aplicar fácilmente un *skin* a una página sin necesidad de realizar modificaciones desde la administración, simplemente colocamos al final de la URL de una de las páginas de nuestro sitio Xwiki, el valor “skin=Nombre_Skin”, de la siguiente forma: “http://localhost:8080/xwiki/bin/view/Gavab/?skin=nombre_skin”.

Podemos aplicarlo a todo el Wiki o a un *Space* mediante la modificación del campo “*skin*” en la XWiki o “Web Preferences”. También hay que modificar el valor predeterminado y los estilos alternativos (Si no hemos creado estilos alternativos, colocamos 'style.css' en ambos campos).

Las preferencias globales son controladas por los elementos del *skin*. En el link de Administración basta con sustituir XWiki.DefaultSkin con el nombre del *skin* que desea utilizar. Si su *skin* está en su directorio “/skins”, simplemente escribe su nombre en el campo (“tucan” o “albatros”, por ejemplo.)

Este parámetro lo definimos en el archivo “xwiki.cfg”, se encuentra en la ruta “xwiki_version/webapps/xwiki/WEB-INF/xwiki.cfg”, lo define bajo la propiedad “xwiki.defaultskin”. Si no especificamos ningún *skin* en la propiedad “xwiki.defaultskin”, cogerá por defecto el *skin* definido en “xwiki.defaultbaseskin”.

4.2.2.2. Aplicar *skin* HTML a XWiki

Hemos elegido una plantilla libre de la web <http://www.1234.info/webtemplates/> llamada “Multifex312”, para aplicarla sobre nuestro CMS de XWiki (más adelante haremos lo mismo sobre el CMS Drupal). Se trata de una plantilla HTML, con contenido de ejemplo para mostrarlo por pantalla y con varios CSS diferentes dentro de la plantilla. Mostramos un fragmento de código HTML de esta plantilla en la Figura 4.3, donde podemos ver el código plano insertado para mostrar contenido por pantalla.

```

<div class="round-border-topright"></div>
<h1 class="first">Navigation Title</h1>
<!-- Navigation with grid style -->
<dl class="nav3-grid">
<dt><a href="#">Navlink 11</a></dt>
<dt><a href="#">Navlink 12</a></dt>
<dd><a href="#">Navlink 121</a></dd>
<dd><a href="#">Navlink 122</a></dd>
<dd><a href="#">Navlink 123</a></dd>
<dt><a href="#">Navlink 13</a></dt>
<dt><a href="#">Navlink 14</a></dt>
<dt><a href="#">Navlink 15</a></dt>
</dl>
<!-- Template infos -->
<h1>Multiflex Templates</h1>
<p>A design series for those who want a template that can be used
for almost any situation. It is recommended to have at least
beginners knowledge of XHTML/CSS to work with this template in a
satisfying way.</p>
<h3>Multiflex-1</h3>
<p>Released: 15.05.2006<br />OK for operational use, but has
heavy code.<br /><a
href="http://www.1234.info/webtemplates/multiflex1/">Download latest
update</a></p>

```

Figura 4.3: Fragmento de código HTML del skin Multiflex

En la Figura 4.4 podemos ver el aspecto visual en un navegador de esta plantilla Multiflex312 que hemos elegido para adaptar a los CMS de XWiki y Drupal.

Figura 4.4: Skin Multiflex312

Para comenzar necesitamos crear una nueva carpeta en el mismo directorio donde hemos comentado que se encuentran los *skin* genéricos (albatros, colibri,...). Ejemplo: “...\XWiki_version\webapps\xwiki\skins\multiflex”, y copiamos dentro todos los fuentes de nuestra plantilla. A partir de aquí iremos analizando qué ficheros generan cada una de las partes del *skin* original para poder adaptar nuestra plantilla Multiflex312 a XWiki.

Si sólo se quieren modificar algunos aspectos del diseño básico de XWiki, será necesario modificar los siguientes archivos:

- El archivo principal de estilos: `style.css`
- La plantilla de encabezado: `header.vm`
- El pie de página de plantilla: `footer.vm`
- El logo principal.

Para ello es tendremos que comenzar por:

- Crear una página vacía.
- Adjuntar las imágenes necesarias en el *skin* a esta página.
- Editar el Objeto en el menú de edición.
- Añadir los objetos de la clase "XWiki.XWikiSkins"
- Modificar el `style.css`
- Modificar los archivos `header.vm` y `footer.vm`.

Para exponer la forma en la que hemos actuado para adaptar el *skin* HTML sobre el CMS XWiki, tomaremos como ejemplo algunas de las partes sobre las que hemos trabajado.

En primer lugar vamos a presentar el *Breadcrumb* (hilo de Ariadna), se trata de una técnica de navegación que recupera y muestra por pantalla el camino que hemos recorrido hasta llegar a la página en la que nos encontramos. Tienen un aspecto similar al siguiente: Página→Sección→Subsección.

El archivo “`hierarchy.vm`”, que se encarga de recoger el *breadcrumb* en la cabecera del XWiki.

Veamos como Multiflex muestra la información del *breadcrumb* en la Figura 4.5, separando los nombres de las páginas mediante flechas.

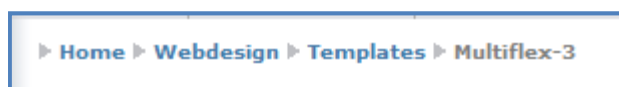


Figura 4.5: *Breadcrumb* en Multiflex312

XWiki muestra el *breadcrumb* como vemos a continuación en la Figura 4.6, con el *skin* “colibrí”:



Figura 4.6 Breadcrumb en XWiki con el *skin* colibrí.

Adaptamos el *skin* Multiflex para que nuestro XWiki muestre la información de la misma manera. Para ello, analizamos en el código fuente del XWiki cómo recuperamos esa información y de dónde se obtienen los estilos. En la Figura 4.7 mostramos un fragmento del código fuente del archivo *hierarchy.vm*:

```
#if(($parent != '') && ($level < 6) && (!$parents.contains($parent)))
  #set($discard = $parents.add($parent))
  #set($pdoc=
  $xwiki.getDocument($parent).getTranslatedDocument()
  #set($pdocurl = $pdoc.getURL("view"))
  #set($nstring = "<a
href='$pdocurl'>$xwiki.getXMLEncoded(${pdoc.displayTitle})</a> <span
class='separator'>&raquo;</span> $result")
  #set($level = $level + 1)
  #computeHierarchy($pdoc $nstring $level)
#else
  $result
#end
#end
#end
#if($context.getMode()==0) ## Visible only in a page
  <div id="hierarchy">
    <span class="currentSpace"><a
href="$SpaceViewUrl">$xwiki.getXMLEncoded(${doc.Space})</a>: </span>
    #computeHierarchy($doc "<a
href='${doc.getURL('view')}'>$xwiki.getXMLEncoded($doc.displayTitle)<
/a>" 0) </div>
#end
```

Figura 4.7: Contenido del archivo *hierarchy.vm*

En la figura 4.8 podemos ver el código HTML de Multiflex encargado de mostrar esa parte:

```
<!--Breadcrumbs -->
<div class="header-breadcrumbs">
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Webdesign</a></li>
    <li><a href="#">Templates</a></li>
    <li>Multiflex-3</li>
  </ul>
</div>
```

Figura 4.8: Código HTML de Multiflex

En la Figura 4.9 podemos ver la adaptación que hicimos en el código fuente de XWiki con el código HTML del Multiflex para poder adaptar el *skin*, fue la siguiente:

```

#ife($parent != '') && ($level < 6) && (!$parents.contains($parent))
  #set($discard = $parents.add($parent))
  #set($pdoc =
  $xwiki.getDocument($parent).getTranslatedDocument()
  #set($pdocurl = $pdoc.getURL("view"))
  #set($nstring = "<li><a
href='$pdocurl'>$xwiki.getXMLEncoded($pdoc.displayTitle)</a></li>
$result")
  #set($level = $level + 1)
  #computeHierarchy($pdoc $nstring $level)
#else
  $result
#end
#end
#ife($context.getMode()==0) ## Visible only in a page
  <ul>
  <li><a
href="$SpaceViewUrl">$xwiki.getXMLEncoded(${doc.Space})</a></li>
  #computeHierarchy($doc"<li>
  <a
href='${doc.getURL('view')}'>$xwiki.getXMLEncoded($doc.displayTitle)
</a></li>" 0)
  </ul>
#end

```

Figura 4.9: Código XWiki con el skin de Multiflex integrado.

La imagen que se utiliza como separador debe estar en la ruta donde la buscará el CSS. El resultado de las modificaciones realizadas podemos verlo en la Figura 4.10:

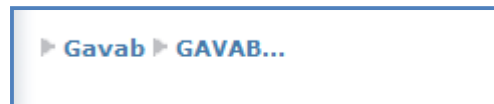


Figura 4.10: Resultado de la integración del skin en la cabecera.

En segundo lugar vamos a exponer los pasos realizados para presentar uno de los menús que aparecen en el skin Multiflex. Tomaremos en menú de la parte lateral izquierda, recordamos como los muestra Multiflex en la Figura 4.11.

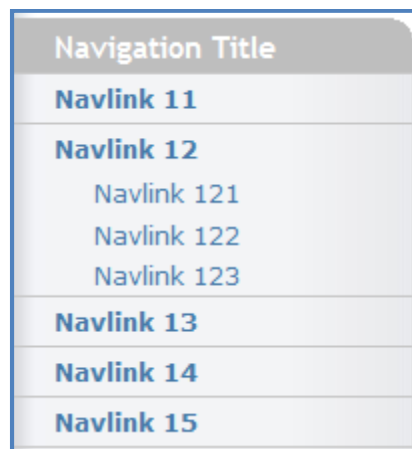


Figura 4.11: Menú lateral izquierdo del skin Multiflex312

Ahora veamos como lo genera XWiki con el *skin* colibrí en la Figura 4.12:

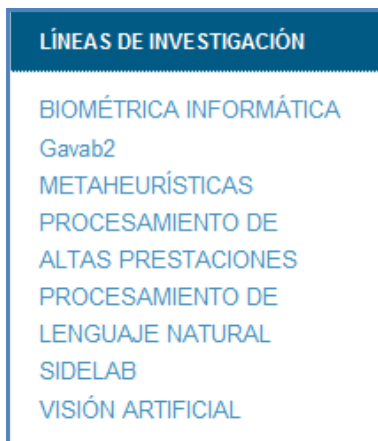


Figura 4.12: Menú lateral generado con el *skin* colibrí

Hemos modificado el archivo *view.vm* que recoge los parámetros, de tal forma que muestre la información de la misma forma que el *skin* Multiflex. Vemos parte del código insertado en este archivo en la figura 4.13:

```
#set ($query = "where (doc.parent is null or doc.parent='' or
doc.parent='${doc.space}.WebHome'
or doc.parent='xwiki:${doc.space}.WebHome') and
doc.space='${doc.space}' and (doc.name <> 'WebHome')
and (doc.name <> 'WebPreferences') order by doc.name asc")
#foreach ($item in $xwiki.searchDocuments("${query}"))
#if ($xwiki.hasAccessLevel("view", $context.user,
"${context.database}:${item}"))
#set($bentrydoc = $xwiki.getDocument($item))
<dt><a href='${xwiki.getDocument("${bentrydoc.fullName}").externalURL}'>
${bentrydoc.displayTitle}</a></dt>
#end
```

Figura 4.13: Código Velocity adaptado al *skin*

A continuación, el resultado final de la adaptación del menú en la Figura 4.14:

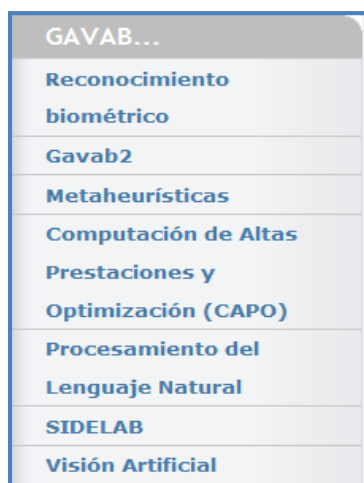


Figura 4.14: Resultado del menú XWiki adaptado al *skin* Multiflex.

Como podéis observar hemos modificado el código Velocity para generar el menú con el mismo formato exacto que el *skin* Multiflex inicial con código HTML. Este es el mecanismo que hemos seguido para adaptar el resto de menús al *skin* deseado.

En algunos casos ha sido necesario modificar varios ficheros distintos para conseguir mantener la funcionalidad del sitio web. También han sido modificadas CSS y rutas con imágenes para completar el proceso de integración.

4.2.3. Pruebas

Hemos contado sólo un par de ejemplos de los múltiples archivos que han sido necesarios modificar para poder aplicar el *skin* de forma total, y conseguir que el sitio web tenga el aspecto que aparece en la plantilla Multiflex321.

Para la completa implantación del diseño de la plantilla Multiflex sobre el sitio web, ha sido necesario modificar varios archivos de XWiki, el mayor problema que hemos encontrado durante este proceso de aplicación del *skin* ha sido la integración del código Velocity de XWiki en los ficheros HTML planos que venían con la plantilla.

A continuación la Figura 4.15 muestra un ejemplo de la aplicación completa del *skin*, donde se puede comprobar que se ha conseguido adaptar toda la plantilla sobre el sitio web de XWiki.

The screenshot shows a web page for GAVAB (Grupo de Algorítmica para la Visión Artificial y la Biometría) at the Universidad Rey Juan Carlos. The page has a green header with the GAVAB logo and navigation links like 'Administration', 'Administrador', and 'Log-out'. Below the header is a green banner with the text 'GAVAB • Departamento de ciencias de la computación' and 'URJC'. A navigation bar contains 'CREATE', 'EDIT', 'EXPORT', 'ACTIONS', and 'WATCH'. The main content area is divided into three columns: a sidebar menu on the left with items like 'Página Principal', 'Reconocimiento biométrico', 'Gavab2', 'Metaheurísticas', 'Computación de Altas Prestaciones y Optimización (CAPO)', 'Procesamiento del Lenguaje Natural', 'SIDELAB', and 'Visión Artificial'; a central content area with a 'GAVAB...' heading, a video player showing an eye, and a welcome message; and a 'Novedades:' section on the right listing recent updates from 2007 to 2009.

Figura 4.15: Integración completa del *skin* sobre XWiki

El proceso de integración completa del *skin* Multiflex312 sobre nuestro sitio XWiki ha sido bastante minucioso, a medida que íbamos incorporando de forma paulatina partes del *skin*, el profesor evaluaba cada parte del diseño, certificando que el proceso iba cumpliendo los objetivos marcados.

Se ha validado que funcionan de manera esperada todos los menús, pestañas, accesos a la parte de Administración de XWiki, el buscador, paneles laterales, etc.

4.3. Drupal

4.3.1. Diseño

Como dijimos anteriormente, un *theme* es una colección de archivos que definen la capa de presentación. Los archivos típicos (Figura 4.16) que podemos encontrar en un *theme* de Drupal son:

- Archivos de plantillas o *templates files* (.tpl.php)

Estas plantillas se utilizan para el código XHTML y las variables de PHP.

Cada archivo **.tpl.php* controla la salida de un tipo específico de datos tematizable, y en algunas situaciones, puede manejar múltiples archivos, si no existe ninguno en el tema volverá de nuevo a la salida predeterminada.

La extensión **.tpl.php* se debe a que usa *PhpTemplate* para el formateo de los contenidos. *PhpTemplate* es la librería que se encarga por defecto de formatear los contenidos en Drupal.

Los *themes* que no son del tipo *PhpTemplate* se colocan en una ruta como:

“/home/myweb/sites/all/theme/miplantilla”

- Archivos de hojas de estilo terminados en **.css*

Podremos colocar estos archivos en la misma ruta que las plantillas **.tpl.php*, o meterlos en una carpeta a parte si se trata de varios archivos.

Para que el *theme* reconozca el archivo **.css* es necesario agregarlo al archivo **.info* como veremos a continuación.

- Archivos de JavaScript terminados en **.js*

Se trata de un fichero en el que se incluyen funciones de tipo JavaScript a las que podremos acceder desde el *template.php*.

- Un archivo **.info*.

Metadatos, hojas de estilo, JavaScripts, las regiones de bloques y más se puede definir aquí. Todo lo demás es opcional.

El nombre de este archivo debe ser el mismo que el del nombre interno del *theme*.

- Y un archivo *template.php*

Para toda la lógica condicional y tratamiento de los datos de la salida, es el archivo *template.php*. No es necesario, pero para mantener los archivos **.tpl.php* ordenados puede ser utilizado para sostener preprocesadores para la generación de variables antes de que se fusione con el marcado dentro de los archivos **.tpl.php*. Las funciones habituales, anulando las funciones de tema o cualquier otra personalización se deben hacer aquí. (Este archivo debe comenzar con una etiqueta de apertura de PHP "`<?Php`", pero la etiqueta de cierre no es necesario y se recomienda que la omita). [20]

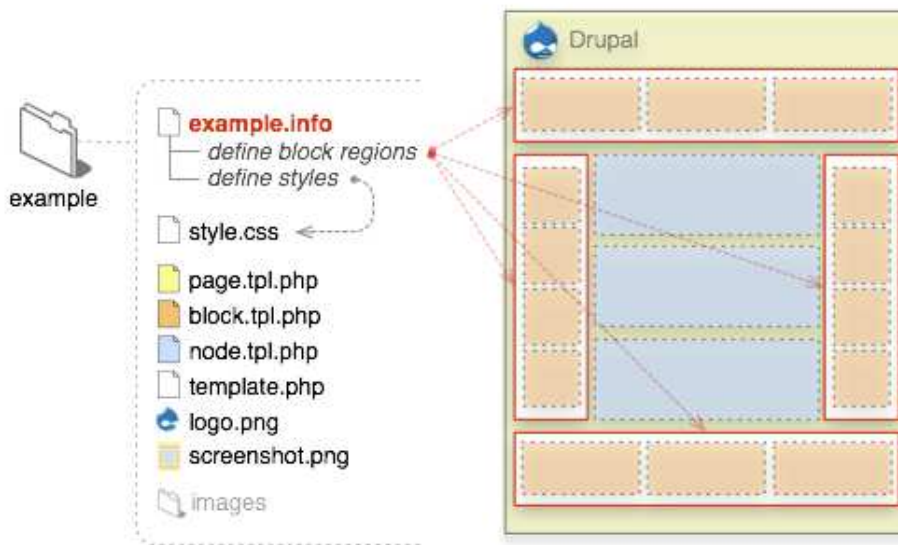


Figura 4.16: Diagrama típico de un Theme en Drupal

4.3.2. Implementación

Vamos a adaptar un *skin* de tal forma que tenga el mismo formato que un *theme* de Drupal, modificando los CSS y algunos archivos más para obtener el nuevo diseño.

En este caso, al igual que hicimos en XWiki, hemos utilizado la plantilla libre Multiflex312 como ejemplo para la creación del *Theme* de Drupal a partir de ella. Realizaremos los pasos generales para poder aplicarlos a cualquier plantilla HTML. Partimos de cualquier tema original de Drupal, en este caso Garland, y le aplicaremos el *skin* Multiflex, que consiste en archivos HTML y CSS.

En la Figura 4.17 podemos ver el diseño del *theme* Garland original de Drupal aplicado a la página web que hemos creado con Drupal:



Figura 4.17: Sitio Gavab en Drupal con theme Garland.

En la figura 4.18 recordamos el aspecto del *skin* Multiflex312 visto anteriormente:



Figura 4.18: Skin Multiflex312

Primer paso, para que Drupal 6 reconozca y detecte el nuevo tema Multiflex312, es copiar la plantilla HTML a la carpeta “sites / all / theme”.

También necesito crear un archivo de información, donde indicaremos el nombre del tema, una pequeña definición, el número de versión y motor del tema que lo soporta.

Simplemente crear un nuevo archivo de texto *.info dentro de la carpeta Multiflex312, llamado de la misma forma " Multiflex312.info", que tenga el contenido que mencionamos anteriormente:

```

name = Multiflex312

description = Descripción que queremos darle al
template.

core = 6.x

engine = phptemplate
    
```

Ahora el nuevo tema se registrará en la ruta “admin /build /theme” y desde aquí podremos configurarlo.

En la administración aparecerá el nuevo tema, como vemos en la Figura 4.19:

Captura de pantalla	Nombre	Versión	Activado	Predeterminado	Operaciones
	Greenery2 Drupal Themes by eShiok.com	1.0	<input checked="" type="checkbox"/>	<input type="radio"/>	configurar
no hay captura de pantalla	indication A sexy little theme with a huge orange block and a large orchid, originally hacked from a free HTML template.		<input checked="" type="checkbox"/>	<input type="radio"/>	configurar
	Marvin Tema gráfico con tablas, en forma de cajas, en tonos grises.	6.16	<input type="checkbox"/>	<input type="radio"/>	
	Minnelli Tema de ancho fijo, con múltiples columnas, sin tablas y de colores configurables.	6.16	<input type="checkbox"/>	<input type="radio"/>	
	multiflex321 multiflex321 is a new theme for Drupal 6.	6.x-1.1	<input checked="" type="checkbox"/>	<input type="radio"/>	configurar

Figura 4.19: Nuevo theme multiflex312

Si habilitamos el *theme* de Multiflex, en la Figura 4.20 vemos el HTML del tema que aparece en las páginas de Drupal, pero sin el formato de las CSS:

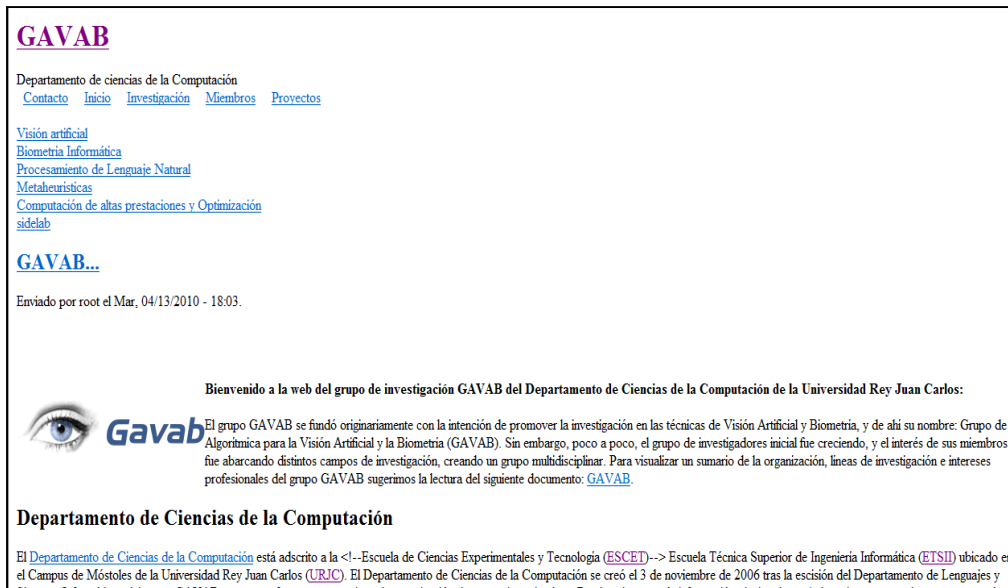


Figura 4.20: Theme Multiflex sin formato.

Aparece sin formato porque Drupal busca dentro del directorio de temas determinados archivos que Multiflex312 no todavía se encuentra vacío. Tenemos que cambiar el archivo original de Multiflex312 *index.html* por un archivo llamado *page.tpl.php*.

Del mismo modo, cambiamos el fichero *default.css* Multiflex312 por un archivo *style.css* para que coincida con la convención de nomenclatura para los archivos CSS que le gusta a Drupal.

A partir de este punto iremos reemplazando partes del código de los *theme* de Drupal en nuestra plantilla HTML para darle el formato adecuado.

Hemos ido comparando los distintos archivos *page.tpl.php* de cada uno de los *theme* de Drupal para averiguar cómo generan contenido los distintos *theme*. Además nos hemos ayudado de la herramienta Firebug de Firefox para ver que archivos dan estilo a los elementos HTML.

Nos centraremos en la parte del *theme* que genera la cabecera los nombres de las páginas por las que hemos pasado hasta llegar a la página en la que nos encontramos, es decir, el *Breadcrumb*. Veamos cómo se genera en la siguiente Figura 4.21, según el *skin* Multiflex312:



Figura 4.21: Breadcrumb con el diseño Multiflex312

En Drupal el archivo que genera esta parte es llamada *breadcrumb*, y el archivo *system.css* es el encargado de darle el formato, se encuentra en “modules/system/system.css”. En la Figura 4.22 vemos como Drupal lo muestra por pantalla con el *theme* Garland:



Figura 4.22: Breadcrumb con el theme Garland.

Veamos el código fuente de cada uno de los *templates* por separado y su integración en la Figura 4.23 que mostramos seguidamente:

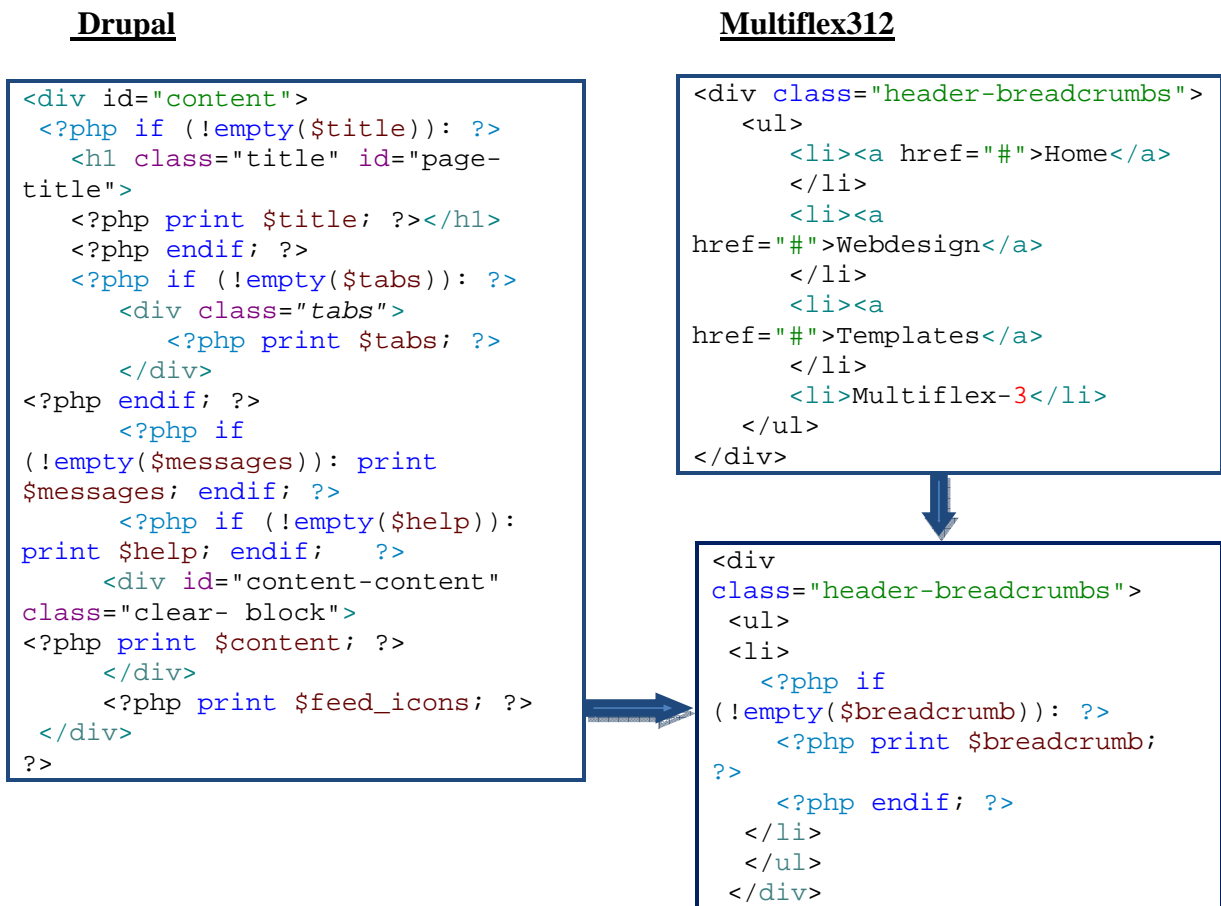


Figura 4.23: Pasos para la transformación del código fuente breadcrumb

El resultado de integrar el *skin* Multiflex es el que mostramos en la Figura 4.24:

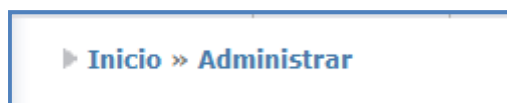


Figura 4.24: Breadcrumb con el skin Multiflex en Drupal.

Vamos a poner otro ejemplo para que quede más claro, al igual que hicimos con XWiki, vamos a mostrar cómo hemos modificado el código de los menús laterales. Veamos cómo muestra Drupal los menús laterales con su theme *Garland*, Figura 4.25.



Figura 4.25: Menú lateral theme *Garland*.

Hemos modificado el código fuente del archivo “page.tpl.php”, que se encarga de dar formato a los contenidos de Drupal, en concreto la parte de código que genera la parte lateral izquierda. Veamos la transformación que hemos realizado en el siguiente ejemplo (Figura 4.26):

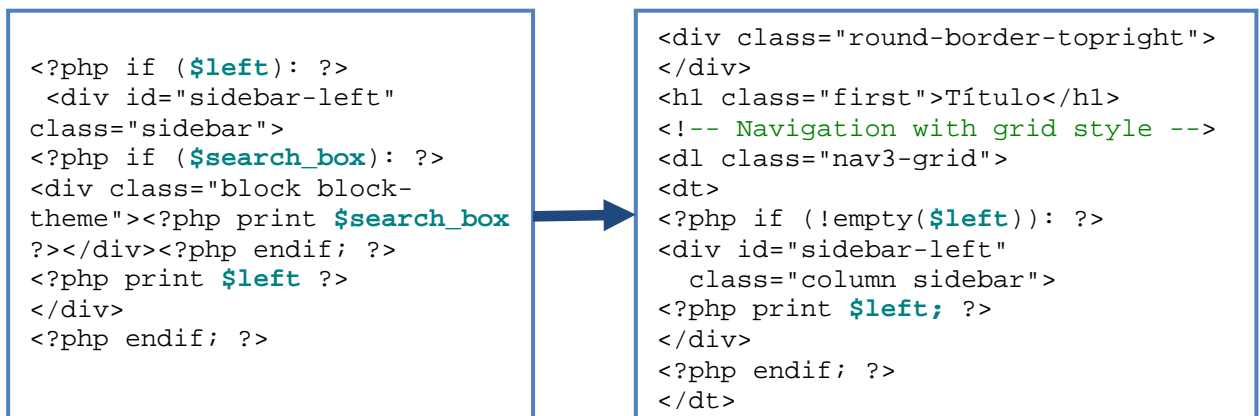


Figura 4.26: Transformación del código del menú lateral

En la figura 4.27 mostramos el resultado, podemos ver que hemos conseguido que el menú lateral tenga el mismo aspecto que en el *skin* Multiflex. Se ha mantenido la funcionalidad de todas las opciones del menú.



Figura 4.27: Menú lateral de Drupal con skin Multiflex

4.3.3. Pruebas

Hemos explicado sólo dos ejemplos de todos los archivos que han sido necesarios modificar para que el sitio web tenga el diseño del *skin* Multiflex321.

Para poder conseguir el diseño de la plantilla Multiflex sobre el sitio web, ha sido necesario modificar varios archivos de Drupal, tanto ficheros PHP, como CSS y algunas imágenes propias del *skin* utilizado.

A continuación vamos a mostrar el resultado completo de transformar el *skin* Multiflex312 HTML en un *theme* nuevo aplicable a Drupal, lo vemos sobre nuestro sitio web como en la figura 4.28.



Figura 4.28: Integración completa del skin Multiflex en nuestro sitio web Drupal.

Durante el proceso de transformación del skin Multiflex312 HTML, en un theme de Drupal aplicable sobre nuestro sitio web, el profesor ha ido evaluando cada parte del diseño, certificando que el proceso cumplía con los objetivos requeridos.

Se ha probado que la funcionalidad del sitio web no se veía afectada por la inclusión del nuevo theme. Se ha verificado el correcto funcionamiento de todos los menús, módulos, el buscador, bloques laterales, así como el acceso a la administración del sitio. Hemos navegado por todo el sitio web, probando además, botones, links, etc. El profesor ha validado que todas ellas tenían un comportamiento correcto.

5. Conclusiones

Con el proyecto realizado hemos logrado demostrar que es posible aplicar un *skin* independiente en HTML sobre algunos sitios web desarrollados mediante sistemas de gestión de contenidos (CMS).

Podemos adaptar de forma completa cualquier *skin* estático y así modificar el aspecto de nuestro CMS, sin perder en ningún momento la funcionalidad del sitio web. Además, los *skins* una vez adaptados, pueden ser exportados a un fichero del sistema de archivos del CMS que estemos utilizando, y así poder ser utilizado por múltiples usuarios.

Hemos logrado modificar el aspecto de varios sitios web cuyas plantillas de estilos eran escasas, les hemos aplicado un *skin* independiente investigando los ficheros necesarios que requerían ser modificados para la correcta adaptación del *skin* los CMS XWiki y Drupal.

A lo largo del proceso de desarrollo e investigación, he adquirido múltiples conocimientos en las tecnologías que están en pleno desarrollo como son los CMS. Su instalación, sistema de archivos, administración de sus datos, etc., han sido conocimientos previos que he adquirido para la realización del proyecto. Además he aprendido a utilizar lenguajes script como Velocity y Php, también hojas de estilo CSS, y *skins* en HTML.

Cada vez son más las empresas y organismos que optan por la utilización de un CMS para el desarrollo de sus portales. El gestor de contenidos facilita el acceso a la publicación de contenidos a un mayor número de usuarios. Permite que sin necesidad de tener conocimientos de programación ni maquetación que cualquier usuario pueda insertar contenido en el sitio web. Además permite la gestión dinámica de usuarios y permisos, la colaboración de varios usuarios en el mismo trabajo, la interacción mediante herramientas de comunicación. Los costes de gestión de la información son mucho menores ya que se elimina un eslabón de la cadena de publicación, el maquetador. La actualización, *backup* y reestructuración del portal son mucho más sencillas al tener todos los datos vitales del portal, los contenidos, en una base de datos estructurada en el servidor.

De cara al futuro sería interesante poder aplicar de forma más o menos automatizada cualquier *skin* que queramos utilizar sobre cualquier CMS con el que estemos trabajando para el desarrollo de nuestro sitio web. Para ello sería necesario continuar el proyecto de investigación aplicando nuevos *skins* sobre otros CMS como pueden ser Joomla, Wordpress, *e-learning*, etc.

6. Bibliografía

- [1] Definición de HTML, <http://es.wikipedia.org/wiki/HTML>
- [2] Descarga de XWiki, <http://www.xwiki.org/xwiki/bin/view/Main/WebHome>
- [3] Descarga de Drupal, <http://drupal.org/project>
- [4] Jommla, <http://www.joomlaspanish.org/>, <http://ayuda.joomlaspanish.org/ayuda-joomla/>
- [5] WordPress, <http://www.misrespuestas.com/que-es-wordpress.html>
- [6] Referencias a los *skins*, http://es.wikipedia.org/wiki/Skin_%28software%29
- [7] CSS, <http://librosweb.es/css/capitulo1.html>
- [8] Sobre PHP, <http://es.wikipedia.org/wiki/PHP>
- [9] Introducción a Velocity,
http://www.programacion.com/articulo/migrar_de_jsp_a_velocity_270
- [10] Definición de Velocity,
http://velocity.apache.org/engine/devel/translations/user-guide_es.html
- [11] Guía del usuario XWiki, <http://enterprise.xwiki.org/xwiki/bin/view/UserGuide/>
- [12] Descarga de Apache 2.0, <http://www.apache.org/>
- [13] Descarga de Php, <http://www.php.net/>
- [14] Descarga de la Base de Datos Mysql, <http://www.mysql.com>
- [15] XAMPP, <http://www.apachefriends.org/en/xampp-windows.html>
- [16] Descarga de módulos oficiales de Drupal, <http://drupal.org/project/modules>
- [17] Home of the Mozilla Project, <http://www.mozilla.org/>
- [18] Definición de Firebug, <http://techtastico.com/post/que-es-y-para-que-sirve-firebug-en-firefox/>
- [19] FCKEdito, <http://weblatam.com/wp/fckeditor-un-editor-html-para-su-sitio-web/>
- [20] Anatomía de un *theme* de Drupal, <http://www.ubuntuz.com/drupal/node/44>