BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

"Sistema de Gestión y Administración de Contenidos Web Inter-facultativa DESIT-BUAP"

Tesis Profesional
para obtener el título de
Licenciado en Ciencias de la Computación

Presenta:
Rodríguez Vergara Arturo

Asesor:
Dr. Manuel Martín Ortíz

H. Puebla de Zaragoza Junio 2012

A mi familia, profesores y amigos, a quienes me enseñaron y quienes me apoyaron, y especialmente a quienes hicieron ambas cosas, gracias.

RESUMEN

Un sitio web o sitio es un elemento muy importante para diferentes tipos de organizaciones, compañías, empresas, asociaciones y por supuesto, colegios, escuelas, universidades y facultades, quienes lo aprovechan como medio de promoción y difusión.

Utilizando las estrategias adecuadas se puede tener un sitio que resulte efectivo y de esta manera aprovechar todas sus ventajas, la obligación a cambio es administrarlo de manera atenta; un sitio mal manejado disminuye considerablemente la calidad del contenido, afectando consecuentemente el impacto positivo y causando el efecto contrario al que se espera.

La propuesta de un sistema que facilite la administración del sitio de la Facultad de Ciencias de la Computación es una solución natural al problema común de administración de contenidos, y resulta razonable plantear la creación de un sistema a la medida que pueda resolver cuestiones particulares del sitio de la Facultad.

Con un sistema dirigido a la necesidad particular, se pueden además construir otras funcionalidades de interés aprovechando la relación que pueden tener con el sitio. En ese sentido se propone entre otros, la creación de mecanismos para permitir administrar los archivos de un proyecto llevado por varios usuarios, de modo que se pueda trabajar colaborativamente entre instancias, algo explotable con las diferentes facultades que conforman a la División de Educación Superior en Ingeniería y Tecnología (DESIT BUAP) que mantiene proyectos entre las Facultades de Ciencias de la Computación, Arquitectura, Ingeniería Química, Ciencias de la Electrónica e Ingeniería, que son las facultades que la conforman.

Para que el resultado sea un sistema sustentable, se deben considerar los problemas que tienen todas las personas o roles de todos los niveles involucrados como usuarios. Al conocer la problemática se pueden crear de manera acertada los componentes requeridos. Teniendo los componentes tenemos conceptualmente la solución, la implementación es ahora el problema, y por supuesto, donde inicia la diversión.

Para la creación del sistema se pasaron varias etapas, cada una por cada arista que implica la creación de un sistema de esta naturaleza. Por la parte tecnológica, ¿Qué herramientas se deben usar?, sobre la lógica de desarrollo, ¿Qué patrones o estándares son más convenientes?, y sobre el mismo proceso de construcción, ¿Qué metodologías son las adecuadas? Las respuestas a las anteriores preguntas han sido los elementos sobre los que se trabajaron para obtener el sistema que se propuso, un gestor de contenidos propuesto con funcionalidades para administrar, compartir y publicar documentos de un proyectos colaborativos.

ÍNDICE

INTRODUCCIÓN

Resumen— Se describen diferentes elementos que son parte del concepto de Web, con atención especial a los sitios, elementos y problemas de éstos. Se hace una inmersión en el problema de un sitio funcional y manejable. Se hace un estudio del estado del campo y se analizan las acciones realizadas por algunas instituciones para conseguir administrar un sitio de manera efectiva. Finalmente se presentan los motivos para usar un gestor de contenidos personalizado.

Conceptos – páginas, sitios, CMS

El concepto de web fue concebido por Tim Berners-Lee al desarrollar el protocolo HTTP (Hypertext Transport Protocol) usado básicamente para transferir archivos, el HTML para dar formato a los archivos y una URL (Universal Resource Locator) para poder localizarlos bajo un identificador único en una red de computadoras.

La web es una red compleja que permite acceder a contenidos ubicados en prácticamente cualquier parte del mundo de manera rápida y sencilla, lo que la ha convertido en un importante medio de comunicación, con un funcionamiento que opera básicamente con páginas e hipervínculos. Una página de Internet es un documento electrónico que además de contener texto, puede cargar archivos multimedia o en otros formatos.

Una página web es el elemento más simple bajo la visión de contenedor de información en la WWW (World Wide Web o web). Es funcional para desplegar contenidos cuando los datos publicados tienen un tamaño considerable pero no necesitan ser seccionados en diferentes vistas.

De la creación del sistema de la WWW al día de hoy, la manera en que se utiliza este recurso ha cambiado. En la mayoría de los casos, los autores de la información necesitan más de una página para presentar todo el contenido, o en otra instancia, dividir la información de una manera más amigable. Esta problemática se soluciona fácilmente haciendo más páginas y creando enlaces entre éstas, es el conjunto de varias páginas que generan un sitio. Los sitios son los contenedores más comunes pues tienen las características que la mayoría de los publicadores necesitan: transmitir información de una manera estructurada y de manera accesible, organizada de manera seccionada.

Existe una gran variedad de sitios por la relativa facilidad de creación y la enorme cantidad de temas que pueden ser expuestos. No existe una estructura definitiva para la creación de un sitio y dependiendo de las necesidades o enfoques se tienen diferentes tipos de sitios, y por tanto distintas clasificaciones. Además de sitios y páginas en senti-

dos estrictos existen más estructuras especializadas disponibles dependiendo del paradigma y fin de lo publicado, como bitácoras o portales web.

Un portal web, de manera similar a los sitios (incluso considerado un tipo de estos), es un conjunto de páginas con una gama de soluciones diferentes. Los portales pueden verse como un centro de servicios con difusión de contenidos. Los contenidos ofrecen secciones informativas y dentro de los servicios se pueden encontrar foros de opinión, herramientas web o salas de charla, servicio de correo, etc., esto dirigido a un grupo definido de personas. Los portales no sólo actúan a modo de redirección hacia sitios, sino que ofrecen también accesos a recursos, sistemas y aplicaciones [1].

Actualmente la Facultad de Ciencias de la Computación mantiene un sitio que de alguna manera cumple su función, esto sin hacer uso de elementos de un portal. Del mismo modo, el sistema que se planteará tiene como fin generar un sitio con las mismas características, pero con mecanismos para la administración y gestión.

Con base en la clase de contenidos podemos catalogar a los sitios en dos tipos: horizontales y los verticales. Los horizontales gestionan información universal y de distintos tópicos; están orientados a un público general (aunque esto no significa que no puedan existir secciones dirigidas a un perfil de usuario específico). un ejemplo de portal vertical puede ser un portal de especialidades académicas. Los verticales giran en torno a un contenido especializado, están dirigidos a un grupo de usuarios de acuerdo a un perfil por gustos, intereses o necesidades; un sitio de facultad universitaria pertenece a este segmento, desplegando contenidos directamente relacionados con la facultad o a sus miembros, sean estudiantes, académicos, etc.

El crecimiento constante de usuarios que publican información por Internet ha sido uno de los antecedentes directos en la evolución de la manera en que se crea y estructura un sitio o un portal y sus contenidos. Esta evolución se ha dado tanto en front-end (la capa que ve y con la que interacciona un usuario) como en el back-end (la capa interna con la implementación de funciones).

Acerca del front-end se puede mencionar que el soporte para estilizar documentos ha evolucionado de contar sólo con etiquetas de titulación, estructuras en listas y tablas, y otros pocos recursos a un control separado para diseñar el estilo (ampliando las posibilidades, facilitando la aplicación en diferentes documentos de un mismo estilo y la actualización) junto con la posibilidad de incrustar gráficas o insertar diferentes tipos de medios enriqueciendo el documento a presentar. Además, se han hecho análisis de accesibilidad para hacer implementaciones que faciliten la comunicación cuando existen capacidades diferentes en los usuarios.

En back-end se hicieron replanteamientos arquitectónicos aplicando patrones de desarrollo y esquemas más completos que permitieron la creación y administración de sitios sin importar el tamaño de la información y separando la implementación de vista al usuario. De aquí parten los CMS (Content Management Systems), sistemas de gestión y administración de contenidos que permiten trabajar con facilidad cuestiones administrativas que además eximen al usuario de realizar varias tareas técnicas y generando front-ends de calidad manejables fácilmente.

Un CMS es un sistema que se utiliza para crear sitios y gestionar fácilmente su información, ignorando hasta cierto punto el tipo de contenido que administra. La meta principal del gestor de contenidos es hacer que la máquina trabaje más los aspectos internos de la administración dejando a los encargados de la aplicación como responsables de la parte final del proceso, la de gestión de información; esto significa mayor productividad con un trabajo relativamente menor. Otra ventaja palpable es hacer posible que los diferentes roles como administración o edición puedan ser llevados por menos personas.

Análisis del problema

Internet es un medio de difusión fundamental para diversos tipos de asociaciones. Para institutos de nivel superior, es un mecanismo que permite fácilmente publicar constantemente cualquier cantidad de información y hacerla accesible mundialmente. Específicamente hablando de la Facultad de Ciencias de la Computación, está claro que es un elemento que juega un papel importante como medio informativo y de proyección.

Es fácil caer en un sitio inconsistente si es que se tiene más de un administrador, si no se cuenta con una guía de estilo y un sistema de plantillas que establezca los esquemas de presentación, o si no se tiene una manera eficiente de gestionar las páginas; el motivo que cause inconsistencia, cual sea, da como resultado un sitio poco funcional.

Mantener un sitio con el tamaño del de la Facultad de Ciencias de la Computación de manera bien administrada y consistente es una tarea demandante que además requiere de conocimientos particulares para poder hacer las modificaciones necesarias de funcionamiento y mantenimiento, aun cuando levantar el servicio pueda ser relativamente sencillo.

Para el caso de la Facultad, se ha optado por un sitio administrado sin una herramienta propiamente adecuada, algo que ha dado lugar a inconsistencias en diversos elementos, como publicaciones que no presentan la información con un esquema visual regular, algo evidente en el área de Secretaría Académica; o con páginas perdidas, borradas o con enlaces rotos, como el enlace del menú principal de 'Asociación de Alumnos' (no funcional al momento de redactar este documento).

La inconsistencia manifiesta una baja calidad del sitio, que se traduce en dos efectos: el sitio simplemente no se ve bien o trabaja mal, lo que refleja una mala imagen del equipo de administración y la organización o empresa que representa. Por otro lado, el sitio es un lugar poco interesante.

Dependiendo de las posibilidades podría pagarse un servicio y delegar el trabajo a personal externo, sin embargo otra opción viable es hacer uso de una herramienta que ofrezca la funcionalidad para realizar todas esas tareas administrativas de manera rápida, y sin requerir de una formación técnica sobre computación por parte del o los administradores.

La administración de la imagen del sitio puede considerarse secundaria, pero que además es una tarea tediosa de hacer manualmente y que además implica mucho más trabajo que la administración a través de herramientas. La presentación visual influye claramente en la impresión de la institución hacia el exterior, de modo que contar con un sistema que permita delegar la cuestión de estilos para concentrarse en contenidos es de interés.

Sin una herramienta adecuada pueden generarse cuellos de botella causados por la creación de nuevas páginas de manera manual. Aunque el esquema visual se encuentre definido con plantillas, o páginas de donde extraerlas, se dificulta la aplicación y modificación, haciendo cada vez más fácil caer en una sobrecarga de trabajo para los tiempos planeados [2]. Al utilizar un CMS se puede fácilmente publicar contenidos de manera dinámica, permitiendo una interacción adecuada entre los grupos que publican y los que reciben la información.

Al migrar el sitio deben considerarse ciertas cosas como el grado de dificultad que se tendrá al hacer la migración y el hecho de que el sitio estará fuera de linea. También hay que considerar el nivel de dificultad al trabajar con el nuevo sistema por cada uno de los usuarios y roles, en caso de requerir autores, editores o incluso desarrolladores [3].

Acerca del proceso de migración, para el caso de la Facultad de Ciencias de la Computación, puede considerarse que, si bien requiere un esfuerzo que podría llegar a considerarse algo más que trivial, la complejidad y duración de la tarea será en total mucho menor al trabajo de mantenimiento futuro de manera manual. Del mismo modo, el manejo de un nuevo sistema implica una curva de aprendizaje y adaptación que aunque obligue al administrador a aprender el uso de nuevos elementos, éstos al final resultan ser más efectivos parar realizar las mismas tareas y más sencillos de usar.

Por otro lado, y del mismo modo en que la publicación de información a través de sitio de la facultad es importante, lo puede ser la publicación de documentos generados al

realizar un proyecto colaborativo, explotando el sitio como medio de divulgación de conocimientos.

Así como no existe un mecanismo de publicación general, tampoco existe un mecanismo que permita trabajar de manera colaborativa sobre los documentos de proyecto, y hacerlos públicos en caso que se desee, algo deseable para compartir fácilmente hallazgos o información resultado de investigaciones. Idealmente la funcionalidad para trabajos colaborativos debe permitir que documentos que sean definidos como públicos, sean accesibles desde las instancias que tienen usuarios colaborando en un proyecto, facilitando la difusión de la información generada. La creación del mecanismo mencionado anteriormente necesita una plataforma sobre la que pueda ser desarrollada, y que en este caso no está disponible.

Poniendo en una balanza ventajas y desventajas entre el método actual o la implementación de un sistema que automatice las actividades y permita la inclusión de nuevas funcionalidades, es evidente que un CMS daría mejores resultados en tiempo, calidad y esfuerzo.

Estado del campo

Crear un sitio web puede volverse un trabajo difícil y un poco pesado si las herramientas con las que se cuentan no son suficientes. Anteriormente, pasar por ello era necesario para tener un sitio que tuviera la cantidad de información y la imagen visual esperada para un sitio competitivo. Ésta es una de las causas del nacimiento de los sistemas de gestión de contenidos.

Los CMS, son sistemas que ofrecen una base estructurada bajo un patrón visual en páginas web para organizar la información, así como una interfaz gestora de la información contenida. Al gestionar un sitio manualmente se pueden detectar fácilmente los problemas de administración y ha sido con esa problemática identificada que se han desarrollado sistemas encargados de trabajar automáticamente las tareas posibles para simplificar en trabajo y disminuir ampliamente la probabilidad de cometer errores al manejar la información [4].

Originalmente los sistemas gestores de contenidos fueron desarrollados como herramientas internas de apoyo para la publicación de contenidos de corporaciones. Las actualizaciones constantes en las publicaciones mostraron que métodos de automatización de parte de los procesos mejorarían el rendimiento con un trabajo menor, y los esfuerzos fueron dirigidos al desarrollo de herramientas para la administración; como ejemplo puede mencionarse el sistema desarrollado por CNET (portal de tecnología, http://www.cnet.com/) y Vignette (Sistema para portales sobre Java, http://www.vignette.com/) a mediados de los 90, que tenía como meta hacer publicaciones personaliza-

das y que utilizaba código que había sido escrito inicialmente para la gestión interna de CNET [5].

Con el avance del tiempo este tipo de sistemas ha ido evolucionado, de forma que al día de hoy existe un abanico de opciones que abarcan un rango considerable de soluciones disponibles para administrar sitios de distinta índole con problemáticas comunes.

Implementaciones de gestores de contenidos existen de diversos lenguajes, como Java, PHP, Python o Ruby, tanto de uso libre como a través del pago de una licencia. Actualmente tan sólo en PHP existen varios CMS de uso libre, con licencias amigables, tanto de propósito general como Joomla! (http://www.joomla.org/), centrado en publicación de páginas, bitácoras, calendarios y varias características más avanzadas, o Drupal (http://drupal.org/), un CMS nacido a partir de un BBS (Sistema de Tablón de Anuncios) especialmente útil para la creación de comunidades, sistemas de este tipo son desarrollados con un esquema de portal en mente.

Las categorías de sitios más comunes tienen gestores particulares, como Wordpress (http://wordpress.com/) un gestor de contenidos especializado para bitácoras en linea que cuenta con múltiples funcionalidades añadidas para mejorar la experiencia de uso o LifeType (http://lifetype.net/) una plataforma para múltiples bitácoras y usuarios. Otras soluciones incluyen sistemas como OpenCart (http://www.opencart.com/) para ventas en línea, otros especializados en galerías de imágenes como Zenphoto (http://www.zenphoto.org/) o redes sociales como Elgg (http://elgg.org/). La variedad de opciones refleja en cierta medida la cantidad de necesidades. De particularidades nacen generalidades y la cantidad de necesidades diferentes en torno a un sitio ha favorecido la generación de un número interesante de opciones disponibles para sitios con esquemas comunes.

Los CMS tienen diversas características importantes que hacen tentativo o necesario su uso: la facilidad de poder editar y manejar la información para tener actualizado el sitio, la generación de páginas dinámicamente al momento de demandarlas, la posibilidad de utilizar bases de datos para guardar información de manera ordenada y cómodamente accesible, y un estilo visual homogéneo adaptable a las necesidades de quien lo utiliza, a esto se puede agregar el hecho de que los recursos necesarios para soportar el uso de un administrador no son mucho más altos que los necesarios al trabajar directamente los archivos de un sitio [6].

A pesar de la afortunada variedad de opciones no hay un sistema disponible públicamente que implemente las funcionalidades que la Facultad de Ciencias de la Computación requiere y algunos elementos deben ser programados de acuerdo a exigencias particulares, algo probado con desarrollos internos hechos con anterioridad para distintos servicios como la sala de revistas o el administrador de material docente. Para dar solu-

ción a esto se puede considerar modificar un sistema existente o construir uno personalizado.

En caso de la bifurcación, es una opción viable si se cuenta con un sistema que contenga varios de los elementos necesitados, una arquitectura acorde y la complejidad de las adaptaciones tanto iniciales como futuras sean costeables. Es importante también que el sistema adaptado no use más recursos de los que realmente son necesarios ni que incluya funcionalidades no requeridas, que incluso puedan en algún momento comprometer el sistema.

El desarrollo de un CMS a la medida, del que tratará este proyecto, tiene la ventaja de que al ser un sistema personalizado utilizará los recursos de acuerdo a las necesidades propias. La construcción del sistema será para que éste se adapte de manera a las necesidades, sin embargo la construcción completa puede agregar esfuerzos mayores dependiendo de la complejidad de los requerimientos y de las funcionalidades, como el caso de las funcionalidades para gestionar los documentos de un proyecto entre diversos usuarios o facultades.

Instituciones con la misma problemática y sus soluciones

Además de organizaciones y sitios corporativos, las soluciones de gestores de contenidos pueden ser muy bien aprovechadas por instituciones educativas, de modo que facultades y entidades similares se han dado a la tarea de analizar el impacto de la implantación de un CMS para administrar su sitio finalizando con una migración a la herramienta más adecuada para sus necesidades en unos casos, y en otros desarrollando un sistema a la medida.

No hay una estadística que nos permita saber el número de universidades que optan por esta solución, sin embargo existen algunas instituciones que han documentado el proceso de análisis y adaptación o migración a un CMS de manera análoga a la propuesta que se hace con este proyecto. A continuación se presentan los trabajos de algunas universidades sobre el análisis e implantación de un sistema gestor de información.

La Universidad de Brandeis de Waltham, Massachusetts, EEUU tiene un estudio y análisis de necesidades, casos y requerimientos de lo que un CMS debería de cubrir en su caso (http://lts.brandeis.edu/about/projects/cms/index.html).

La sección tiene como fin informar y dar detalle del análisis de funcionalidades web necesarias basadas en la experiencia ganada en su comunidad y con organizaciones de TI (Tecnologías de Información). Ese análisis fue usado para seleccionar un gestor de con-

tenidos que cubriera sus necesidades, haciendo la aclaración de que cada institución tiene su propio grupo de necesidades, criterios y objetivos.

El proyecto desarrollado está dividido tres en fases:

Usuarios y usos – Incluyendo encuestas de necesidades y análisis en el flujo de publicaciones. Esta fase es la que resulta más interesante e importante. Basándose en este análisis se realizarán las fases siguientes.

Dentro de las problemáticas identificadas y que el sitio de la Facultad de Ciencias de la Computación comparte se pueden mencionar las siguientes:

- La publicación web es complicada, tediosa y frustrante
- La falta de tiempo
- La falta de habilidades
- La falta de soporte
- Necesidad de bases de datos
- Uso de Javascript para mejorar la usabilidad

Además se identifican necesidades similares a las del caso particular, entre las que resaltan:

- La falta de una guía
- Los publicadores necesitan servicios de tecnología web para comunicarse de manera más fácil y efectiva
- Tener plantillas que simplifiquen la publicación web

Selección de sistema – Usando como apoyo el sitio web de CMSWatch para seleccionar candidatos, y en última instancia la selección del CMS.

Implementación – Basada en preguntas que cuestionan lo que tiene que hacer un sitio web, qué estructura se quiere llegar a tener, o cómo un CMS puede ayudar a dar una mejor estructura a la presencia completa en la red.

Este estudio aunque no culmina con la implementación de un sistema propio, sirve como buen análisis de necesidades que plasma elementos considerables para este caso siendo que la problemática es semejante.

La Universidad de Bradford, Bradford, West Yorkshire, Reino Unido creó el proyecto CMS: The Content Management System (CMS) Programme of projects (http://www.brad.ac.uk/admin/cmsproject/what.php)

Iniciado con la fase de análisis de negocio, diseñada para analizar el flujo de información desde que es publicada por los miembros del equipo de trabajo hasta que llega a audiencias externas, con la premisa de mejorar el proceso para garantizar una comunicación simple y efectiva.

La segunda fase es un reporte que contiene un documento de requerimientos, un compendio de CMS y sus creadores y una auditoría al sistema que manejan ya internamente. El reporte final (http://www.brad.ac.uk/admin/cmsproject/documents/final-report-small.pdf), en la sección 2.3 se listan los hallazgos identificados, entre los compartidos por el sitio de la Facultad de Ciencias de la Computación se pueden mencionar:

- Existen varias personas que editan la información, ésta es causa de un sitio inconsistente
- No todos tienen las mismas habilidades técnicas
- No existe un software estándar para editar la información
- Distintas escuelas han implementado su propio sistema de publicación duplicando esfuerzos

La tercera fase es la elección de un CMS que cumpla con las necesidades encontradas.

La Universidad Florida del Norte, Jacksonville, Florida, EEUU, tiene desarrollado un proyecto de análisis de funcionalidades y requerimientos para la selección de un CMS capaz de dar solución a las necesidades de la universidad - http://www.unf.edu/cms/.

Dentro de los problemas a atacar, se encuentran los más comunes: el sitio es inconsistente, uso de diferentes estilos, enlaces rotos, mal diseño, etcétera; que aplican igualmente en este caso.

El proyecto se divide en tres etapas: una para compra de equipo y software, configuración y primeros diseños de plantillas; otra etapa para la conversión de páginas principales, y una etapa final para la conversión de sitios por división, complejidad, etc., además de soporte continuo.

La Universidad de Alberta, en Edmonton, Alberta, Canadá después del análisis formal, en el departamento de sistemas web, hicieron el desarrollo de un CMS capaz de ofrecer los servicios necesarios para su instituto - http://www.uofaweb.ualberta.ca/uofaweb/cms.cfm. Este sistema se ha implantado exitosamente en varios de los sitios de la universidad, incluyendo diferentes facultades y departamentos (http://www.uofaweb.ualberta.ca/uofaweb/completedsites.html).

Existen más ejemplos de migración de sitios universitarios hacia una solución CMS (como por ejemplo http://louisville.edu/web/migration), donde se opta por un software que si bien es de uso libre tiene que ser adaptado para cubrir requerimientos propios.

Los análisis desarrollados por los departamentos de las universidades mencionadas anteriormente hacen un trabajo completo para detección de necesidades y funcionalidades, sin embargo la creación de un sistema CMS propio generalmente no es la última meta sino la selección de uno creado por terceros, sin importar que no sea libre y que en algunos implique un gasto monetario para el pago de una licencia. De manera contraria a algunas facultades y del mismo modo que la Universidad de Alberta, el sistema CSM que se plantea para el sitio de la Facultad de Ciencias de la Computación se concibe como un desarrollo completo del sistema pasa trabajar los elementos necesitados.

Motivos para usar un CMS

Se puede necesitar un CMS si el conjunto de la información, su administración y su proceso de publicación son muy complejos para administrarlos de manera informal. La complejidad puede ser entendida bajo las siguientes pautas [7]:

- La cantidad de datos que se tienen. Esta cantidad no sólo incluye el total de los datos, sino también el número de tipos de datos que se intenta administrar.
- La cantidad de contribuciones. Hay que contar no solamente el número de contribuidores, sino también la relación o rol que juegan, y el tipo de información con que contribuyen.
- La cantidad de cambios que se espera en el contenido. Los cambios pueden ocurrir tanto en la cantidad de información que se agrega o se elimina como en el número de cambios en el diseño que se considera soportar.
- El número de publicaciones que se desea crear. La complejidad en la publicación depende del número de diferentes publicaciones que se pretende crear así como del grado de personalización que se piensa incluir.

Al contextualizar lo anterior sobre el sitio de la Facultad de Ciencias de la Computación se puede notar que, acerca de la cantidad de información, esta abarca diferentes ámbitos y en ocasiones esta llega a ser considerable. Sobre los contribuidores o colaboradores, puede considerarse que aún cuando actualmente la administración cae sobre un encargado, idealmente deberían de poder hacerse modificaciones por otros usuarios dependiendo de la información que se quiera editar. De la cantidad de cambios y el número de publicaciones, éstos varían constantemente conforme al curso de la institución, algo evidente con los constantes cambios que se hacen.

Además de los elementos generales que deben ser solucionados, el sitio para la Facultad de Ciencias de la Computación tiene necesidades específicas que hay que considerar. La funcionalidad de desplegar información de la planta docente con algunos detalles es un elemento fundamental, y el sistema debe considerar una funcionalidad para permitir administrar esa información de manera rápida y por los propios individuos, una funcionalidad con la que se cuenta actualmente, pero dando opción a que la información sea editable por el mismo usuario al que pertenece el perfil.

Dentro de la Facultad de Ciencias de la Computación existen departamentos y equipos de investigación. Los estudios e investigaciones nacidos en esos departamentos son en ocasiones compartidos entre diferentes facultades que pertenecen a la misma área. Este modelo de trabajo es usado por la DESIT que mantiene proyectos realizados conjuntamente entre las facultades que la integran.

Los elementos mencionados anteriormente y otros, como la funcionalidad para publicar noticias de distintos tipos clarifican el interés de tener un CMS que considere y solucione además de las problemáticas comunes, las problemáticas particulares. Enfocado directamente a solucionar las necesidades que el sitio web de la facultad tiene, que provea facilidades para publicar de manera rápida y ordenada la información deseada, con herramientas sencillas que ofrezcan productividad, facilidad de administración en un ambiente cómodo.

De la necesidad de tener un sistema gestor de contenidos que ofrezca a la Facultad de Ciencias de la computación un mecanismo práctico para mantener constantemente actualizado su sitio web es que se concibe la creación de un sistema que genere un sitio que mejore la experiencia del usuario al hacer uso de ella, incluyendo distintos tipos de usuario como alumnos, profesorado, egresados, departamentos administrativos y aspirantes.

Se propone la creación de un CMS que además de contener la funcionalidad para crear y gestionar páginas a través de usuarios y roles, contenga funcionalidades para solventar las necesidades antes mencionadas, de modo que se tenga un gestor de información que permita la administración de proyectos internos o inter-facultativos, algo que puede ser utilizado por las facultades que conforman a la DESIT. La integración esta funcionalidad permitirá que los documentos de los proyectos administrados en el CMS puedan hacerse disponibles públicamente de manera sencilla, facilitando la difusión de la información generada a partir de trabajos colaborativos.

Siendo una solución con código abierto, al término del desarrollo se tendrá una herramienta con posibilidad de crecimiento, y que pueda ser adaptable a nuevas necesidades. El sistema puede funcionar también como base de futuros desarrollos o permitir utilizar las secciones de código que sean de utilidad sobre otras aplicaciones.

CAPÍTULO 1. SITIOS

Resumen— Se trata la evolución de los elementos de una página y consecuentemente el sitio generado. Se estudian los diferentes tipos de sitios considerando características o elementos clave, además de los usos que se les pueden dar, y relacionando este análisis con el sitio de la Facultad de Ciencias de la Computación. Finalmente se presentan los motivos para el uso de un gestor de contenidos y sus ventajas para administrar fácilmente el sitio de la Facultad.

Los sitios web son un conjunto de páginas (X)HTML generalmente con relación directa, accesibles comúnmente desde un mismo dominio o subdominio. Tienen normalmente una página inicial dentro del primer nivel accesible. Las páginas que se acceden mediante los URL tienen una jerarquía de ubicación y acceso, que sin embargo, se ve saltada por el control que tienen los hipervínculos y sus enlaces sobre las páginas.

Con el crecimiento en el número de accesos a los servicios en la red y que cada vez es una alternativa más usada como medio de comunicación, y el avance y maduración de las tecnologías disponibles para generar los sitios, se tienen sitios web más completos y complejos.

1.1 Evolución de la web

La web ha sido un medio con cambios importantes desde su aparición. En principio la red estaba formada por sitios de contenido estático, se estaba explorando un nuevo medio de comunicación.

Desde el inicio de la web ha habido un proceso de evolución que ofrece mejores y cada vez más completos servicios. Comparando los primeros sitios con los actuales, se observan cambios en cuatro elementos: El tipo de contenido, la estructura del contenido, la presentación con que se despliega el contenido, y la capacidad de interaccionar con este contenido.

Aunque la calidad de la información es intemporal, la diversidad de información se ha incrementado. Esto en parte por la capacidad de incluir elementos de diferentes del contenidos basado en texto, como videos o fuentes de audio; además, la información se ve afectada por la manera en que está organizada y es presentada.

Una estructura correcta facilita el acceso a todas las páginas, reemplazando secuencias lineales por árboles con enlaces mixtos. Una buena estructura permite tener un acceso sencillo e intuitivo sin importar la cantidad de páginas.
La limitante inicial de cómo presentar los contenidos ha disminuido drásticamente. Actualmente es posible obtener el efecto de presentación de prácticamente cualquier otro medio, por la característica esencial del manejo de contenidos en múltiples medios, lo que provee una cantidad suficiente de recursos para poder presentar el contenido.

La interacción de usuarios en un origen no estaba conceptualizada enteramente. Inicialmente se seguía el esquema de los medios de comunicación convencionales donde el flujo de información es en un solo sentido. La capacidad de la web de poder interaccionar con el usuario fue despertando interés al mismo tiempo en que las tecnologías ofrecían la posibilidad de explotar el nuevo elemento de comunicación. Este cambio ha sido también impulsado por tendencias sociales y los beneficios de la comunicación e interacción directa.

Los anteriores cuatro elementos son usados como un método de categorizar los sitios, separando en generaciones de la web; la primera, en una versión inicial donde las funcionalidades eran limitadas y básicas y las posteriores que contemplan la interacción entre personas o sistemas.

1.2 Categorización de sitios

Dependiendo del enfoque con que se analice un sitio y considerando distintos elementos fundamentales, los sitios se pueden catalogar de diferentes maneras. El sitio que debe generarse con el sistema a desarrollar ciertamente debe solventar las necesidades sobre diferentes categorías. Algunas de las categorías más comunes de los sitios son:

Ubicación y acceso. Por la localización y el método de acceso al sitio se puede tener un sitio de Internet, que es abierto al acceso público (aunque se necesiten en algunos casos credenciales de autenticación), el sitio a generar por el sistema CMS que se contempla generará es evidentemente un sitio de Internet, siendo que el fin principal es hacer disponibles públicamente la mayoría de los contenidos. Un sitio de Intranet tiene un acceso delimitado a una zona de manera local, esto es muy útil para empresas y organizaciones. Un sitio en Extranet es usado con los mismos fines que en Intranet, para tener sitios privados para entidades, sin embargo el acceso puede ser hecho a través de Internet.

Tipos de apertura. Dependiendo del nivel de acceso del usuario a la información, puede ser abierta cuando el acceso a los contenidos puede hacerse directamente como en el caso del sitio de la Facultad de Ciencias de la Computación, donde la información

se publica para que tenga un acceso general; cerrada cuando el acceso a la información es siempre pasando por un punto de control y semicerrada en caso de tener acceso limitado a algunos contenidos de manera directa.

Dinamismo. El sitio puede ser dinámico, donde los contenidos dependen de alguna petición y se muestran de manera personalizada, o estáticos donde el contenido está fijo. Aquí se contemplan los paradigmas de web 1.0, 2.0 y 3.0. La web concebida originalmente formada de sitios web estáticos está catalogada como 1.0. El concepto de una sitio web sólo consideraba la publicación de información sin cambios por periodos largos de tiempo y con visitantes como consumidores de contenidos. La comunicación es unilateral con la información servida por el servidor y enviada al navegador donde es desplegada. La página web 2.0 está viva; en contacto con el servidor con el que se comunica además de hacerlo con el usuario [8].

La web 2.0 supera las limitantes de interacción existentes en la versión conceptual anterior. Es principalmente una web colaborativa. La versión 2.0 es un concepto un tanto disperso asociado a un fenómeno social, donde caen varios elementos que proporcionan mayor facilidad de acceso a contenidos y retroalimentación (Fig. 1.1). Aunque el término 2.0 podría pensarse que hace referencia a una actualización en tecnologías, engloba a los diferentes cambios que se han ido dando tanto del lado de la tecnología como en el aspecto social de la comunicación.

La Web 3.0, es un paradigma similar al 2.0, un conjunto de elementos y avances tecnológicos, sin embargo el proceso de comunicación considera la interacción de sistemas de manera similar a la interacción entre usuarios y el significado semántico de la información.

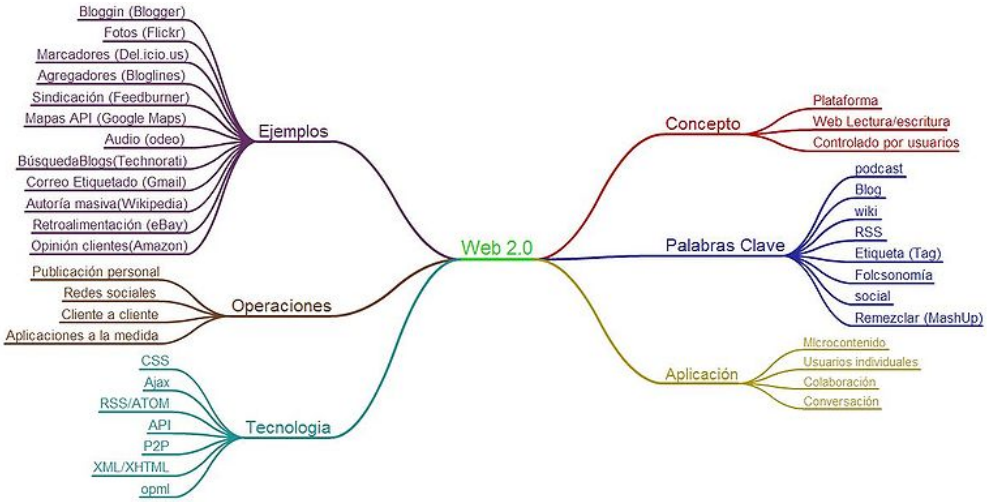

Fig. 1.1. Mapa mental web 2.0, fuente: wikipedia.org

La clasificación según esta categoría es un tanto difusa. El sitio generado claramente permite a distintos usuarios administrar los contenidos, sin embargo, no es una funcionalidad totalmente abierta o de interacción comunitaria. Se integran elementos para facilitar el trabajo como validadores de formularios y editores avanzados, herramientas usadas normalmente por sitios web 2.0. La naturaleza del sitio de la Facultad (por tanto del sitio generado por el CMS del proyecto) permite explotar funcionalidades propias de la web 2.0 sin que necesariamente se le catalogue dentro de esta versión, siendo como muchos sitos similares un sitio híbrido entre 1.0 y 2.0.

Por el fin que persigue. Diferentes sitios tienen diferentes objetivos, como el concepto de negocio o entretenimiento, entre otros. Esta diferencia no implica el uso de diferentes tecnologías. Un sitio comercial puede ser de negocios, de servicios, imagen corporativa, etc. Un sitio corporativo se utiliza para afianzar el conocimiento de la empresa y la marca dentro de los consumidores, esto para crear oportunidades de negocio. Otro tipo de sitios son los deportivos, comunitarios y personales. El sitio a generar esta enfocado en publicar información relacionada con la institución o las personas que ahí participan, de modo que se puede identificar como un sitio de información y de noticias entre otros, esto será tratado en la sección 1.3 Tipos de sitio por su uso.

Por estructura lógica. La estructura es un conjunto de relaciones establecidas en, entre y más allá de piezas de contenidos individuales. Para que un conjunto esté bien estructurado debe tener las siguientes características [9]:

Su contenido se divide en un conjunto de categorías bien definidas, que pueden llamarse tipos de contenido. Con cada tipo de contenido, la información es segmentada en porciones manejables, que pueden llamarse componentes de contenido. Cada porción de componente de contenido se divide en un conjunto de partes bien definidas, que pueden llamarse elementos. Si se piensa en el sitio de la Facultad, se pueden considerar como parte de un grupo a las páginas regulares, dentro de otro las páginas encargadas de publicar noticias, y otro más con las páginas de los profesores, más adelante se explicará acerca de cada uno de éstos y otros elementos.

La estructura lógica indica la manera en que se relacionan las páginas entre ellas y sus vínculos existentes. Pueden considerarse cuatro formas principales de organización de la estructura considerando la estructura: Los lineales, los de rejilla, los jerárquicos y los enrejados o de tela de araña [10].

- *Lineales*. Una de las estructuras más comunes donde se usa el mismo esquema de la impresión tradicional (Fig. 1.2). Sigue paso a paso una línea definida aunque en algunos casos se modifique parcialmente el comportamiento agregando alternativas lineales o laterales. Este tipo está relacionado con páginas con listados que seccionan el resultado en diferentes páginas, con la botones de siguiente y anterior.

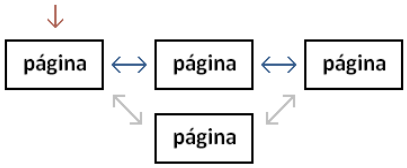

Fig. 1.2. Diagrama de esquema lineal con alternativas

- *Jerárquicos*. (Fig. 1.3) Es una estructura común en la que desde el inicio o raíz se ofrecen accesos a nuevos niveles de información, éstos de la misma manera ofrecen enlaces a niveles más bajos, dando una estructura de árbol. Pueden ser ampliados fácilmente y adaptarse a distintos tipos de necesidades. Este esquema es el que se utilizará al ser el que mejor se adapta a las necesidades de actualización, crecimiento y adaptación de la información del sitio web que se quiere generar con el sistema a construir. La Fig. 4.31 la sección del módulo de micrositios plasma la estructura de las páginas donde se puede observar que este esquema se adapta naturalmente al sitio actual y que de igual manera se considera generar a través del sistema.

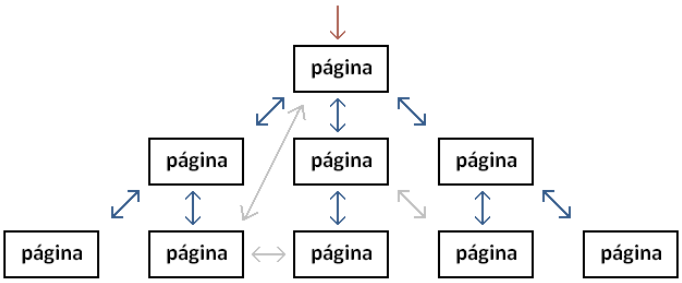

Fig. 1.3. Diagrama con estructura en árbol, las flechas claras representan acceso mixto

- *En red y tela de araña*. (Fig. 1.4) En esta estructura poco común los elementos están comunicados linealmente y entre diferentes niveles. El esquema de red comunica de manera horizontalmente y verticalmente los accesos, el de malla completa contiene enlaces prácticamente entre todas las páginas y el esquema de tela de araña contiene enlaces que comunican algunas secciones con otras de manera análoga a como se ve una tela de araña, enredada.

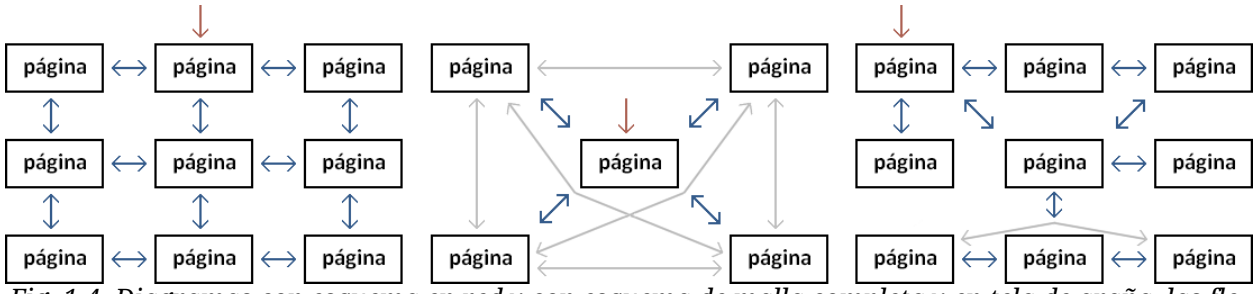

Fig. 1.4. Diagramas con esquema en red y con esquema de malla completa y en tela de araña, las flechas claras representan enlaces alternativos

1.3 Tipos de sitios por su uso

Formalmente no hay reglas para el uso de un sitio; si la información con la que se cuenta es lo suficientemente grande para ser dividida, el uso de un sitio generalmente es apropiado, esto hace que sea aplicable en diferentes ámbitos. Recordando la categorización por el fin perseguido, pueden mencionarse entre otros los siguientes:

Sitio de información. Son sitios de información general, la información que se ofrece no es de manera comercial. Este tipo de sitios se ofrece por parte de distintas entidades y asociaciones como gobiernos o instituciones educativas. Este tipo de entidades tiene generalmente sitios que informan sin intenciones directas o primarias de lucro. El sitio con el que cuenta la Facultad es un sitio de información, precisamente dirigida a diferentes usuarios, como alumnos, egresados e incluso a externos que tengan interés. El módulo de micrositios explicado en el capítulo 4 contiene las funcionalidades para que el CMS a construir permita gestionar las páginas por los usuarios de manera administrada.

Sitio de noticias. Los sitios de noticias son similares a los sitios de información, sin embargo el contenido está enfocado a noticias y eventos de actualidad. Muchos de estos sitios aprovechan tecnologías actuales como RSS (un mecanismo para publicar noticias con estructura XML) para facilitar el acceso a la información que está apareciendo constantemente. Los sitios de noticias engloban también a los sitios de periódicos, revistas y algunos sitios de canales de televisión. En este tipo de sitios la comunicación puede ser unilateral con el lector, o interactivo con secciones para hacer comentarios sobre las publicaciones. La Facultad de Ciencias de la Computación es una institución con diferentes tipos de eventos, o hechos que son precisamente noticias. El sistema que se propone considera funcionalidades para facilitar la publicación de noticias, y presentarlas en el sitio generado, el módulo de noticias explicado en el capítulo 4 trata al respecto.

Empresas y negocios. Un sitio de negocios da a conocer los productos de una empresa basados en mercadotecnia para generar oportunidades de negocio.
La información principal es el catálogo de productos, información de las instalaciones, de los certificados de calidad, etc.

Sitios de servicios. Tiene como productos a sus servicios, que puede ser el acceso a bancos de información o a bases de datos. Los sitios de servicios pueden ser parecidos a los portales por el tipo de manejo de información. Un sitio de servicios puede ser uno de publicación de ofertas de empleo, de salas de plática, páginas amarillas, etc.

Comercio electrónico. El comercio electrónico o e-business está basado en tiendas virtuales para la publicación de mercancía que está a la venta. Son los productos lo de mayor importancia, dejando en segundo término la información de la empresa. Estos

sitios generalmente utilizan un motor para la publicación y control de los datos de manera automatizada. Este tipo de sitios están muy comprometidos con accesos seguros y políticas de seguridad avanzadas.

Comunitarios y colaborativos. Toman el concepto de comunidad y se construye un sitio alrededor de él. Los sitios comunitarios se basan en la interacción de usuarios y sus conocimientos. Un sitio de comunidad puede girar en torno a una agrupación o un tema. Estos sitios se basan en la interacción a pesar de no ser una red social aunque se aplican conceptos similares. Un sitio de comunidad, tiene contenidos son publicados , editados, enriquecidos y mejorados tanto por los editores del sitios como por parte los demás integrantes de la comunidad. Los sitios colaborativos tienen una comunidad que la soporta, sin embargo la metodología es diferente, aunque la información se edita colaborativamente se hace de manera controlada, algo que crea una interacción menor. El sistema a desarrollarse considera que los contenidos pueden ser administrados por todos los usuarios que se deseen de modo que sea posible gestionar colaborativamente la información si es que se desea, tanto para páginas, como para artículos o proyectos colaborativos.

Personales. Tipos de sitios que se usan como medio de expresión individual, con información publicada por uno o algunos autores. La estructura de la información llega a ser en ocasiones poco convencional. La información contiene generalmente puntos de vista personales y opiniones. La temática es elegida por su creador y basada en ideas y pensamientos propios aunque algunas veces son derivados de otros contenidos. El contenido es muy variable. Estos sitios también llamados en ocasiones blogs o weblogs, son en esencia bitácoras en web.

Artísticos. Un sitio como medio de expresión, permite ser usado con motivos o fines artísticos. Son un medio de expresión artística de su creador o creadores. Este tipo de sitios suele saltarse convenciones o patrones, dejando al autor o artista como responsable del efecto positivo y del impacto que se quiere lograr al ser visitado. Estos sitios aprovechan en gran medida el avance en la estilización de la presentación y la diversidad de medios disponibles.

Buscadores y directorios. Los buscadores son sitios que permiten consultar el contenido de diferentes páginas, regresando las ubicaciones de las páginas que contienen los datos buscados. Los directorios contienen información variada dividida en categorías y subcategorías. Actualmente sitios buscadores contienen también un directorio, como Yahoo! o Google.

Entretenimiento. Un sitio de entretenimiento contiene elementos que dan al usuario opción de esparcimiento y diversión. Se utilizan con fines similares a los de otros medios de comunicación enfocados al entretenimiento, como la televisión y radio,

además de poder incluir esas categorías. Estos sitios publican información acerca de cine, música, juegos o cualquier información de interés general. Los sitios de entretenimiento normalmente ofrecen sus servicios de manera gratuita y a cambio presentan publicidad constantemente a los usuarios usando esto como el modelo de negocio.

Existen más tipos como portales, sitios educativos, de descargas, de archivado, entre otros.

1.4 Uso de recursos por un sitio

Un sitio es funcional cuando ofrece los servicios que sus usuarios necesitan y lo hace utilizando los recursos correctos. Cada sitio utiliza métodos y estrategias diferentes, y en la medida en que se usen los elementos más adecuados será el éxito obtenido.

Existen diversas tecnologías y técnicas desarrolladas que permiten obtener un gran número de resultados de diferentes tipos, esto facilita obtener un sitio con las funcionalidades deseadas. La inclusión de efectos con Javascript pueden ser muy útiles. La inclusión de editores de código mejorados puede incrementar la productividad, del mismo modo una validación de formularios antes de enviar la información pueden eliminar errores antes de enviar la información. Es importante agregar las funcionalidades cuando están justificadas y son necesarias y no agregar elementos que puedan afectar, en rendimiento o usabilidad la funcionalidad del sitio.

Dependiendo de las funcionalidades o recursos de interacción ofrecidos por un sitio se obtienen beneficios que en algunos casos pueden también generar desventajas. Un ejemplo puede ser el sitio de Wikipedia (http://www.wikipedia.org/), una enciclopedia en línea que permite a cualquier usuario modificar los contenidos, lo que la ha hecho una de las fuentes más grandes de información en Internet, sin embargo como la información puede ser falsa o incorrecta, la fiabilidad es cuestionable en ocasiones.

El sistema CMS a desarrollar considera el uso de mejoras por Javascript y mecanismos de trabajo colaborativo, sin embargo la intención no es hacer un sitio web 2.0, que hace uso de estas y otras herramientas, sino uno que cumpla con los fines para los que se dispone el sitio de la Facultad; esto es, ofrecer al exterior la información con una estructura similar a la del sitio actual pero incluyendo funcionalidades para que internamente el proceso de gestión y administración de la información sea más sencillo y menos propenso a errores humanos.

1.5 CMS para la creación de sitios

Un CMS es un conjunto de herramientas y funcionalidades integradas para la administración de contenidos de una manera sencilla y práctica que facilita la tarea de tener un sitio con buena imagen y en un estado consistente.

Recordando las características de los CMS que se han mencionado, uno de los objetivos de un sistema CMS es presentar sitios con un contenido de calidad, uniforme, con información fácilmente accesible para el usuario y fácilmente gestionable por administradores. Un usuario que navega en sitio construido con un CMS navegará seguramente sobre un sitio organizado y actualizado. Estas características ayudan y facilitan la consulta además de hacerla más agradable. Este efecto es más difícil de conseguir sin un sistema que ayude a gestionar la información.

Se pueden mencionar dos razones generales de peso generales para utilizar la gestión de contenidos [11] aplicables directamente sobre el sitio de la Facultad de Ciencias de la Computación:

1. Un gestor de contenidos es la solución a la lluvia de información que actualmente se maneja. Los sitios web se salen de control; se espera cosechar información útil de muchas fuentes. Este fenómeno se ve claramente en el sitio de la Facultad de Ciencias de la Computación al desplegar cambios de manera constante en la portada para agregar nuevas ligas a nueva información.

2. Un gestor de contenidos dirige al paradigma de esta época: “¿Cómo es posible dar un valor particular y sustento a una porción de información?” El uso de sistemas CMS permite administrar la información y desplegarla con una presentación consistente, algo que agrega valor tanto al visitante como al administrador, algo que es de interés.

Los puntos anteriores abordados por los gestores de contenidos cambiarían de manera positiva el sitio de la Facultad posibilitando una administración en la medida en que va siendo necesario, sin depender de un usuario administrador que modifique de manera técnica el sitio para agregar los cambios. Al visualizar el CMS a generar hay que considerar las preguntas: ¿Qué funcionalidades se deben tener disponibles? ¿Se necesitan listados de noticias, productos, o catálogos? Aunque un gestor de contenidos es una sola aplicación, en ocasiones interconecta sus servicios (publicando interfaces de programación) o se conecta a otras aplicaciones como foros de discusión, listas de correos o a otras aplicaciones [12].

CAPÍTULO 2. PLANTEAMIENTO DEL PROBLEMA

Resumen— Se expone el problema: El sitio de la Facultad de Ciencias de la Computación tiene problemas de inconsistencia por su administración manual y no permite la implementación de nuevas funcionalidades. Se presentan los justificantes para el desarrollo del sistema que se propone, un CMS para las necesidades particulares. Se trata el papel de la ingeniería de software y el manejo de metodologías en el desarrollo del sistema.

Los CMS implementan elementos de uso común que son necesarios en un sistema para administrar el sitio de la Facultad, sin embargo estas implementaciones no solucionan todas las necesidades particulares.

El sitio web de la Facultad de Ciencias de la Computación ha sido un recurso importante que ha ido creciendo y mutando al paso del tiempo. El inconveniente actual es la aproximación que se utiliza a veces al actualizar el sitio: el manejo de nuevos archivos por nuevos contenidos, que dificulta otras tareas como el dar formato de manera consistente y las habilidades necesarias para realizar la tarea para conseguir presentaciones homogéneas, y la mecánica para comunicarse con la base de datos, que no considera filtros como elemento de seguridad.

Aunque el sistema ha sido mantenido por años, los métodos para hacerlo en algunos casos resultan poco eficientes, y en otros, regresan resultados diferentes a los esperados. Se hace evidente la necesidad de un mecanismo que brinde facilidades sobre este proceso, y que permita concentrarse en manejo del contenido delegando la tarea de la imagen final.

Recordando la sección que trata el análisis del problema en la Introducción se puede retomar el tema de la problemática de los sitios inconsistentes. La inconsistencia afecta distintas partes del sitio, lo que impacta negativamente en diferentes áreas. Como se ha mencionado, el sitio de la Facultad tiene diversas inconsistencias por las deficiencias en los métodos usados para su mantenimiento. Las inconsistencias afectan tanto en accesibilidad con ligar rotas que no permiten la consulta de información, como en el ámbito visual en donde el esquema visual en unas ocasiones pierde continuidad y en otras presenta la información en una manera claramente mal estructurada.

En el tema de *Motivos para usar un CMS*, también parte de la Introducción, se menciona la consideración de cambios y contribuciones. Esto impacta en los cambios y modificaciones y en los usuarios que deberían de poder realizar esos cambios. El uso de un CMS con manejo de usuarios permite delegar la tarea publicación y modificación de contenido además de ofrecer una manera sencilla para realizar la tarea, algo muy complicado si se trabaja manualmente. El sistema a realizar contempla un sistema de usuarios que

permita distribuir las colaboraciones conforme a roles, algo explotable con administradores de algún departamento.

Existen beneficios agregados al contar con un gestor de contenidos además de facilitar la actualización de contenidos, como eliminar algunos posibles errores al hacer el trabajo de modo manual, o definir desde un centro de control configuraciones que afectan elementos generales en el sitio. Estos beneficios extras muestran que el uso de un CMS es una opción más viable frente al recurso actual. Además, la plataforma sobre la que se ejecuta el CMS, puede utilizarse para integrar nuevas funcionalidades como la propuesta para administrar documentos de proyectos.

Partiendo de lo anterior es que se propone el desarrollo de un sistema que gestione los contenidos de la Facultad de Ciencias de la Computación, que por un lado permita trabajar con la información de manera sencilla, y que por otro lado despliegue de manera ordenada los contenidos para facilitar la navegación.

En el CMS a desarrollar se propone la creación de diferentes funcionalidades para las particularidades que hay que atender. El sistema a planteado considera las funcionalidades para la publicación de páginas y noticias o la adición de enlaces externos por distintos usuarios, distribuyendo el trabajo de actualización. El mecanismo de usuarios se explotará para crear el manejo de perfiles de la planta académica de modo que cada profesor pueda actualizar sus datos para mantener un directorio de la planta docente correcto y actualizado.

2.1 Justificantes

Aunque existen gestores de contenidos disponibles, son implementaciones con funcionalidades generales, y que no consideran algunos elementos específicos del sitio de la Facultad (como un gestor de páginas de personal académico), esos elementos deben de ser desarrollados para solucionar una necesidad que debe ser atendida. Otro problema con un gestor general es la existencia de funcionalidades innecesarias que pueden afectar el desempeño de las funciones que son utilizadas, en cambio un sistema escrito con fines particulares puede utilizar de una mejor manera los recursos disponibles.

Al revisar el planteamiento del problema se ve necesaria la creación de un sistema web que cuente con un administrador manejable de manera sencilla y cómoda, que presente una alternativa viable a la necesidad de la Facultad de Ciencias de la Computación y posiblemente la DESIT, de tener un espacio de publicación en Internet que mejore la difusión de sus servicios, proyectos y/o eventos; sin que los encargados de alimentar en contenidos al sitio necesiten formación técnica para la tarea, siendo necesaria sólo una familiarización promedio con el uso de aplicaciones de cómputo.

Este proyecto busca dar solución a las necesidades de la Facultad de Ciencias de la Computación con la creación de un sistema que permita publicar y gestionar el contenido de una manera sencilla, que automatice aspectos visuales y de manejo de contenidos. Con servicios como gestores de datos para profesores que permitan crear perfiles del profesorado, mecanismos de colaboración entre usuarios de ya sea del mismo sitio u otros, o sistemas de redifusión de fuentes de contenidos entre diferentes instancias, además de considerar mecanismos para gestionar la información y las páginas que le dan forma al sitio. La meta es obtener al final un sitio con información actualizada y con un esquema visual adaptable, dos características muy importantes en sitios de calidad.

Como se ha mencionado en los *Motivos para usar un CMS*, la Facultad de Ciencias de la Computación es parte de la DESIT. Entre las facultades integrantes surgen proyectos inter-facultativos. El sistema propuesto contempla un conjunto de funcionalidades que permitan administrar los documentos llevados por el proyecto. Eso aprovecha la capacidad de almacenamiento de los documentos dentro de la aplicación y el mecanismo de usuarios que debe de existir para poder colaborar en el sitio. Esta funcionalidad pretende facilitar y organizar los documentos de un proyecto inter-facultativo y permitir a la vez que que esos documentos puedan hacerse disponibles de manera controlada en las diferentes instancias que tengan usuarios colaborando en el proyecto, favoreciendo la difusión de la información y el conocimiento generado por el equipo de trabajo.

2.2 Ingeniería de software en el proyecto

La ingeniería de software es un proceso constante que se lleva a lo largo del desarrollo de un sistema para facilitar su implementación, tratando de prever y mitigar de manera efectiva los posibles problemas durante el ciclo de creación. Es un marco conceptual de trabajo que a manera de etapas -parcialmente- ordenadas y definidas, establece una guía de desarrollo desde la conceptualización hasta la finalización de un producto de software de manera efectiva. El fin es mejorar la productividad e incrementar el rendimiento de los involucrados en el desarrollo tratando de facilitar en la mayor medida posible las tareas a realizar.

La ingeniería es el análisis, diseño, construcción, verificación y administración de entidades técnicas. Al considerar la ingeniería sobre el desarrollo de este proyecto se pueden analizar las siguientes preguntas [13]:

¿Cuál es el problema a resolver? o ¿Que resultado se quiere obtener?
Tener un sitio que resulte fácilmente administrable y que permita gestionar la información por diferentes usuarios además de facilitar el trabajo colaborativo entre diferentes instancias o facultades.

¿Qué características de la entidad son usadas para resolver el problema? y ¿Cómo se realizará la solución?
Se toman en cuenta a los elementos principales que han mostrado ser necesarios en el sitio actual de la Facultad de Ciencias de la Computación, así como los problemas existentes que hay que solucionar. Se implementarán las funcionalidades explotando tecnologías, metodologías, arquitecturas, esquemas que se han desarrollado para la construcción de sistemas.

¿Bajo qué metodología se solucionarán los errores encontrados?
Los errores detectados al ser notificados se solucionan considerando el nivel de importancia del error y el nivel de impacto al realizar el arreglo, siendo que el desarrollo es por un integrante no existen problemas colaterales por trabajar paralelamente el desarrollo.

El trabajo asociado a la ingeniería de software puede ser dividido en tres etapas [13]:

La fase de definición, que se enfoca al *qué*. Durante la definición, quien realiza el análisis tiene la tarea de identificar qué información será procesada, el desempeño, el comportamiento esperado, las interfaces necesarias para obtener un sistema exitoso, esta se genera el listado de requerimientos, presentada al final del capítulo.

La fase de desarrollo que se enfoca en el *cómo*. Eso es, durante el desarrollo, quien lleva la ingeniería de software, se encarga de definir cómo deben ser estructurados los datos, qué función debe implementarse bajo qué arquitectura, cómo deben ser implementados los detalles de los procedimientos, cómo se traslada el diseño al lenguaje de programación. Esta es una etapa continua desde el inicio del desarrollo que finaliza al obtener el producto.

La fase de soporte se enfoca en los cambios asociados a la corrección de errores y las adaptaciones en la medida en que se va requiriendo.

El desarrollo de este proyecto sigue una metodología formada por un conjunto de tareas organizadas mediante un esquema basado en SCRUM de una manera simplificada. SCRUM es una metodología de desarrollo de sistemas con un esquema iterativo incremental, esto es, el proceso está formado por una serie de iteraciones o ciclos para construir en cada una de éstas un conjunto de funcionalidades que constituyen una porción funcional del proyecto. Esta metodología resulta útil para este proyecto por la manera en que se pueden organizar y desarrollar sus componentes, usando un ciclo por cada componente o módulo.

La metodología tiene dos elementos importantes: pilas de producto y sprints. Una pila de producto es una lista de requerimientos o necesidades y algunas de sus características.

La pila está formada por historias de usuario correspondientes a diferentes requerimientos normalmente funcionales. Los elementos que contiene comúnmente una historia de usuario son: un identificador, un nombre o descripción corta, un nivel de importancia, una estimación inicial y una manera de comprobar el funcionamiento.

Un sprint es un ciclo corto de desarrollo en un lapso de tiempo que considera los recursos que se utilizan para implementar por partes los requerimientos en una pila de producto. A cada grupo de requerimientos asignados a un sprint se le llama pila de sprint.
Cada sprint, en el desarrollo del sistema propuesto tiene como meta la implementación de un módulo, un conjunto de librerías o el código base del sistema sobre el que correrán los módulos. que contiene el conjunto de funcionalidades para solucionar un problema, esto es tratado más a fondo en el capítulo 3, en la sección de requerimientos funcionales.

2.3 Metodología de desarrollo

Para este proyecto se consideran a los dos elementos principales de la metodología SCRUM: la pila de producto y los ciclos de implementación. Para el uso de esta metodología sin embargo los dos elementos son adaptados para ser prácticos en el proceso.

La pila de producto es un documento que normalmente tiene una orientación al dueño del producto. El dueño del producto es quien concibe el sistema que se va a desarrollar, y es normalmente un gestor de negocios dando una solución a un cliente. En este caso la pila no tiene un enfoque de negocios sino dirigido a la implementación de modo que se utiliza el listado de requerimientos funcionales y no funcionales para este fin. El manejo de más de una pila se da en sistemas seccionados o gran envergadura, particularmente una pila es suficiente pues permite además tener una mejor concentración en las necesidades y su desarrollo [14].

Al ser la pila de producto el listado de análisis de necesidades; no se necesitan definir identificadores, cálculos iniciales o priorizaciones siendo que no habrán liberaciones parciales. Asimismo, existe un conjunto de requerimientos no funcionales. Los requerimientos no funcionales son los deseables inherentes al tipo de sistema, no hablan del las funciones, sino de las características del funcionamiento y no deben de perderse de vista durante el proceso de desarrollo.

CAPÍTULO 2. PLANTEAMIENTO DEL PROBLEMA

CAPÍTULO 3. NECESIDADES A CUBRIR

Resumen— Se discuten los elementos que tienen que ser atendidos por el gestor de contenidos, considerando las necesidades que se deben cubrir. Por el lado técnico, se hacen consideraciones para que el sistema sea internamente manipulable. Presenta un listado con los requerimientos funcionales y no funcionales que se consideran para el sistema a desarrollar.

Aunque la característica principal de un CMS es gestionar contenidos en la web, si tomamos ésta como la única funcionalidad truncaríamos el avance que ha generado. Los sitios tienen páginas de texto, sin embargo no todas estas páginas tienen el mismo fin. Aún cuando el texto es la parte principal, muchos sistemas son explotados con una demanda que exige más flexibilidad como en quién tiene acceso a qué, y quién puede hacer qué [15].

Después de conocer la situación actual del sitio y entender cuales son los problemas tanto comunes como particulares, se puede proceder a listar las necesidades en los diferentes niveles que deben de ser cubiertas por el sistema que se propone.

3.1 Funcionalidad básica

Para desarrollar el sistema propuesto se necesita una plataforma que tome en cuenta funcionalidades básicas. En este caso plataforma se refiere a framework, que es una infraestructura digital que permite la construcción de sistemas de manera organizada. Los elementos básicos que deben ser parte del framework y consecuentemente del CMS a generar son las siguientes [16]:

Continuidad: A pesar de las limitaciones de protocolos web básicos, muchas funciones de los sitios web necesitan retener información a través de una serie de interacciones de usuario y la información debe de ser protegida. El CMS debe de manejar esto de una manera en que haga sencilla esta tarea, para que las funcionalidades y extensiones del sistema puedan manejar cualquier tipo de datos que necesite.

Gestión de usuarios: El framework necesita proveer una base para que el CMS pueda controlar usuarios a través de algún tipo de autenticación. Sin embargo esto debe de ser lo suficiente flexible de manera que se necesite una cantidad mínima de código para manejar el requerimiento, que puede ir desde un solo usuario con el rol de administrador hasta el manejo de cientos o miles de usuarios diferentes con un método flexible de verificación de credenciales.

Control de acceso: Siempre requerido, aunque sea solamente para definir quién puede configurar el sitio. Generalmente se necesita más, pues varios grupos de usuarios pueden tener permisos sobre diferentes funcionalidades. Es ampliamente aceptada la idea de que el mejor acercamiento es el sistema de control de acceso basado en roles (Role-Based Access Control o RBAC). Esto significa que son los roles los que tienen concedidos los permisos, y quienes acceden necesitan tener un rol asignado.

Gestión de extensiones: Un framework es más útil si puede extenderse fácilmente. No hay ninguna funcionalidad visible de usuario que sea esencial para cualquier sitio web, así que idealmente el framework debe de estar dividido en funcionalidades. Cada capacidad visible para los usuarios debería de ser agregada como una extensión. Cuando los requerimientos para construir un sitio web son considerados, sale a la luz que puede haber muchos tipos de extensiones diferentes. Una clasificación conocida es la división en componentes, módulos, extensiones y plantillas.

Seguridad y manejo de errores: Todo el mundo está consciente de la existencia de diferentes tipos de problemas o amenazas, desde la publicidad no deseada hasta los intentos de violar la seguridad de sitios web. Para que la seguridad sea efectiva tiene que ser construida dentro del sistema desde el inicio para que de esta manera no sólo el framework consiga la mejor seguridad posible, sino también provea un ambiente de ayuda para construir extensiones seguras. Ambos errores son significativos, tanto los problemas de usabilidad, como las fallas de seguridad, así que un mecanismo de manejo estándar de errores es necesario.

Es deseable para la administración del sitio un sistema que tenga funcionalidades que permitan crear, editar y gestionar contenidos, y de igual manera poder asignar y administrar usuarios. Estos usuarios, dependiendo de los roles que tienen asignados deben de poder usar diferentes herramientas como: un mecanismo que permita a un profesor mantener una página con su perfil que pueda administrar directamente, un conjunto de funcionalidades para poder realizar trabajos en colaboración con otros usuarios, siendo del mismo sitio o de otra instancia (y poder realizar proyectos en conjunto por más de una facultad), una manera de presentar un RSS personalizado que sea capaz de desplegar los contenidos de otros sitios (y lograr una difusión más grande a trabajos de diferentes facultades), o poder usar la autenticación SSH.

El sistema en un principio debe de contar con un método para desplegar un sitio completo y funcional. El sitio debe ser navegable fácilmente. Los contenidos deben estar esquematizados y con una identificación sencilla de las funcionalidades del sitio.
Esta idea y los conceptos antes expuestos –continuidad, gestión de usuarios y extensiones y un manejo de errores– junto con los requerimientos para el sistema conforman a los elementos mínimos necesarios para el sistema que se propone. Las vistas que tiene

el usuario deben de facilitar la navegación a diferentes áreas. Idealmente debe de existir un menú generado dinámicamente dependiendo de los contenidos.

3.2 Una estructura manejable dentro del CMS

Un CMS idealmente debe de contener ciertas características dentro de su construcción para tener un buen rendimiento, tanto al utilizarse desde cualquier perfil del usuario final, sea visitante, usuario o administrador (el objetivo principal del sistema) como al desarrollador para adaptar la tecnología a las nuevas necesidades dentro de la administración de un sitio.

Al desarrollar un gestor de contenidos deben de considerarse estructurar y organizar correctamente el código. Esta es una buena práctica un tanto independiente del tipo de sistema que se desarrolla, aunque la arquitectura del sistema juega una parte importante en la manera en que se estructura y divide el código. Cada porción de código debe estar en el lugar que le corresponda; eso facilita enormemente las actualizaciones y la manipulación por otros desarrolladores. El código debe estar organizado en archivos dependiendo del tipo de contenido y de qué funcionalidad forma parte. Si se maneja una correcta separación del código, se puede definir una manera para cargar el código a necesidad evitando el gasto de memoria y procesamiento, y facilitando el mecanismo para poder manejar código de otros orígenes.

Al codificar, se deben de generar clases que aunque sean particulares para el sistema, estén desarrolladas de manera que puedan ser expandidas fácilmente en un futuro.
Algunas de las maneras de hacer el código fácilmente extensible y de mantenimiento sencillo incluyen [17]:

Usar un sistema de plantillas que separe la lógica de aplicación de la lógica de despliegue.
Usar una abstracción de la base de datos para manejar la interacción con el servidor de la base de datos.
Hacer uso extenso de las capacidades de programación orientada a objetos para organizar el código.

Utilizar patrones de diseño. El código desarrollado basado en patrones es fácilmente entendible y extensible. Los patrones son soluciones a diferentes problemas en el desarrollo de software que han sido probadas ampliamente. Al utilizar un patrón de diseño tenemos varias ventajas entre las que se encuentran: una guía para estructurar el código que evita buscar soluciones a problemas ya conocidos, y una estandarización de la estructura del código que facilita la comprensión del código por otros desarrolladores. Por otro lado, el uso excesivo puede entorpecer el avance o dificultar la manipulación

del código por otras manos, por lo que hay que aplicar patrones cuando sea conveniente.

Manejar una base de datos y a través de una interfaz. Muchas aplicaciones web necesitan acceder a bases de datos para funcionar de una manera efectiva. Normalmente un CMS está construido bajo un framework que por sí mismo necesita una base de datos para realizar sus funciones. Si el framework tiene una interfaz de acceso a la base de datos, se tendrá normalmente un único flujo para el manejo de datos, eso se puede aprovechar para filtrar de una manera sencilla cualquier acceso y aplicar medidas de seguridad. Asimismo, con la interfaz se puede independizar el código de interacción con la base del tipo de manejador, facilitando la implementación en diferentes gestores de bases de datos.

Codificar con planeación al futuro. Además de utilizar patrones para estructurar el sistema, hay que tener en mente que el sistema puede ir creciendo en funcionalidades. Si la lógica de programación considera que nuevas soluciones pueden ser construidas, al momento de actualizar será mucho más sencillo y cómodo.

3.3 Análisis de necesidades

Con el análisis de necesidades se inicia el proceso de ingeniería del sistema, donde se especifican las necesidades de un sistema en computación en uno o diferentes niveles. El reto para quien lleva la ingeniería de software es cómo asegurarse de que se conceptualice un sistema que contenga propiamente las necesidades del cliente y cumpla con sus expectativas. Para solucionar de manera efectiva se debe de realizar un proceso sólido de ingeniería de requerimientos, esto provee un mecanismo apropiado para entender qué es lo que se quiere [18].

3.3.1 Requerimientos no funcionales

El sistema debe contemplar los siguientes puntos:

Debe ser seguro. Que proteja la información y dar acceso controlado a las diferentes áreas.

Debe ser controlable y fiable. Que permita realizar las tareas que se requieran desde una interfaz amigable y sencilla y que garantice que la información se mantenga consistente y coherente.

Debe ser escalable. La escalabilidad considerada desde el diseño de la arquitectura permite que en un futuro se pueda dar crecimiento al sistema de una manera fácil, reduciendo ampliamente las modificaciones al código base al realizar tareas de mantenimiento.

Debe tener una presentación manejable. El sistema tiene que desplegar sus contenidos de modo que facilite la navegación y la consulta de información.

El usuario debe de tener acceso a las funcionalidades independientemente de si tiene Javascript activo o no.

El modelo de arquitectura debe separar el código dependiendo su tipo para facilitar el mantenimiento en diferentes ámbitos (reglas de negocio, optimización de código, cambios visuales, etc.).

Debe de tener una manera sencilla de instalación que facilite la identificación de errores de configuración al instalar el sistema.

3.3.2 Requerimientos funcionales

Funcionalmente la meta es conseguir un CMS orientado a la Facultad de Ciencias de la Computación que facilite el proceso de publicación y administración de contenidos en diferentes formatos como texto publicado, imágenes o archivos de otro tipo; que dé como resultado publicaciones rápidas con una presentación final homogénea y de calidad. Que cuente con mecanismos de interacción para fomentar la participación de las comunidades que rodean a la institución.

Se han separado las funcionalidades de manera modular para organizar e identificar de una manera más sencilla los elementos que deben estar presentes. El desarrollo del sistema debe dar como resultado un CMS que sea capaz de realizar las siguientes tareas:

Módulo de administración

Debe contar con un área de administración general que:

- Permita acceder a un administrador a un área de configuración general, donde se puedan configurar elementos generales, como el título y descripción del sitio, la personalización de mensaje de error.
- Contenga un sistema para controlar la página principal de manera sencilla, permitiendo modificar la estructura en que se presenta la información por la vista general del sitio (no a nivel de modificación de plantilla completa).
- Contenga un sistema para configurar el despliegue o no de los enlaces para acceder a que funcionalidades (módulos implementados).

- Permita definir un estado de mantenimiento para bloquear temporalmente el acceso al sitio para poder hacer modificaciones o adaptaciones.
- Cuente con una zona para subir imágenes para usarlas al crear páginas.

Módulo de usuarios

Debe contar con un área accesible por los visitantes que:

- Despliegue una forma de autenticación para iniciar una sesión como usuario registrado.

Debe contar con un área de administración general que:

- Permita configurar un método de autenticación de usuarios directo o por SSH.
- Permita crear, gestionar y eliminar usuarios.
- Permita crear cualquier número de grupos.
- Permita asignar cualquier número de grupos a un usuario.
- Permita asignar a un grupo el permiso de acceso a todos los módulos que se necesiten.

Debe contar con un área de administración de usuario que:

- Posibilite a un usuario editar los datos de su cuenta.
- Permita acceder al modo de administración por usuario a cualquier módulo que tenga permitido por el o los grupos a los que está asignado.

Módulo de profesores

Debe contar con un área accesible por los visitantes que:

- Despliegue un listado de todos los profesores registrados.
- Despliegue los datos de los profesores con una estructura consistente.
- Despliegue el horario del profesor si se ha configurado.

Debe contar con un área de administración general que:

- Permita crear usuarios con el perfil de profesor.
- Permita editar o eliminar los datos de un profesor.

Debe contar con un área de administración de usuario que:

- Posibilite a un usuario con perfil de profesor editar sus datos considerando los elementos que se presentan actualmente en la página de perfil de un profesor en el sitio de la Facultad, estos son: Grado, cubículo, teléfono, sitio web, áreas de interés y un campo extra para poner otros datos.
- Permita configurar el despliegue de su horario.

Módulo de sitios

Debe contar con un área accesible por los visitantes que:

- Permita acceder a todas las páginas creadas que se encuentran activas.
- Presente el desplegado de las páginas sobre una misma vista o plantilla.
- Despliegue un menú con los enlaces a las páginas, el menú debe de tener una estructura análoga a la organización de las páginas.

Debe contar con un área de administración general que:

- Permita crear sitios, que serán contenedores de páginas.
- Permita crear páginas y asignarlas a un sitio.
- Permita editar o eliminar las páginas y agrupadores.
- Permita asignar permisos de edición a un usuario sobre uno o varios sitios y consecuentemente sus contenidos.
- Permita modificar la plantilla que se usa para desplegar las páginas.
- Permita crear enlaces a otras URL en lugar de páginas para agregar accesos a páginas externas.

Debe contar con un área de administración de usuario que:

- Permita crear sitios, que serán contenedores de páginas, esto dentro de los sitios que el administrador le ha permitido.
- Permita crear páginas y asignarlas a un sitio sobre el que tiene permiso de edición establecido por el administrador.
- Permita editar o eliminar las páginas y subsitios, esto dentro de los sitios que el administrador le ha permitido.
- Permita crear enlaces a otras URL en lugar de páginas para agregar accesos a páginas externas.

Módulo de artículos

Debe contar con un área accesible por los visitantes que:

- Despliegue un listado de artículos y las áreas a las que están asignados
- Provea una manera de consultar los datos de los artículos contenidos desde otros lugares (como la portada del sitio)

Debe contar con un área de administración general que:

- Permita crear, editar y eliminar artículos
- Permita crear, editar y eliminar áreas para organizar artículos
- Despliegue el listado de artículos de manera completa o aplicando filtros de búsqueda
- Permita dar o quitar permisos de edición a un usuario sobre un área y consecuentemente los artículos que contiene.
- Permita modificar la plantilla que se usa para desplegar los artículos.
- Permita crear accesos a una URL local o externa en lugar de asignar el contenido al artículo.

Debe contar con un área de administración de usuario que:

- Permita crear, editar y eliminar artículos, esto dentro de las áreas que el administrador le ha permitido.
- Despliegue el listado de artículos de manera completa o aplicando filtros de búsqueda, mostrando sólo los artículos de las áreas sobre las que tiene permiso.
- Permita crear accesos a una URL local o externa en lugar de asignar el contenido al artículo.

Módulo de enlaces

Debe contar con un área accesible por los visitantes que:

- Despliegue un listado de enlaces a URL o a documentos para la consulta de información. Estos listados serán usados para publicar archivos y enlaces con información que sea generada normalmente por trabajos de investigación.
- Despliegue en una sección proyectos que son administrados por el CMS y que contengan al menos un documento definido como público.
- Ofrezca el listado en un formato de página HTML para su consulta a través del sitio y en formato RSS para consultar el listado desde un lector de sindicaciones. En ambos casos deben desplegarse también las ligas de las fuentes RSS externas que han sido enlazadas.

Debe contar con un área de administración general que:

- Permita crear entradas en el listado RSS. Las entradas pueden ser archivos o ligas a otro tipo de contenidos.
- Agregue automáticamente elementos al listado RSS al crear o modificar páginas o artículos.
- Permita agregar los listados de fuentes RSS de otras instancias o sitios en general.

Debe contar con un área de administración de usuario que:

- Permita crear un proyecto de trabajo colaborativo
- Permita gestionar a los colaboradores de los proyectos que haya creado
- El mecanismo de trabajo colaborativo debe considerar que los colaboradores pueden ser usuarios que provienen de la misma instancia u otra instancia del CMS.
- Permita solicitar el acceso a proyectos tanto locales como remotos.
- Permita hacer disponibles públicamente para la descargar los archivos de los proyectos que deseen, extendiendo el alcance de documentos de interés entre instancias, además de descentralizar la publicación de documentos generados en un trabajo colaborativo inter-facultativo.

CAPÍTULO 4. DISEÑO E IMPLEMENTACIÓN

Resumen— Se trata todo lo relativo a la implementación del sistema. Se explica y justifica la arquitectura que se utiliza y los mecanismos de funcionamiento que hacen que el sistema trabaje. Se exponen distintos elementos que utilizados como la modularidad implicada en la cohesión y el acoplamiento del sistema o el manejo de patrones de código. Se presentan a estructura del sistema y los mecanismos de interacción entre sus diferentes capas desde el núcleo hasta las API y librerías existentes.

4.1 Bases y notas

Para la construcción del sistema, después de conocer los requerimientos, se puede hacer un análisis previo de la infraestructura necesaria para el desarrollo. Al tener visualizado el alcance se puede definir un marco de trabajo que considere el hardware y software adecuado para la creación del sistema.

4.1.1 Infraestructura

Para el desarrollo y para la ejecución del sistema se involucran dos aspectos; el software necesario para el desarrollo, y los requerimientos en hardware necesarios en las computadoras para correr el sistema.

Software a utilizar

Todo el software que se empleará para el desarrollo es de uso libre. Esta elección tiene ventajas como el nivel de madurez del software a usar, la seguridad de una buena implementación por el número de usuarios que contribuye notificando los errores que encuentran al usarlo y el ahorro de costos.

El uso de software en versiones actuales es por regla general recomendado, nuevas versiones implican mejora en código sobre distintos ámbitos como seguridad, estabilidad o velocidad de procesamiento y respuesta. Se debe considerar que las versiones también dependen del entorno del sistema; más aún, hay casos en los que funciones necesarias para el sistema desarrollado dejan de funcionar correctamente o dejan de ser soportadas. Aprovechando que el sistema es un desarrollo desde cero, se pueden tomar las versiones actuales garantizando que los recursos usados funcionen correctamente; por otro lado, en caso de que existan actualizaciones sobre el software usado, se actualizarán a éstas verificando que el sistema siga funcionando de la manera en que se espera.

El sistema para ser desarrollado hace uso de distintas tecnologías. El sistema planteado se desarrollará bajo el lenguaje de PHP. La base de datos para la implementación a usar será MySQL. La presentación se hará bajo el estándar XHTML estilizada por CSS y en algunos casos con mejoras en funcionalidad por Javascript.

Software a utilizar:

- Servidor HTTP Apache 2.2.4
- Lenguaje de scripting PHP 5.3
- Gestor de bases de datos MySQL 5.0
- Notepad++ 5.6, Gedit 2
- Firefox 6+ (Navegador web)
- MS Internet Explorer 7+ (Navegador web)
- Opera 9+ (Navegador web)
- Apple Safari 3+ Windows (Navegador web)
- MySQL Workbench (Herramienta para MySQL)

La elección de la versión 5.3 de PHP se debe a las nuevas implementaciones y mejoras que se encuentran desde esta versión, dentro de las cuales se pueden mencionar:

  - La existencia de 'namespace' para organizar las clases
  - __autoload(), que permite la carga automática de clases
  - Se permiten llamados dinámicos a las clases
  - La existencia de funciones lambda
  - Clases como Fileinfo soportadas por el núcleo y no como extensión
  - Nuevas clases como DateTime()
  - Función de cifrado bcrypt disponible nativamente
  - A la fecha, oficialmente ya no hay soporte para PHP 5.2

El desarrollo del proyecto se llevará a cabo en una máquina de características estándar que soporte el software mencionado, además de que para hacer pruebas unitarias y funcionales se usará una cuenta en un servidor de la Facultad que tiene las características del servidor final que alojará el sistema.

Los requerimientos mínimos de hardware para servidor son:
Una computadora con un sistema operativo capaz de levantar un servidor web Apache con PHP, un gestor de base de datos MySQL, un editor de código fuente que reconozca los lenguajes sobre los que se trabajarán y con algún navegador web, preferentemente alguno que cumpla con los estándares de buena manera.

Hardware necesario

Los requerimientos mínimos para el servidor que hospede el sistema son:

Por parte del servidor HTTP Apache:
Espacio en disco: 50 MB de espacio libre. Después de la instalación Apache ocupará 15MB aproximadamente, tomando considerando algunos módulos básicos.
Procesador: El uso del procesador es bajo por el tipo de servidor, la orientación del sistema y una interacción baja con la base de datos, siendo que un Intel Pentium IV o superior 2+ GHZ o equivalente puede ser suficiente si se usa un servidor dedicado.
Memoria RAM: Dependiendo de la demanda de clientes al servidor, 2GB puede son suficientes.

Por parte de PHP:
Espacio en disco: 25 MB de espacio libre.
Procesador y memoria: Igual que el servidor HTTP Apache.

Por parte del gestor de bases de datos MySQL:
Espacio en disco: 350 MB de espacio libre.
Procesador: La demanda de procesador por parte de MySQL no es muy alta, un procesador reciente puede hacer la tarea. El proyecto desarrollado supone un nivel de acceso intermedio por la naturaleza del sistema.
Memoria RAM: 1~ GB recomendado.

Las capacidades de los componentes necesarios mínimos para el servidor que hospede el sistema son:

Procesador: Intel Pentium IV a 2 GHZ o equivalente
Espacio en disco: 300MB
Memoria RAM: 1280 MB

Los requerimientos del lado del cliente son mínimos, un navegador que cumpla con los estándares y una conexión a Internet. La diferencia entre un usuario visitante, uno registrado y un administrador, es que mientras el visitante sólo necesita el navegador para acceder a los contenidos, un usuario registrado y el administrador deben preferentemente tener Javascript activo. Javascript se usa para mejorar las funcionalidades disponibles como editores de texto enriquecido para código (X)HTML / PHP o selectores de color entre otros.

4.1.2 Bases arquitectónicas y patrones usados

La implementación del sistema está basada en algunos patrones y arquitecturas. Utiliza un patrón MVC que organiza la estructura de los módulos que componen al CMS. El pa-

trón Singleton se implementa en las librerías y API. La modularización es resultado de la implementación de los módulos bajo el esquema que define framework.

MVC

MVC son las siglas de Modelo-Vista-Controlador. Esta arquitectura lógica separa en tres bloques los elementos de un sistema. Es una manera ordenada de codificar y por ende más fácilmente controlable. Esta separación ofrece varios beneficios al sistema que lo implementa. Primero, tenemos una distribución organizada del código dependiendo de su tipo, todas las funciones y funcionalidades se encuentran en el modelo, mientras que los controladores contienen las llamadas a las funciones del modelo para asignarlas a la vista que se encarga de mostrar la información que se solicita.

Al hacer una petición, el proceso inicia en el controlador. Aquí se recibe la petición (y los parámetros si es que existen); con esa información el controlador sabe qué llamados hacer al modelo para procesar, almacenar y/o extraer datos. Aunque manipula las entradas, solicitudes y parámetros, no realiza un procesamiento propiamente y se limita a hacer las invocaciones que corresponden a la petición y asignar las respuestas a la vista para terminar el proceso.

El modelo es un contenedor de funciones con la lógica del sistema para realizar las tareas para las que ha sido diseñado. Al hacerse la petición, el controlador llama al modelo para que éste conteste después de computar datos para la respuesta requerida. El modelo no tiene información directamente de la vista, a menos que se implemente el patrón observador (externamente) para generar ciertos flujos entre modelo y vista.
Este acercamiento minimiza la cantidad de tomas de decisión que la vista necesita, para que pueda escribir usando código simple (considerando código simple una descripción vaga) [19].

La última parte del proceso termina sobre el elemento de vista. Después de que el controlador ha extraído los datos a través del modelo, los asigna a la vista que se encarga de presentar el contenido. Al tener toda la información, la vista está lista para desplegar el contenido; de esta manera es muy sencillo implementar y adaptar la presentación y en cierta medida los idiomas de despliegue. Dependiendo de la presentación que se desee, se puede cargar una plantilla para generar una página web, o presentar los datos en bruto con formatos como XML o JSON, algo muy útil para servicios web. Hay que notar que aunque la vista no se utiliza para procesar los datos, normalmente contiene lógica simple como condiciones para elementos dependientes de otros o ciclos para presentar conjuntos de elementos.

Singleton

El patrón Singleton es probablemente uno de los mejor conocidos patrones de diseño. Existen muchas situaciones donde se tiene un objeto que maneja una operación centralizada dentro de la aplicación, como un objeto logger por ejemplo. En esos casos es usualmente preferido que exista una instancia del objeto y que toda la aplicación tenga acceso a ella [20]. Se puede agregar que el consumo de memoria es menor, además de compartir de manera transparente los recursos que la clase maneja.

Bajo este patrón existe una sola instancia de la clase, y es la misma clase la que se encarga de controlar la creación de su instancia y el acceso a ella. En el sistema, el patrón Singleton aparecerá repetidas veces, siendo que muchos objetos son naturalmente Singletons, como el objeto de acceso a datos de configuración, el de manejo de imágenes o el de manejo errores.

Modularidad

La modularidad es la eliminación de acoplamientos fuertes entre las diferentes partes del sistema manteniendo una alta cohesión entre cada componente. Se considera como módulo a un subconjunto de soluciones con un nivel muy alto de relación para solucionar una serie de problemas de manera integral; en este caso el problema es la necesidad de un administrador de contenidos para la Facultad de Ciencias de la Computación, la solución integral es el gestor per se, y un módulo, por ejemplo, el conjunto de funcionalidades para administrar páginas solamente.

La modularidad en un sistema permite que en su ejecución no tengan que estar integrados todos los módulos desarrollados, y que el núcleo del sistema pueda ejecutar lo que se encuentra disponible (y en caso de ser necesario tener un manejo especial para las peticiones a módulos no existentes).

La capa que contiene la implementación del framework, considera la separación por módulos al implementar las funcionalidades que ofrecerá el sistema, esto puede permitir que por un lado sea fácil la implementación de nuevas funcionalidades, y por otro, que la actualización del sistema pueda hacerse de manera parcial, sólo en el módulo involucrado (o involucrados en caso de que el cambio influya en una API en uso).

Base de datos y normalización

La base de datos debe de contar con la estructura adecuada para poder gestionar la información de manera correcta y eficiente. Si la base de datos presenta anomalías se

pueden ocasionar problemas como el incremento injustificado de complejidad en las consultas para extraer la información bajo diferentes condiciones (y decrementando el desempeño) y el fomento de datos redundantes; la normalización de la base de datos soluciona estos problemas.

Los niveles de normalización solucionan de manera incremental los problemas estructurales de la base, es por eso que el sistema debe tener una base de datos normalizada para un manejo correcto. Los módulos implementan sus tablas respetando la quinta forma normal. Al observar las figuras del esquema de la base de datos se pueden notar algunos pocos elementos que pueden ser nulos.

Como varios autores señalan, la existencia de nulos sobre las tablas son válidos bajo ciertas circunstancias y reglas [21], [22], [23]. Los valores nulos no deben ser elementos de llaves candidatas ni llaves foráneas de una tabla derivada de una relación muchos a muchos puesto que sólo la entrada completa tiene sentido en la tabla; los valores nulos son posibles para estados indefinidos temporales (como un campo para guardar una fecha de actualización), o para los elementos que han perdido una relación de padre pero contienen información que se desea o necesita conservar (como un artículo o página). A esto se puede agregar una de las reglas del modelo relacional definidas por el Dr. Codd: El manejador requiere el soporte para el manejo de información faltante, o no aplicable que es sistemáticamente diferente de valores regulares [24].

4.2 Arquitectura de sistema

El CMS tiene una arquitectura basada en los paradigmas descritos en la sección 4.1.2.
El sistema está compuesto por:

- Un núcleo, que es un ambiente base donde se preparan constantes de acceso y funciones básicas.
- Un conjunto de librerías que dan diferentes tipos de servicio.
- Un manejador de información MVC que procesa la petición del usuario, accede a la lógica de operaciones y presenta el resultado.
- Un conjunto de módulos desarrollados sobre la capa MVC donde:
  - Existen diferentes modos de ejecución. Esto es, un usuario que entra al sitio generado por el sistema tiene la vista común (modo de aplicación), un usuario registrado puede acceder a áreas especiales y ejecutar funciones particulares sobre los módulos a los que tiene acceso permitido (modo de usuario), un usuario con el acceso de administración definido al instalar el sistema tiene acceso a un área para configuraciones y gestión de usuarios y grupos (modo de administración).

  - Existe un método de acceso a librerías API para dar información al exterior acerca de sus datos.
- Un filtro de acceso para permitir o bloquear acceso a zonas de administración.

En el esquema de la Fig. 4.1. se plasma la estructura del sistema a desarrollar. El framework estará formado por un núcleo que preparará el ambiente de ejecución y filtrará las peticiones de acuerdo al tipo de usuario. La estructura MVC define la manera en que se construyen los módulos, y estos últimos demás utilizan el conjunto de librerías para tener acceso a los datos almacenados.

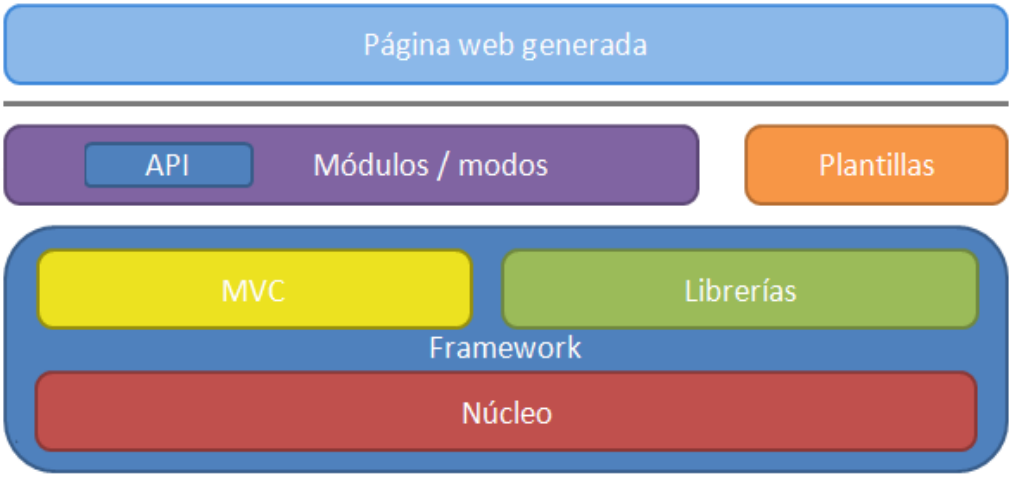

Fig. 4.1. Estructura general del sistema

Los módulos construidos contendrán en algunos casos una API, de manera que pueda existir un mecanismo de comunicación modular, y las plantillas son usadas para desplegar finalmente el sitio con los contenidos de los módulos en un formato único y más fácilmente administrable.

4.2.1 Framework

Un framework es un marco o entorno de trabajo que facilita el desarrollo ordenado y es la parte fundamental del sistema. Al implementarlo, se tendrá una base con un esquema regular para crear los módulos requeridos.

En esta capa del sistema se define el modelo MVC creando una clase para cada uno de los elementos del modelo, una clase para el controlador, otra para el modelo y una más para la vista. Las clases del controlador y el modelo son clases padre que nunca son ins-

tanciadas directamente sino que heredan a las clases de los módulos sus funciones y variables. La clase control por ejemplo, contiene: un constructor que genera un objeto del modelo del que se obtendrá información; un método set para definir variables para la vista, variables que son llenadas con llamados al modelo; un método para definir la acción a ejecutar, etc. El modelo por otro lado tendrá funciones para acceder a archivos de idioma o para generar cadenas de texto para la inclusión de archivos de JS o CSS en la vista. La clase de la vista es utilizada por el controlador, que de cierta manera inyecta la información, y el código de la vista define la manera en que la información se despliega.

Las librerías son el otro elemento necesario para el framework. Éstas contienen todas las funciones de uso general para construir los módulos; las podemos ver como un elemento intermedio entre el código puro y un modulo implementado; estandarizan, simplifican y facilitan la forma de acceder a los recursos. Las librerías están estructuradas bajo el patrón Singleton.

4.2.2 Modularización

La arquitectura contempla una división en módulos; éstos tienen una estructura heredada del MVC definido en el framework, de forma que, por un lado se asegura una estructura única para la creación de módulos, y por otro, se automatiza la herencia de métodos e información de sus meta-clases (control y modelo).

Cada módulo es un conjunto de funcionalidades relacionadas, sin embargo, dependiendo del tipo de usuario es que se permite o no el acceso a estas funcionalidades. La manera de controlar el acceso es mediante la creación de modos en los módulos. Un modo es la manera en que se está trabajando en un momento determinado con un módulo. Como se ha mencionado, se consideran el modo de aplicación que no filtra el acceso, y el modo de usuario y de administración que requieren de permisos en el usuario. Usar esta técnica resulta práctico porque de esta manera se pueden crear nuevos modos en caso de necesitar otros tipos de servicio; ejemplo de esto sería un modo webservice que permita comunicarse directamente con otros sistemas.

Aunque los módulos no estén integrados directamente, pueden interaccionar entre sí mediante el uso de una interfaz, conservando así su acoplamiento débil. Para conseguir la comunicación entre módulos se utiliza una API desde la que cada módulo que pretenda exponer la información que gestiona (o más propiamente parte de ella) publica una serie de métodos; de esta manera se puede obtener información de otros módulos sin tener conocimiento de la estructura de código y tablas de las bases de datos donde está almacenada.

La estructura de las API está estandarizada para tener un mecanismo simple, común y automatizado de acceso. El flujo de los datos a través de la API es simple y casi siempre directo, y sigue la misma mecánica que las librerías.

4.2.3 Librerías y acceso a recursos

El sistema hace uso constante de librerías para el funcionamiento. Las librerías contienen el acceso a los diferentes recursos disponibles. La creación de librerías para el manejo de funciones es muy práctica pues agrupa a las funciones de la misma índole, facilitando el mantenimiento y administración a nivel desarrollo.

A través de las librerías la capa del framework expone funciones a los módulos. Con este enfoque, las librerías deben de dar acceso a recursos como información de ubicaciones y acceso a archivos y directorios, acceso e interacción con la base de datos, consulta de la configuración del sistema, funciones para loggers o manejadores de errores, procesamiento de imágenes, manejo y procesado de contenidos, etc.

4.2.4 Objetivos y limitantes de la arquitectura

El objetivo principal de la arquitectura definida es obtener un sistema que con simplicidad y un bajo impacto en recursos, pueda realizar las tareas de un manejador de contenidos que una facultad como la Facultad de Ciencias de la Computación necesita.
Otro objetivo es obtener un sistema que al estar divido en capas pueda ser actualizado de manera sencilla en sus diferentes niveles; que permita poder afinar el framework para incrementar la responsividad sin tener que tocar los módulos, afinar un módulo, agregándole nuevas funcionalidades internas o a su API, o crear nuevos modos de ejecución si es necesario. Esto con el fin de tener un sistema fácilmente editable, escalable y actualizable para que a futuro nuevos desarrolladores puedan adaptar el sistema a nuevas necesidades.

Por otro lado, las limitantes son que al estar construido el sistema sobre un framework, las modificaciones al nivel de módulos están sujetas a las reglas sobre las que está diseñado el sistema para ser implementado, esto significa que para la implementación de API y librerías, debe seguirse el patrón establecido para ellas. Además, la estructura del framework que soporta a los módulos no puede cambiar drásticamente la manera en que responde las solicitudes porque esto implicaría modificar la manera en que están implementados todos los módulos.

4.2.5 Descripción de la arquitectura

La descripción de la arquitectura utiliza el modelo de vista de arquitectura de software "4+1". Estas múltiples vistas permiten localizar de manera separada las problemáticas identificadas desde diferentes ángulos por diferentes actores involucrados en la necesidad del sistema, como el usuario final, el grupo de desarrollo e ingeniería de software, etc., y manejar de manera separada los requerimientos funcionales y no funcionales [25].

Arquitectura lógica

Esta arquitectura define una descomposición orientada a objetos. El objetivo principal es plasmar en términos de servicios la funcionalidad al usuario.

Los elementos para describir esta arquitectura se muestran en la Fig. 4.2.

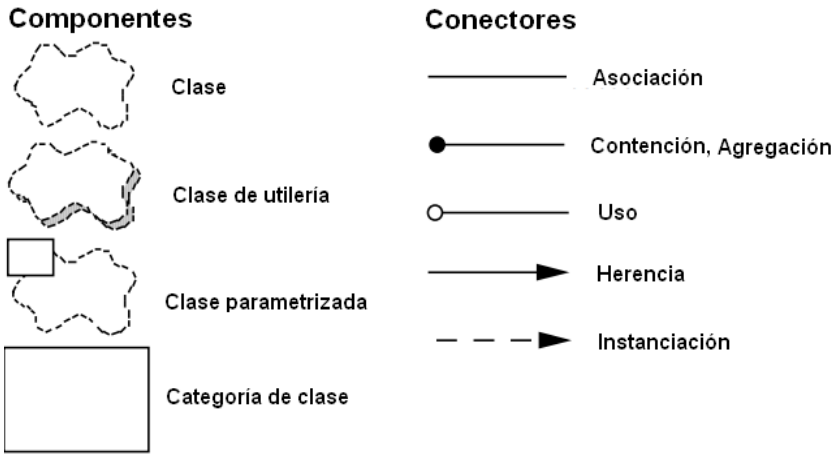

Fig. 4.2. Elementos para describir la arquitectura lógica

La arquitectura lógica del sistema se indica en la Fig. 4.3., a continuación se comenta su diagrama: La base del sistema es la encargada de crear un ambiente para poder ejecutar la aplicación. Comprende una serie de constantes que definen rutas de acceso, la definición de funciones esenciales para el funcionamiento del sistema, y la definición del flujo a seguir (único a nivel de usuario) para poder acceder a las funcionalidades del CMS.
Cuando se realiza una petición, se crea un mapeo de acceso que identifica el módulo, el modo y la acción que se requiere y se continúa el proceso pasando por el filtro de acceso.

Las base o núcleo se comunica con un filtro o control de acceso, para permitir, bloquear o pedir los datos de acceso dependiendo de la zona a la que se quiere entrar. Este es un script que permite o niega el acceso con base en las credenciales provistas.

El controlador crea una instancia de su modelo y vista correspondientes. El control recibe entre otros parámetros la acción a realizar (y algunas veces el tipo de despliegue de la vista). Al realizarse la acción del controlador se hacen peticiones a diferentes funciones del modelo, y en menor medida el llamado directo a una librería o API del actual u otro módulo. El modelo por su parte es una librería con funciones específicas del propio módulo, por ejemplo en el módulo de profesores, una función del modelo sería la función de guardar los datos de un nuevo registro de algún profesor. El modelo normalmente es el que hace uso de las librerías del sistema y/o API disponibles.

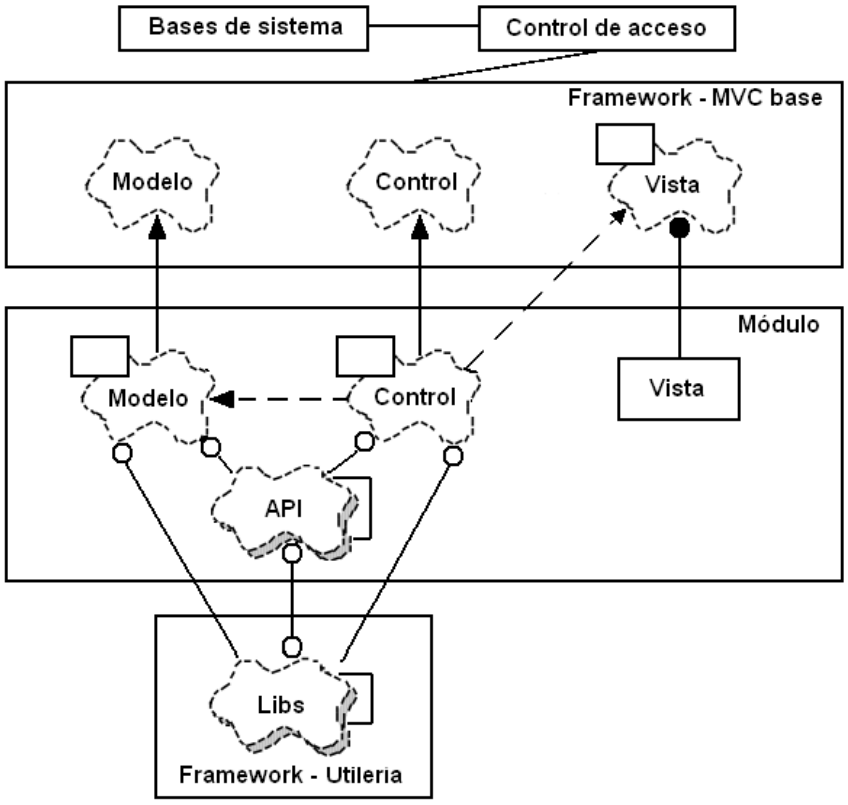

Fig. 4.3. Arquitectura lógica del sistema

La diferencia fundamental entre una librería de sistema, un modelo de módulo y una API es que la librería de sistema proporciona acceso a recursos del sistema, son concebidas con una finalidad de uso general; el modelo contiene funcionalidades propias del módulo que, o son funciones que sólo pueden ser requeridas por el módulo, o son funciones para almacenar datos; mientras que las funciones de una API son de interés ex-

terno aunque sean funciones de un módulo, una función de la API del módulo de profesores puede ser el listado de profesores o la obtención de los datos de un catedrático en particular, estas funciones pueden usarse para desplegar información básica propia de un módulo desde otro, o incluso desde una plantilla.

El controlador hace uso de las funciones y almacena los resultados para que sean utilizados por la vista. La vista muestra o no la plantilla dependiendo de los argumentos enviados al crearla y despliega la información almacenada resultado de la ejecución de diversas funciones. La vista tiene pequeños ciclos y evaluaciones que permiten obtener y desplegar datos de manera dinámica.

Bajo ese esquema vemos que tanto las librerías como las API tienen una relación entre ellas y de manera recíproca puesto que ambas tienen la capacidad de usar como recurso cualquier clase de estos dos tipos.

Arquitectura de desarrollo

En esta arquitectura se cubre la organización a manera de módulos del sistema, describiendo los elementos en paquetes. El diagrama usado tiene un esquema basado en capas en la que cada una tiene una responsabilidad bien definida. La manera en que se apilan las capas representa claramente las dependencias que existen entre ellas, posicionando una capa sobre otra de la que depende (además de poder depender de elementos que se encuentran en la misma capa). La arquitectura de desarrollo también puede dar una idea de cómo se gestionará el código.

Complementación	5 Manejo de estilos y funcionalidades JS	IMPLEMENTACIÓN (CMS)
Módulos	4 Componentes CMS	
MVC	3 MVC web Estructura base de módulos	FRAMEWORK
Librerías	2 Funcionalidades de manejo de datos específicos, acceso a información	
Núcleo	1 Preparación de entradas, ambiente básico Funciones y filtros	

Fig. 4.4. Arquitectura de desarrollo

Descripción de las capas del sistema plasmado en la Fig. 4.4.
El núcleo contiene los elementos básicos que preparan el ambiente para la ejecución del sistema. Al realizar una petición al sistema es en el núcleo donde se inicia el flujo de trabajo. El núcleo se encarga de recibir la petición de servicio, de cargar los elementos para hacer funcionar a los módulos, y de cargar la clase, acceder al modo indicado y ejecutar la acción requerida. También tiene la tarea de filtrar el acceso a las diferentes áreas dependiendo del tipo de usuario.

La capa de librerías provee un conjunto de funcionalidades operacionales. Dentro del framework las librerías son usadas para el acceso a los recursos en la aplicación. Las librerías forman un entrono extensible y mantenible donde se concentran la mayoría de funciones necesarias para soportar el framework. Esta capa es útil para hacer mapeos a diferentes implementaciones de un mismo servicio (como el acceso a una base de datos).

La estructura MVC establece la manera en que los módulos se deben de construir y cómo interaccionan de manera interna (cómo el controlador hace llamados al modelo dependiendo de la petición y cómo provee a la vista la información que debe mostrar).

Las clases con el componente MVC serán las siguientes:

Clase de controlador (control.class.php) que crea las instancias de los otros dos elementos del modelo y contiene aliases a funciones de la clase del modelo y la vista, y el llamado a la función de desplegado de la vista. El objeto controlador contiene un método abstracto que define a la acción del controlador a llamar si ésta no se indica.

Clase de modelo (model.class.php) que contiene las funciones para definir la información para la vista. Este elemento almacena las funciones que se definen en los módulos.

Clase de vista (view.class.php) que contiene principalmente la función usada por el controlador para definir contenido sobre la vista y un método para desplegar la información. Este método carga la plantilla sobre una variable a la que se le agrega toda la información que ha sido generada como respuesta a la solicitud, de modo que puede ser utilizable como un punto de extensión que permita manipular la respuesta final.

La capa de módulos se refiere a la implementación del CMS sobre el framework. Esta capa toma la estructura especificada en el MVC y hace uso constante de las librerías para conseguir sus fines, esto implementando el o los modos que sean necesarios.

Los módulos tienen archivos obligatorios y opcionales; los obligatorios definen propiamente las funcionalidades y comportamientos, mientras que los complementarios, ofre-

cen funcionalidades extras, o dan información acerca del módulo. La capa complementaria da dinamicidad, mejorando la experiencia del usuario. Aunque no es obligatoria, con ella se facilitan tareas y se hace más agradable la interacción con el sistema. Esta capa afecta a los tres modos en el aspecto de estilos, dando una mejor apariencia y accesibilidad al usuario; el uso de Javascript está reservado en este momento en los modos de usuario y administración, mejorando funcionalidades al momento de editar y manejar algunas configuraciones.

Arquitectura de proceso / Diagrama de secuencia

Para describir la arquitectura de proceso, se usarán diagramas de secuencia, con estos diagramas se puede exponer la manera en que se van realizando los eventos desde que se recibe la petición de un usuario hasta la respuesta regresada.

El diagrama de la Fig. 4.5. expone la vista general del proceso de ejecución del sistema. De esta manera se puede entender a grandes rasgos la interacción entre los diferentes bloques que forman el framework, que a su vez forma al CMS.

1) El evento inicial corresponde al usuario haciendo una petición al sitio, esto a través del navegador, solicitando de manera normal la URL. La petición la recibe el núcleo que la separa para procesarla (identificando el módulo requerido, extrayendo el modo, la acción a ejecutar y los parámetros).

2) Al identificar el módulo, le hace un llamado enviándole la información que recibió previamente.

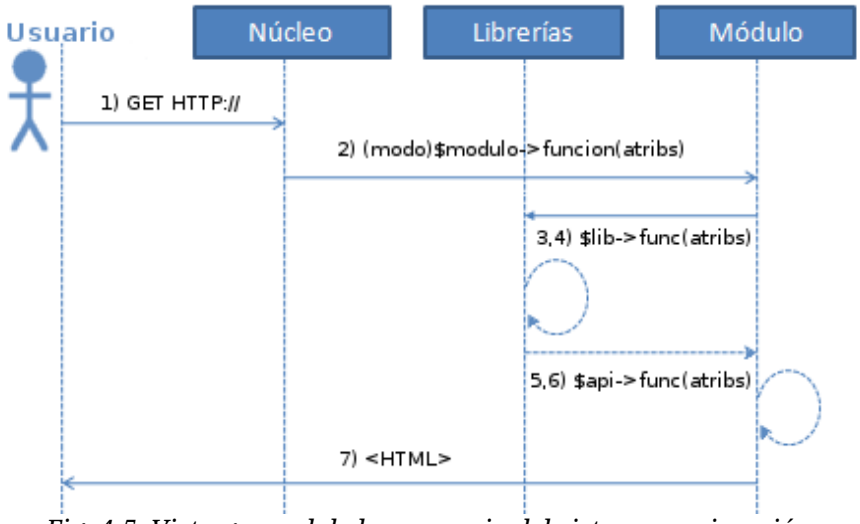

Fig. 4.5. Vista general de la secuencia del sistema en ejecución

3,4) El módulo hace una serie de llamados a las librerías para acceder a los recursos que necesite, estas librerías a su vez pueden hacer llamados a otras librerías.

5,6) El módulo puede hacer llamados también a su API o a las API de otros módulos si lo necesita.

7) Después de obtener todos los datos regresa la información generalmente dentro de una plantilla.

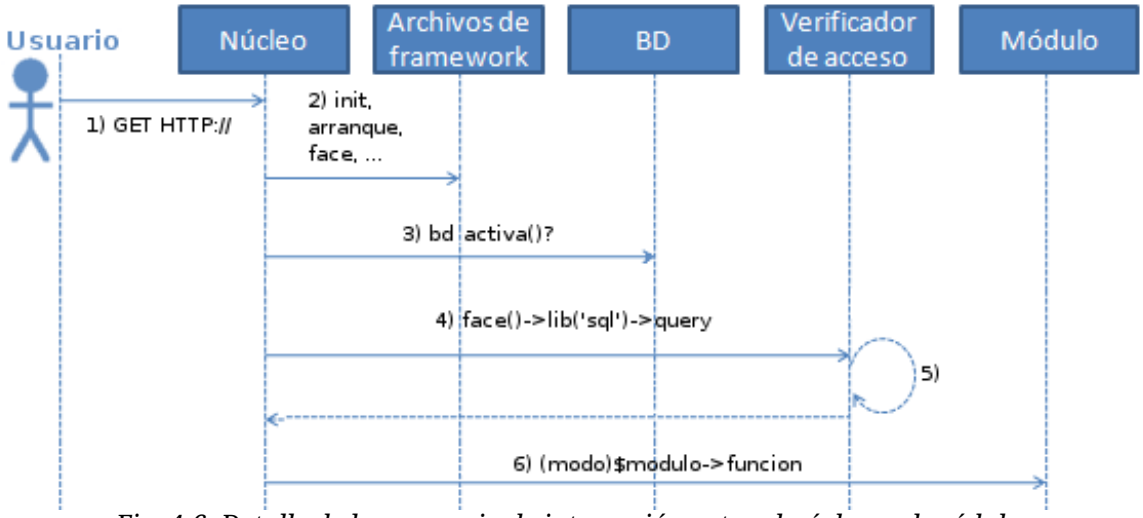

Fig. 4.6. Detalle de la secuencia de interacción entre el núcleo y el módulo

En el diagrama de la Fig. 4.6. podemos observar el detalle del proceso que se realiza entre el núcleo y el módulo.

1) El usuario hace la petición.

2) El núcleo al recibir la petición, comienza a levantar el ambiente de ejecución, esto lo hace cargando los archivos de inicio del framework, que incluyen el archivo de disparo de proceso inicial y otros archivos de entorno.

3) Después de cargar el ambiente correctamente, se verifica el servicio de base de datos.

4) Con el acceso a la base, se accede a un filtro de verificación de acceso.

5) Si el usuario intenta acceder a una zona que necesita una identificación y el usuario no ha proporcionado los datos necesarios, despliega un formulario de acceso, que al recibir las credenciales las valida y en caso de no ser correctas cíclicamente despliega el mismo formulario para repetir el proceso de autenticación.

6) Finalmente, cuando el usuario se ha identificado, si el acceso es permitido se ejecuta la acción del módulo solicitada en el modo pedido; en caso contrario se niega el acceso.

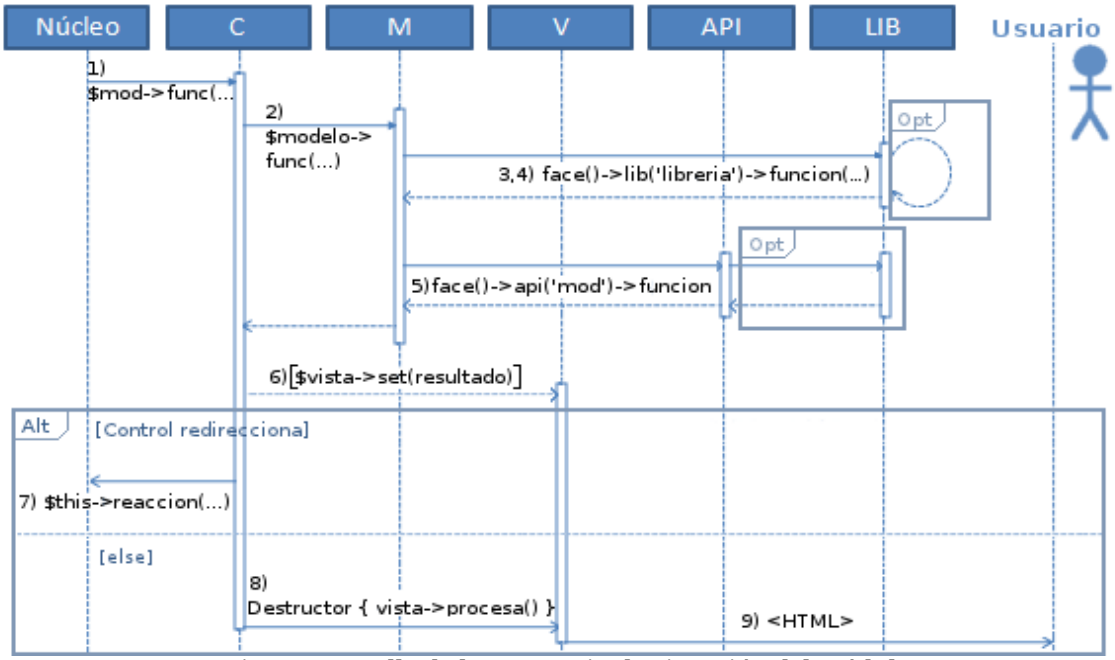

Fig. 4.7. Detalle de la secuencia de ejecución del módulo

El diagrama de secuencia de la Fig. 4.7. corresponde a las acciones realizadas por los módulos siguiendo la estructura MVC.

1) Cuando se ha identificado el módulo y el modo a ejecutar se hace un llamado al controlador. El controlador es el que se encarga de hacer ahora los llamados subsecuentes al modelo.

2) El controlador hace una serie de llamados al modelo. El número de estas solicitudes depende de la acción. Cada llamada sirve para insertar u obtener información, la información obtenida en llamada es agregada a la vista.

3,4) El Modelo comienza a hacer una serie de llamados a las librerías para extraer información. Esa información es procesada; además de la información que se obtiene, se

pueden usar las librerías para almacenar más datos. Las librerías también hacen llamados en algunos casos a otras librerías, aunque esto depende del nivel de abstracción sobre las que están construidas.

5) Para obtener información, el modelo además de valerse de las librerías, puede utilizar las API existentes tanto del mismo módulo, como de otros.

6) El controlador, cuando recibe los datos los inserta en la vista. Éste es el fin de un ciclo de obtención de datos, que se repite por cada uno de los llamados al modelo o a la o las API.

7) El módulo, al nivel del controlador, al terminar de realizar su tarea, sea de inserción, obtención de datos o ambas, puede redireccionarse hacia otra acción, o

8) el objeto del controlador se elimina, y es dentro de su destructor donde se hace el último acceso a la vista para que ésta termine el proceso.

9) La vista con todos los datos almacenados está lista para regresar una respuesta. La vista puede hacer un procesamiento final de los datos, agregarlos a una plantilla. La plantilla a su vez, es capaz de hacer llamados a las API para presentar información de algún modulo de manera persistente.

Escenario de ejecución

Al observar la arquitectura en las vistas anteriores, tenemos una visión de la mecánica que se sigue al tener el sistema en marcha y la manera en que reacciona ante una solicitud de servicio. Con esta visión podemos entender de manera sencilla un escenario de ejecución. Este escenario tiene una relación y un razonamiento similar al diagrama de secuencia. Con el siguiente escenario podemos tener una vista técnica del comportamiento del sistema al momento de realizar el proceso.

El esquema de la Fig. 4.8 representa el flujo del proceso generado al hacer una petición al CMS. Este evento se presenta tanto al acceder al sitio por primera vez, como al ir navegando de página a página por los enlaces que dirigen a acciones de módulos. El flujo de proceso inicia cuando con la petición (1) que se dirige al proceso de creación de entorno, procesando la solicitud, como podría ser el listado de profesores.

Se establece la política de desplegado de errores, se limpian las variables globales, se hacen disponibles varias funciones y se verifica el permiso de acceso.

Después de tener lista la base es momento de invocar al módulo (2). Al pasar el control de acceso se continúa con la petición cargando el módulo indicado en la petición. Para que esto sea posible, los módulos deben utilizar la estructura definida en el MVC del framework. Básicamente, cada módulo hereda del MVC las características mínimas de funcionamiento, extendiéndolas para dar forma a cada una de las acciones del CMS.

Fig. 4.8. Escenario común al solicitar una página al sistema

Al crearse un objeto controlador ,este inicializa sus variables, entre las que están, un objeto de la vista y otro del modelo que le corresponde. El controlador toma la acción a realizar, y con base en ella hace los llamados (3) a las funciones del modelo correspondiente, y en menor medida a librerías y API si es que los datos que se obtienen no tienen que ser procesados.

Normalmente quien recibe las solicitudes es el modelo y éste hace uso de las funciones de librerías y API. La invocación regresa los datos al controlador (4). El controlador ahora inserta esos datos en una estructura de la vista para ser usadas posteriormente (5).

El objeto vista se encarga de procesar (6) los datos que se han conseguido para presentarlos de una manera legible. Este objeto tiene el método `set()` que usa el controlador para guardar resultados. El proceso termina con la llamada a la función `render()` que obtiene el valor de las variables y las integra a la plantilla que se mostrará.

4.2.6 Estructura del sistema

El árbol de directorios del sistema se muestra en la Fig. 4.9.

Fig. 4.9. Estructura de los directorios del sistema

/. El directorio raíz contiene la estructura general en seis directorios: cache, foro, framework, módulos, public y repo; cada una con un conjunto de directorios y archivos bien definidos.

/cache Es un directorio para almacenar elementos temporales, en este caso por las fuentes RSS externas cargadas por el módulo correspondiente dentro de la carpeta /ca-

che/rss. A futuro puede ser usada para almacenar otros elementos temporales, como el resultado de funciones que procesen mucha información.

/foro La aplicación incluye como sistema de foros a punBB, que se encuentra en esta carpeta. PunBB es un sistema pequeño, rápido y poco intensivo gráficamente -con marcado sencillo y poco manejo de medios como imágenes para la presentación- que además tiene una API para extraer información desde el exterior.

/framework Contiene a todos los elementos que en conjunto forman el framework.

/framework/inst Directorio con los scripts de instalación.

/framework/lib Contiene las librerías PHP del framework y otras librerías externas extras, como la librería Snoopy para conectarse remotamente a otros sitios o la librería PasswordHash para el manejo de contraseñas.

/framework/lng Directorio reservado para sistema de idiomas.

/framework/mvc Contiene los archivos de la estructura MVC, un archivo por clase, donde cada clase modela a un elemento de esta arquitectura.

/framework/sys Contiene archivos del núcleo necesarios, entre ellos los archivos con funciones de arranque, de manejo automático de clases y de validación de acceso. Estos archivos no tienen una estructura específica.

/modulos Esta carpeta contiene todos los módulos que pertenecen al sistema.

/public Contiene a los objetos de acceso público, como plantillas, imágenes, scripts, etc.

/public/js Contiene a los plugins y scripts usados en el sistema, como los editores de texto enriquecido, validadores de formularios o selectores de color.

/public /templates Contenedor de las plantillas de las vistas.

/repo Almacén de archivos en general. Los elementos que se guarden aquí pueden a futuro ser gestionados por diferentes partes de la aplicación.

/repo/img Contiene a las imágenes subidas por los usuarios o el administrador.

4.2.7 Archivos, patrones, organización de código

La relación entre la arquitectura del sistema y la distribución de código en los archivos está directamente ligada. La carga de código de manera dinámica utiliza el esquema de archivos y directorios para poder funcionar de manera automática. Además, para identificar fácilmente la ubicación del código del sistema se usan dos elementos: su ubicación en los directorios, que es la manera ordenada en distribuir dependiendo del tipo de función, y la extensión en el nombre del archivo, que define el tipo de contenido.

Las extensiones usadas son:

nombre.php
nombre.abstract.class.php
nombre.lib.class.php
[nombre.]api.class.php
mod_nombre.sql
xx.lng.php
mod_nombre.json.php
inst.php
pag.modv.php
mod.info

Para más información acerca de la relación entre extensiones y tipos de código se puede consultar el apéndice 2, *Tipos de archivo*.

Las librerías son un conjunto de clases que forman una nueva capa de abstracción para el desarrollo del CMS y son una parte muy importante del framework. Las librerías existentes son:

cache.lib.class.php
config.lib.class.php
dir.lib.class.php
errors.lib.class.php
FileUploader.class.php
img.lib.class.php
mod.lib.class.php
PasswordHash.lib.class.php
phpDoc.lib.class.php
punBB.lib.class.php
sql.lib.class.php
valida.lib.class.php

Para más información acerca de cada librería se puede consultar el apéndice 3 *Librerías*.

Comentarios dentro del código.
Para los comentarios dentro del código se usa el estilo PHPDoc cuando es necesario. Esta sintaxis está basada en el sistema JavaDoc de SUN y define una manera limpia y ordenada de documentar el código, facilitando el entendimiento. El código PHPDoc tiene un uso particular en los archivos de controlador de los módulos. Con éstos se indican los nombres asignados a las acciones y las hace visibles.

4.2.8 Seguridad

La implementación de la seguridad está sobre las capas del framework y sobre el CMS; donde el código puede fallar o ser atacado, donde se realizan accesos a recursos y al manejar archivos.

En la capa de framework la seguridad contempla diferentes elementos: el acceso a zonas restringidas, la existencia de funcionalidades de PHP que puedan comprometer los datos que maneja el sistema, la de inserción de datos (en esta última se ve involucrada también la capa de CMS) y el cifrado de contraseñas (gestionado por el componente de usuarios).

En el proceso de ejecución que se sigue tras una petición -con el flujo expuesto en los diagramas de secuencia de la arquitectura y en el caso de uso-, se cuidan diferentes elementos y situaciones. Al acceder al sistema, en la fase de preparación, se definen funciones que controlan la integridad de los datos enviados como parte de la petición, y que son ejecutadas para preparar las entradas.

Las funciones `stripSlashesDeep()` y `removeMagicQuotes()` son utilizadas para limpiar los datos dentro de `GET`, `POST` o `COOKIE` para controlar un posible ataque por inyección SQL. La función `unregisterGlobals()` es usada para que en caso de que `register_globals` esté activado, se eliminen las posibles redefiniciones de las variables por métodos externos como `GET` o `POST` [26], [27]. Posteriormente al proceso de limpieza se realiza el proceso de ingreso/comprobación de datos para permitir o denegar el acceso; este proceso se realiza dentro del núcleo al igual que las acciones anteriores. El método que inserta las respuestas del modelo para desplegarse en la vista pasan por un proceso de limpiado de caracteres especiales HTML para evitar errores por caracteres XHTML no válidos.

El diagrama de la Fig. 4.10 muestra el flujo de acceso a una zona con permisos requeridos.

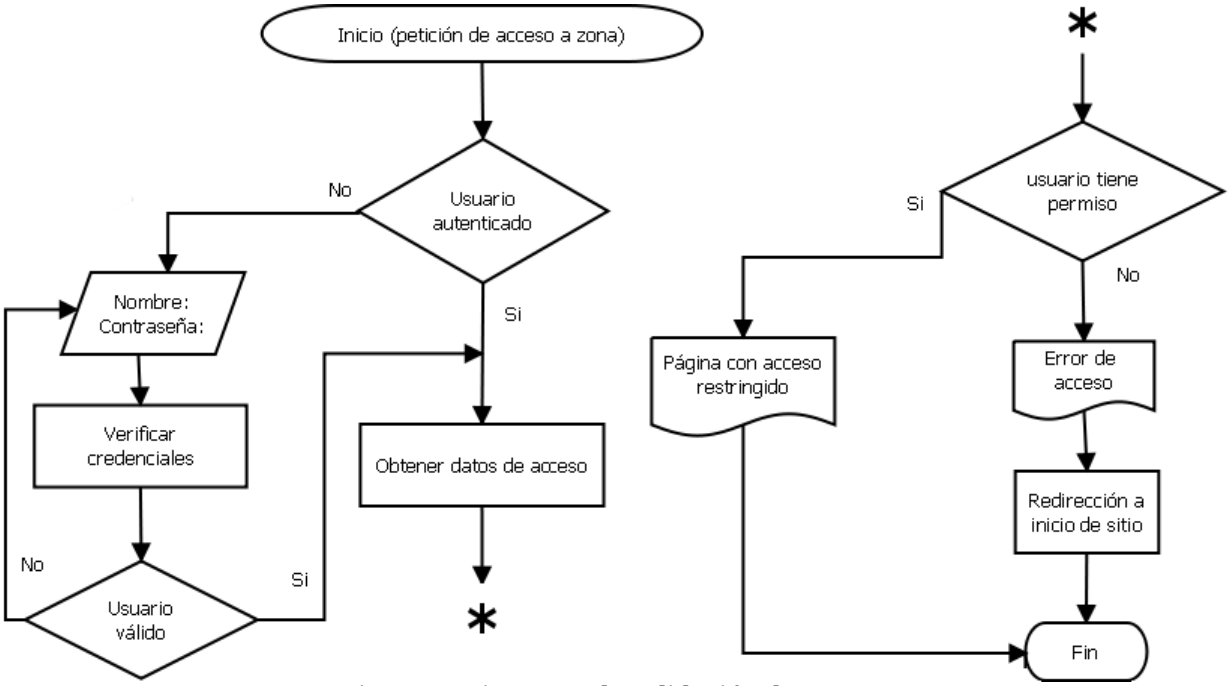

Fig. 4.10. Diagrama de validación de acceso

Seguridad de código usando clases

PHP5 está orientado a objetos. Este paradigma ofrece varias ventajas.
Una de las características principales explotables es la manera en que se encapsula el código que contiene una clase. Las clases tienen el mérito de que su código no es visible ni ejecutado hasta que la clase es invocada [28], así cualquier cadena con una contraseña o ruta de archivo no estará disponible hasta que sea necesaria.

La consistencia entre inclusiones

Es común en sistemas grandes la inclusión de archivos de manera insegura. Llevar el control de inclusión de archivos algunas veces es complicado cuando el sistema tiene un tamaño considerable y es posible que se cargue un archivo que anteriormente había sido cargado.
Este no es un fallo al exterior, sino un posible generador de errores internos, puesto que la inclusión de un archivo puede reiniciar variables que tenían información procesada, disparando errores en inconsistencia de datos.

La inclusión de archivos es delegada al framework, que incluye dinámicamente y bajo demanda los archivos necesarios, de esta manera se tiene también un control sobre las funciones de inclusión evitando en cierta medida el uso de las funciones _once que son de rendimiento menor.

Manejo de cifrado en contraseñas

Diseñar un método de cifrado efectivo es una tarea que puede parecer sencilla, sin embargo implementar un algoritmo realmente efectivo es difícil [29], y con un costo alto de ejecución si se utiliza un lenguaje interpretado y/o que no utiliza funciones directamente del sistema operativo; esto por otro lado podría ser aprovechado para implementar el mismo algoritmo en un lenguaje como C y conseguir ataques veloces. La idea es utilizar un algoritmo que sea fiable y que ofrezca un buen nivel de seguridad; la velocidad propia del algoritmo por otro lado, debe ser relativamente lenta de modo que el uso por el sistema no se vea afectado pero que incremente considerablemente el tiempo que le tomaría a un programa generar un ataque.

PHP ofrece la posibilidad de utilizar algunas funciones de hash directamente del SO si existen, y desde la versión 5.3 se incluyen también algunas funciones de hash compiladas dentro del núcleo de PHP en caso de que las de sistema no estén disponibles y que tienen mayor rendimiento sobre una implementación manual.

El mecanismo de hash usado para proteger las contraseñas es una adaptación de PHPass (http://www.openwall.com/phpass/), que es un pequeño framework de hasheo empleado en varios CMS como Drupal, WordPress, bbPress, Vanilla y otros por su fiabilidad. PHPass provee una capa de abstracción sobre las funciones de hash, incluyendo generación de sal aleatoria y proceso de estiramiento [30]. La adaptación sólo permite el uso de la función bcrypt, que es la mejor opción disponible del framework.

Bcrypt es una función de hash bien aceptada[31], [32] basada en el algoritmo de cifrado de Blowfish modificado Eksblowfish, que incrementa el procesamiento necesario y que permite definir un nivel de costo para la generación de las llaves [33].

De modo que:

- Con las contraseñas obligatoriamente de mínimo 8 caracteres, con una mayúscula y minúscula, números y caracteres especiales; se tienen al rededor de 3.59×10^{15} posibles combinaciones mínimas
- Se pueden generar al más de 5.4×10^{39} posibles *hashes*, por cada password gracias a la sal aleatoria, imposibilitando el uso de rainbow tables precalculadas
- El proceso de enrobustecimiento por iteraciones por defecto implica poco más de cuatro mil veces la necesidad de cómputo, Evitando ataques offline de fuerza bruta
- El algoritmo es adaptable para incrementar la cantidad de ciclos en caso de que se tenga una mayor capacidad de procesamiento por parte del servidor sin dejar de ser compatible con contraseñas generadas anteriormente
  Para más detalles de los números se puede consultar el apéndice 5, *Protección de contraseñas*.

Integridad y seguridad en base de datos

Para salvaguardar la información de la base de datos se necesita trabajo tanto de la librería que interacciona con la base de datos como con los módulos que ordenan la inserción y actualización de los mismos. El trabajo en conjunto es necesario porque a pesar de que la librería podría por sí misma limpiar toda la información recibida para asegurarse de no insertar información con ciertas estructuras, hay casos donde se podrían generar falsos positivos (como en la inserción de datos binarios de una imagen, o en códigos de programación).

El módulo que realiza una inserción o actualización debe de considerar que si los datos provienen de una fuente que no es confiable (como datos de un formulario), debe de indicar que los datos deben ser limpiados antes de almacenarse. El proceso es sencillo de implementar y en código sólo es necesario que al hacer el llamado a la función de inserción/actualización la consulta enviada no contenga los datos directamente, sino identificadores que indiquen dónde se insertarán los datos que se envían como parámetros extras. De esa manera se indica explícitamente cuáles de los datos deben ser limpiados, las variables que no necesiten pasar por ese proceso se pueden insertar directamente, evitando por un lado inyecciones SQL y por otro procesamiento extra en datos que son de una fuente confiable o que pueden ser erróneamente escapados.

4.3 La mecánica del framework

La base del sistema, el framework, está formado por un conjunto de archivos relacionados que en conjunto ofrecen las funcionalidades para la carga y uso del código de módulos, librerías y API.
El código de esta capa hace uso constante de las funcionalidades encontradas en PHP 5.3 para el manejo de los recursos.

Los archivos que forman el framework son esencialmente los siguientes:

/index.php
/framework/init.php
/framework/sys/acceso.php
/framework/sys/defines.php
/framework/sys/face.php
/framework/mvc/*

El framework inicia desde la página principal, index.php, con el siguiente contenido:

```
<?php
/**
* archivo: index.php
* Acceso al sistema
*/
define('DEVELOPMENT_ENVIRONMENT', false);
...
require 'framework/sys/defines.php';
require RUTA_FRAMEWORK . 'init.php';
```

El archivo de inicio carga primero las constantes básicas desde el archivo defines.php. Después de levantar el entorno, ejecuta el script 'init' encargado de ejecutar el framework, y consecuentemente el CMS. El archivo contiene pocas instrucciones y puede extenderse fácilmente. *init.php* es un controlador rudimentario del framework donde se procesa la solicitud recibida a través de index.php y se ejecuta el modo del módulo.

Init.php es el encargado de cargar el archivo face.php. Éste es otro archivo especial que dentro de su código define la función `face()` usada por la implementación para acceder a las librerías y API del framework. La función es un alias de un singleton de la clase /lib/classLoader.class.php con la ventaja de que al ser una función, es utilizable desde cualquier parte del código por su alcance como función y es una instrucción más sencilla de escribir al estar desarrollando. En el archivo face.php se establecen también las reglas de la carga de archivos de manera dinámica con la función __autoload. Esta función carga cada una de las clases del MVC y otras necesarias automáticamente cuando PHP encuentra una referencia a una clase que no ha cargado. Esta práctica es útil ya que al cargar a demanda las clases no hay uso innecesario de memoria [34], [35].

La última parte, el MVC, es ejecutada al finalizar el proceso de cargado del framework. Al terminar la secuencia de preparación, la función `callHook()` de init.php al tener identificados el módulo, el modo, la acción y los parámetros; hace el llamado al controlador del módulo que corresponde para que se realicen las acciones pertinentes.

4.4 Módulos

La última capa del sistema, que es la del CMS, está formada por módulos (o componentes como el caso del sistema de usuarios, donde la diferencia es conceptual; el componente en este caso es obligatorio mientras que el módulo no).

Como se ha mencionado, las soluciones que ofrece un módulo las puede dar en diferentes ámbitos o modos. Por ejemplo, el despliegue de contenidos de manera dinámica es

una solución en el modo de aplicación; esto porque es la aplicación que se está consultando (se llama aplicación porque es el modo más simple, aún estando en el entendido de que cualquier uso del sistema es un acceso a la aplicación realmente).

La administración de una cuenta de un profesor, donde él pueda gestionar sus datos es una solución en el modo de usuario, puesto que es a través de una cuenta de usuario que se está trabajando; en el modo de aplicación será donde se desplieguen esos datos.

En el modo de administración, la cuenta administradora configura globalmente al CMS, a los usuarios y parcialmente a las vistas que se despliegan en el modo de aplicación, entre otras cosas. Se tienen definidos tres modos, sin embargo pueden crearse más de ser necesario, haciendo las adaptaciones al núcleo del framework para que pueda identificarlos como lo hace con los existentes.

Modelo general de los módulos

Estructura de directorios y archivos:

```
modulos/
    mod_foo/
        ad/ | us/ | ap/
            c.class.php
            m.class.php
            v.php
            css.css
            es.lng.php
            js.js
        mod/
            mod.info
            api.class.php
            mod.api.class.php
            mod_foo.json.php
            inst.php
            pag.modv.php
```

Para que un módulo pueda ser identificado tiene que cumplir con algunos requisitos.

1. El directorio del módulo debe estar dentro del directorio de módulos
2. El nombre de la carpeta del módulo debe contener el prefijo mod_
3. El modo de aplicación debe estar en la carpeta /ap
4. El modo de administración debe estar en la carpeta /ad
5. El modo de usuario debe estar en la carpeta /us

6. Cada modo debe contener los archivos del MVC: c.class.php, m.class.php y v.php
7. Los archivos c.class.php y m.class.php deben de extender las clases Control y Model respectivamente y usar su espacio de nombres correspondiente
8. Los archivos de uso común y deben de estar dentro de la carpeta /mod

Si el módulo no tiene modo de administración no será administrable aunque puede funcionar. Por otro lado un módulo sin modo de aplicación ni de usuario pero con modo de administración puede permitir que el administrador gestione los datos que regresa su API, y esta API pueda ser consultada por otros módulos; este es el caso del módulo de administración de sistema, que no cuenta con un modo de aplicación y que se usa para definir la configuración del CMS.

Estructura de archivos

c.class.php La clase del controlador que se extiende del controlador genérico del MVC del framework. El control ejecuta las llamadas acciones, que son las ejecuciones de las funcionalidades del módulo (que implican la ejecución de uno o varios métodos del modelo y normalmente el despliegue de una vista). Por ejemplo, el módulo de micrositios al ejecutar la acción de sitios ejecuta una función que regresa el listado de sitios disponibles y genera una página con la información obtenida. La estructura del archivo se puede consultar en el apéndice 2 *Estructura de archivos, Archivo de controlador*.

Al implementar el controlador se documentan con PHPDoc de manera simple algunas funciones. Esta documentación hará que las funciones sean visibles y accesibles directamente desde la vista del sitio generado. El texto dentro del comentario se utiliza como etiqueta del enlace mostrado para acceder a la acción, un ejemplo se muestra en la Fig. 4.11.

```
	/** Generales */
	public function general() { … }
	/** Archivos/Directorios */
	public function lista() { … }
	/** Cuenta */
	public function adm_cta() { … }
	...
```

Fig 4.11. Vista de acciones disponibles en una plantilla

Una acción es una función invocada por el usuario a través del sistema. Las acciones son funciones del controlador que permiten ejecutar una serie de funciones del modelo (y en algunos casos de API) y despliegan un resultado sobre la vista que corresponde; esta correspondencia se da a través del nombre de la acción, que es el mismo para el controlador que para la vista. El término vista está definido bajo el contexto del framework, sin embargo, para fines prácticos al usuario final, una acción será el despliegue de una página (tanto informativa para un visitante, como de administración).

m.class.php La clase del modelo. Esta clase extiende del modelo genérico del MVC del framework. Es la clase que contiene la lógica de negocio de los módulos. Tiene las funciones que extraen, procesan e insertan los datos que se manejan en el módulo y los regresa de manera estructurada para que el controlador los haga disponibles en la vista que corresponde. Contiene la mayoría de las funciones que se utilizan para gestionar los datos del módulo. Además, el modelo es el que hace normalmente los llamados a las librerías y a las API.

La clase del modelo cuenta también con un constructor que permite inicializar variables que se quieran hacer disponibles para cualquier método, por ejemplo, podría usarse para crear un objeto con el acceso a la base de datos.
La estructura del archivo se puede consultar en el apéndice 2, *Estructura de archivos, Archivo de modelo*.

v.php El archivo de vista. Éste despliega la presentación de los contenidos. El código de la vista no está encapsulado en una clase, puesto que es la clase de la vista MVC del framework la que se instancia directamente, y es a través de ésta que se carga la vista correspondiente del módulo. Como todo el proceso de cargado está automatizado, para lograr este efecto hacemos que el nombre de la vista que se carga sea la misma que el de la acción que se ha llamado desde el controlador.
La estructura del archivo se puede consultar en el apéndice 2, *Estructura de archivos, Archivo de vista*.

Las vistas al servir para presentar los datos deben idealmente contener solamente código PHP para ciclos de impresión o elementos similares, aunque se tenga la posibilidad de programar elementos complejos.

Archivos opcionales Recordando el árbol de archivos y directorios podemos ver además de los archivos MVC del módulo algunos otros que son opcionales. Estos archivos se encuentran dentro del directorio mod.

```
/mod
        mod.info
        api.class.php
```

mod.api.class.php
mod_foo.json.php

mod.info Este archivo provee los detalles del módulo, que son datos no obligatorios usados al desplegar en modo de usuario o administración los módulos. La estructura del archivo se puede consultar en el apéndice 2, *Información de módulo*.

api.class.php / mod.api.class.php Los archivos [mod.]api.class.php son clases que extienden de la clase Singleton y que contienen a todas las funciones (a modo de métodos) que ofrece algún módulo como interfaz para ser llamadas desde cualquier otro módulo, es en sí una API encapsulada en una clase.

En el listado de directorios vemos otros archivos como css.css o es.lng.php. Se tiene la libertad total de definir cualquier archivo en caso de ser necesario, con base en las necesidades del módulo.

Para la descripción de los módulos se utilizará el término sección para referirnos a la vista resultante al ejecutar una acción. El acceso a las secciones se hace por medio de enlaces a modo de hipervínculos como pestañas que se despliegan en las plantillas.

4.4.1 Módulo de administración

El módulo de administración gestiona la configuración global del sitio generado. Permite definir elementos básicos como titulo y descripción y organización del menú del sitio además de implementar otras herramientas como la administración de la galería de imágenes del repositorio a nivel global y el editor de la plantilla de presentación del sitio; para una referencia se puede observar las Figs. 4.13 y 4.14, para mayor información se puede consultar el apéndice 1.2.1 *Módulo de administración.*

El módulo de administración utiliza dos tablas: una que almacena la configuración del menú de módulos desplegado en la plantilla en el modo de aplicación y otra con las configuraciones establecidas desde el área de administración general, que incluye a la plantilla de la portada.

Diagrama de base de datos

El diagrama de la base de datos se expone en la Fig. 4.12.

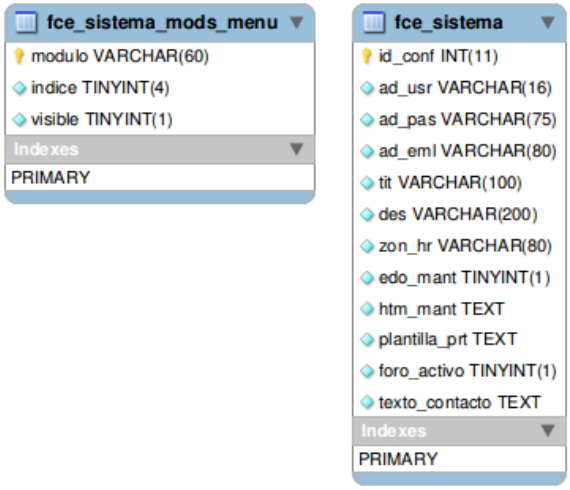

Fig. 4.12. Diagrama de base de datos del módulo de administración

Fig. 4.13. Vista de la sección general y del administrador de imágenes

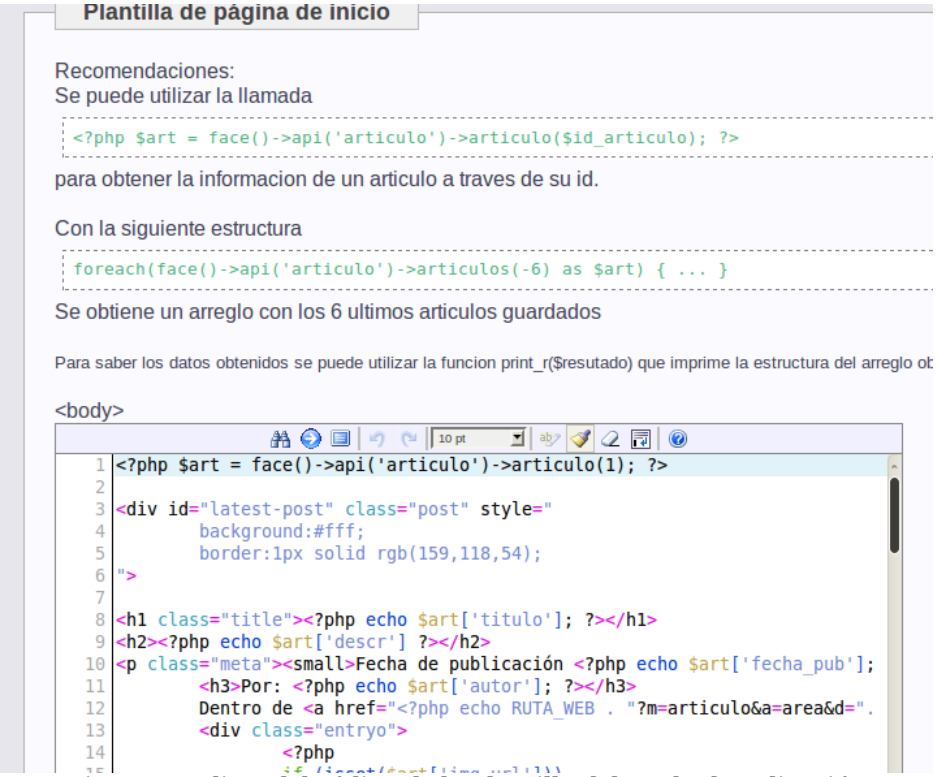

Fig. 4.14. Editor del código de la plantilla del modo de aplicación

Modos y funcionalidades

Considerando que este módulo es para configuraciones generales, el único modo que tiene definido es el de administración, siendo que el usuario que realmente debe tener acceso es el administrador del sitio.

Este módulo no tiene API.

4.4.2 Módulo de usuarios

El módulo de usuarios puede considerarse componente porque a diferencia de los demás módulos, éste contiene la lógica de gestión de usuarios, algo que lo hace de existencia prácticamente obligatoria por la naturaleza del sistema.

El módulo de usuarios implementa un sistema de permisos por grupos o roles (RBAC). Con los grupos se permite al usuario entrar a zonas dependiendo de los accesos permiti-

dos en cada uno de los grupos a los que pertenece, de esta manera se pueden crear distintos tipos de roles como profesores, editores, colaboradores, etc.

Diagrama de base de datos

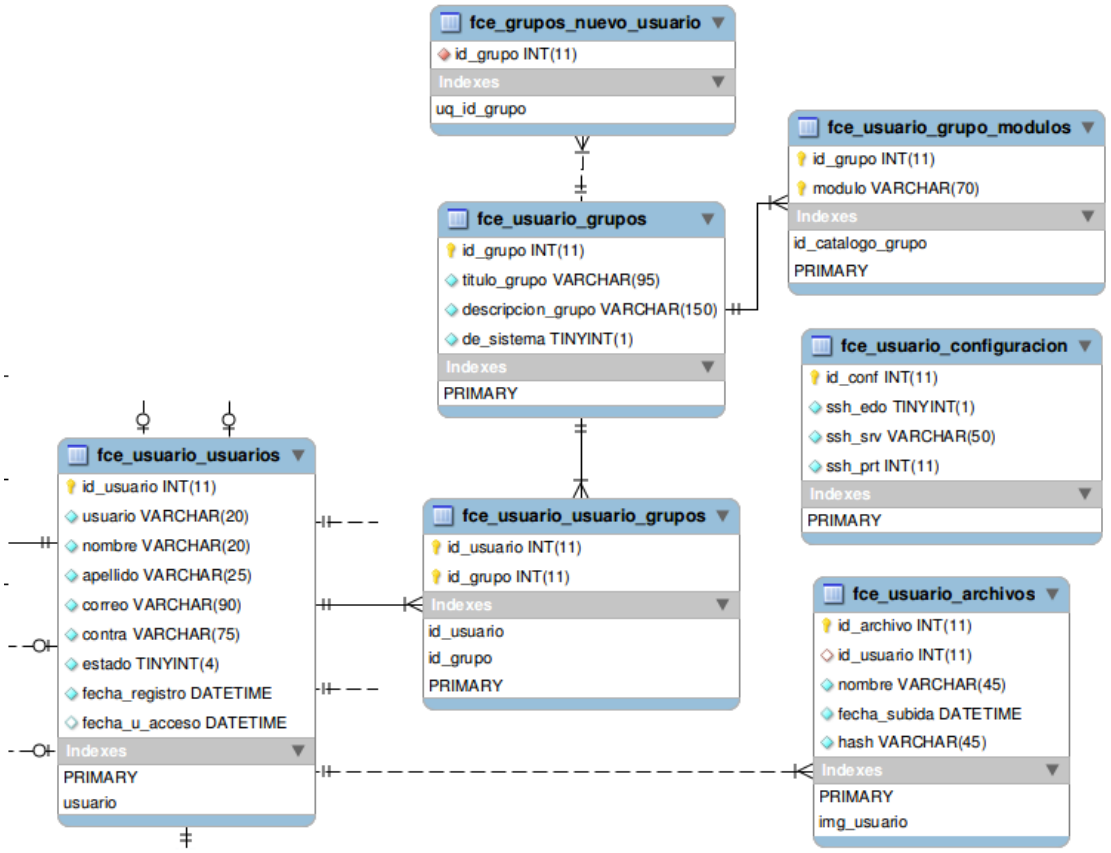

Fig. 4.15. Diagrama de base de datos del módulo de usuarios

Modos y funcionalidades

Este módulo tiene los tres modos definidos. El modo de administración gestiona a todos los usuarios y el de usuario los datos propios; para una referencia se puede observar las Figs. 4.16, 4.17 y 4.18, para mayor información se puede consultar el apéndice 1.2.1 *Módulo de usuarios*.

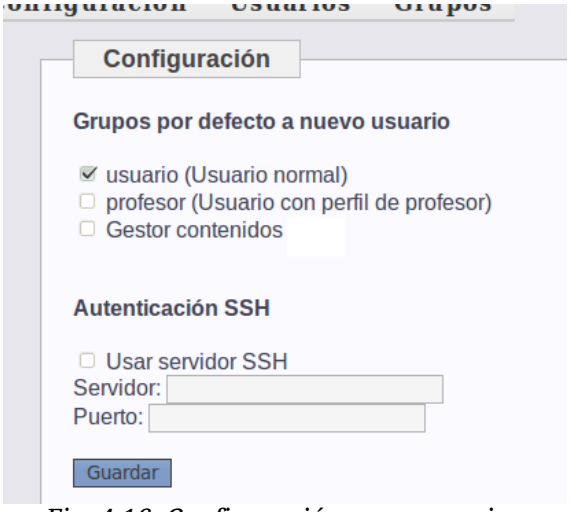

Fig. 4.16. Configuración para usuarios

Fig. 4.17. Vista de la gestión de usuarios

Fig. 4.18. Vista de la gestión de grupos

Tanto el modo de administración como el de usuario tienen un editor de perfiles; desde la administración se puede editar la información de cualquier usuario, desde el usuario se puede gestionar sólo la información propia, este modo tiene también un administra-

dor de imágenes donde puede subir imágenes para usarlas en publicaciones; para una referencia se puede observar la Fig. 4.19, para mayor información se puede consultar el apéndice 1.2.2 *Módulo de usuarios*.

El modo de aplicación es muy simple y sólo contiene un formulario para poderse autenticar como usuario. Acerca de la API, ésta puede consultarse en el apéndice 4 *API de módulos - Usuarios*.

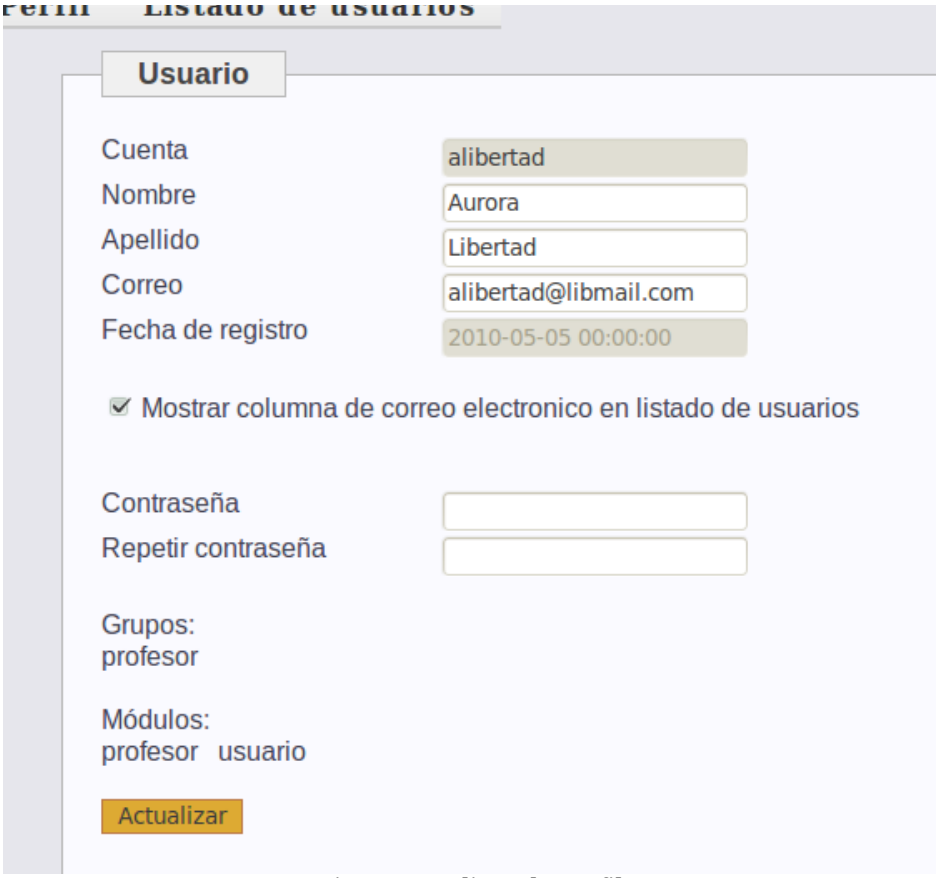

Fig. 4.19. Editor de perfil

4.4.3 Módulo de artículos

El módulo de artículos permite publicar de manera organizada notas de distintos tipos como noticias, eventos o información relevante sobre la Facultad o Universidad.
Los artículos, a diferencia de las páginas comunes (gestionadas por otro módulo), tienen información de un tema concreto de manera completa mientras que las páginas tienen toda la información de un tema de manera distribuida.

Diagrama de base de datos

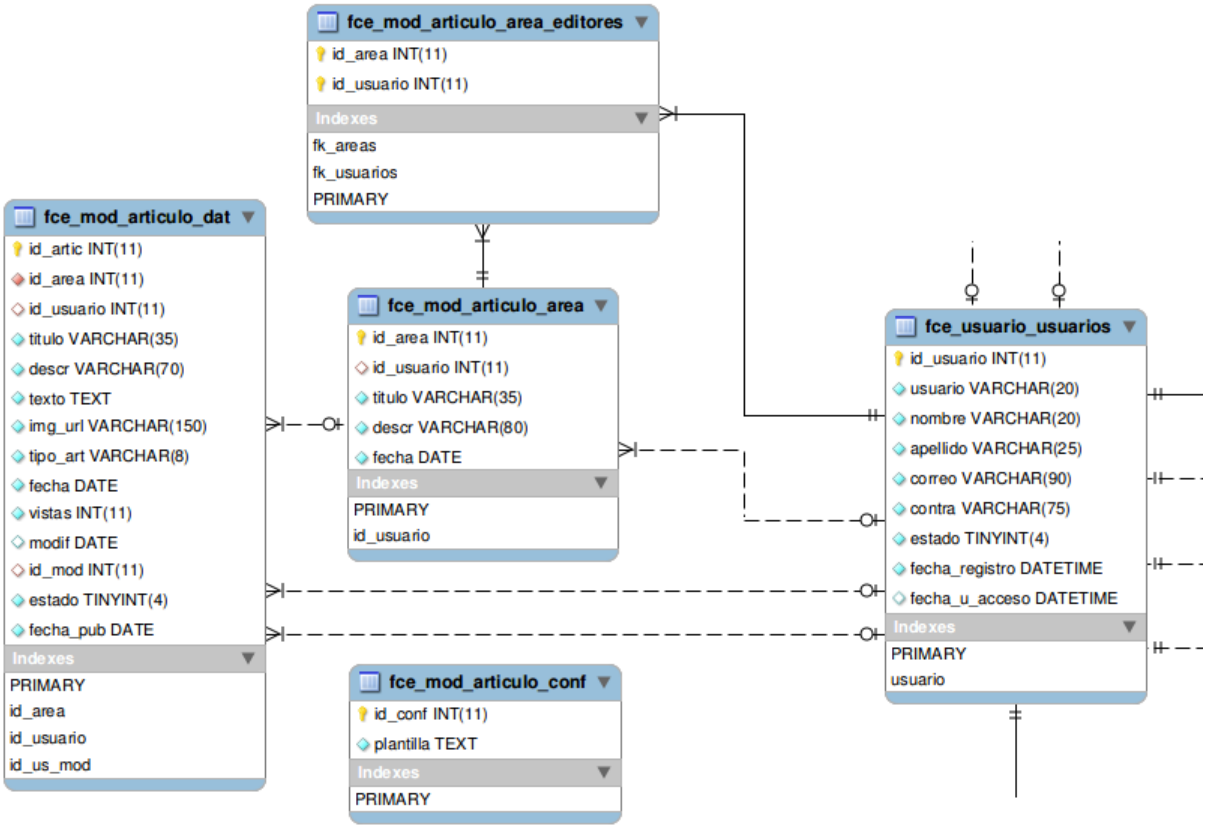

Fig. 4.20. Diagrama de base de datos del módulo de artículos

Modos y funcionalidades

Este módulo tiene los tres modos, aplicación, usuario y administrador.
Como administración tiene secciones para crear y gestionar áreas y artículos; para una referencia se puede observar las Figs. 4.21, y 4.22, para mayor información se puede consultar el apéndice 1.2.3 *Módulo de artículos*.

El modo de usuario permite trabajar de la misma manera a las áreas y artículos que desde el administrador, sin embargo, el usuario tiene limitantes como no poder eliminar o crear nuevas áreas, además de tener acceso sólo a los artículos de las áreas permitidas.

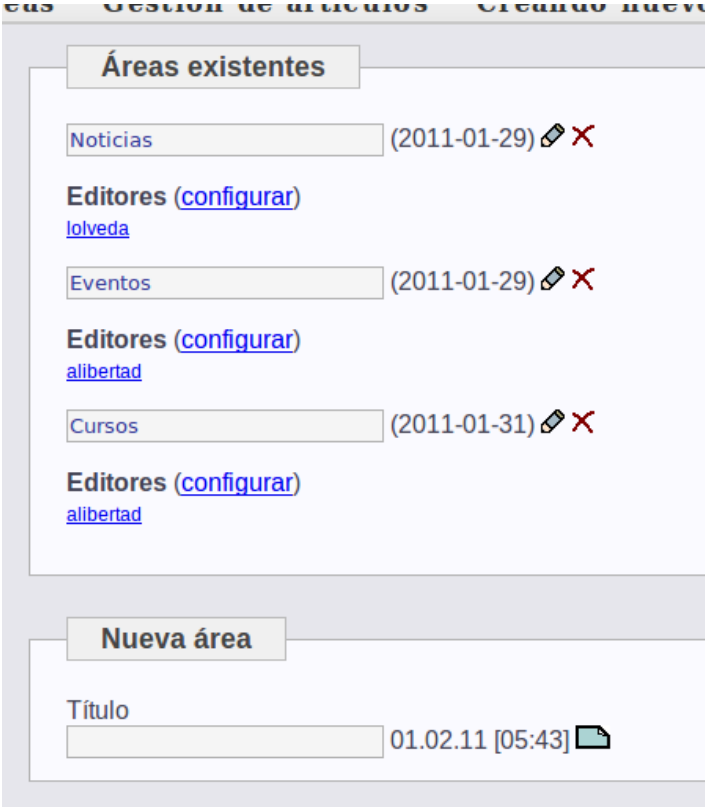

Fig. 4.21. Administración de secciones

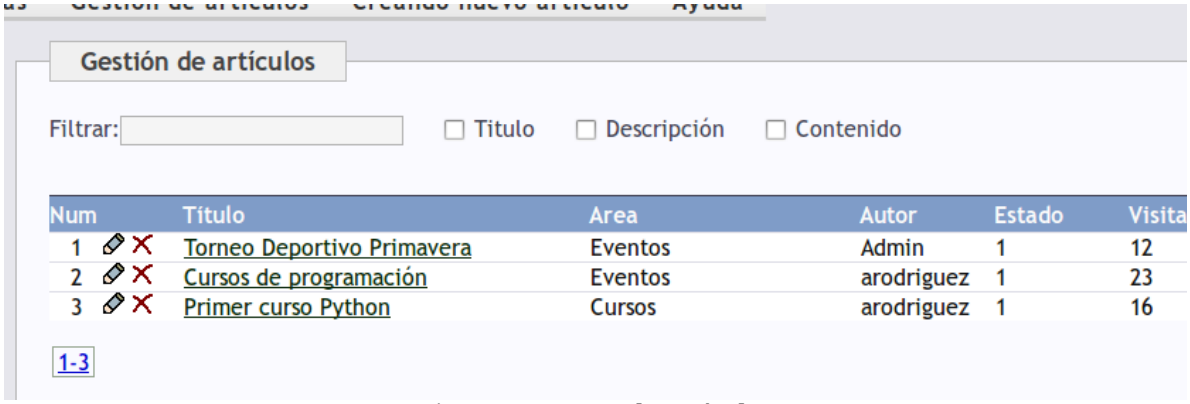

Fig. 4.22. Gestor de artículos

La vista en el modo de aplicación está dividida en dos secciones: Áreas y artículos y Artículo, el primero es un listado de artículos organizados con enlaces para dar acceso al artículo en sí, que es la segunda sección; para una referencia se puede observar las Figs. 4.23 y 4.24, para mayor información se puede consultar el apéndice 1.2.3 *Módulo de artículos.*

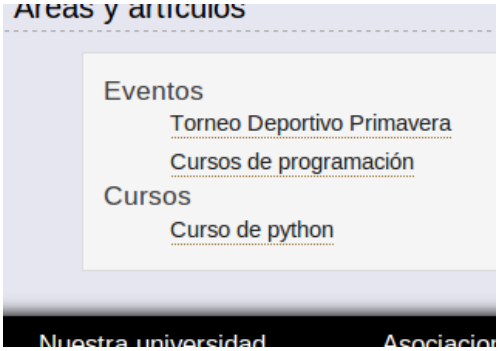

Fig. 4.23. Listado de artículos

Fig. 4.24. Despliegue de un artículo

API

Para consultar la interfaz, ir al apéndice 4 *API de módulos, Artículos.*

4.4.4 Módulo de profesor

Este módulo permite crear y administrar perfiles de profesores. La gestión se puede hacer directamente desde la cuenta de administrador o por los mismos usuarios, que pueden actualizar y modificar sus datos. Este método despliega una interfaz común para la gestión de los datos y automatiza la generación del directorio de personal docente. También despliega la información de todos los registros en el modo de aplicación con la misma estructura.

La implementación está construida usando el componente de usuarios para el uso de las cuentas.

Diagrama de base de datos

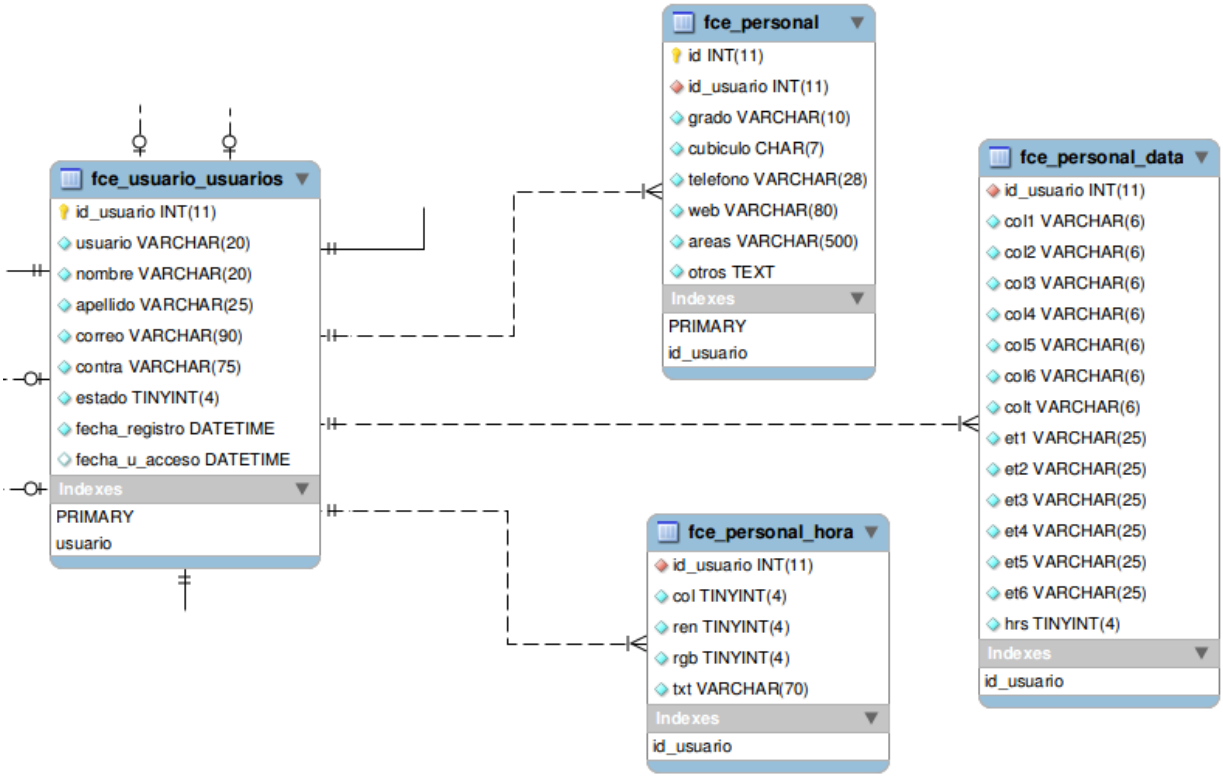

Fig. 4.25. Diagrama de base de datos del módulo de profesores

Modos y funcionalidades

Este módulo es muy similar al módulo de usuarios, incluso podría verse como una extensión de él sobre usuarios que tienen asignado el rol de profesor (o que el usuario pertenece a un grupo con acceso al módulo de profesores). Al igual que el módulo de usuarios, desde el modo de administración se pueden crear y gestionar profesores y su información, mientras que desde el modo de usuario, se puede gestionar solamente la información propia. Se menciona que puede verse como una extensión porque la información gestionada es la información de perfil pero sobre un contexto académico, esto es, se gestiona información de índole profesional desde el título académico hasta el horario laboral. El modo de aplicación despliega una lista de todos los profesores existentes con una liga para consultar su información; para una referencia se puede observar las Figs. 4.26, 4.27, 4.28 y 4.29, para mayor información se puede consultar el apéndice 1.2.3 *Módulo de artículos*.

Fig. 4.26. Listado de profesores sobre modo de administración y aplicación

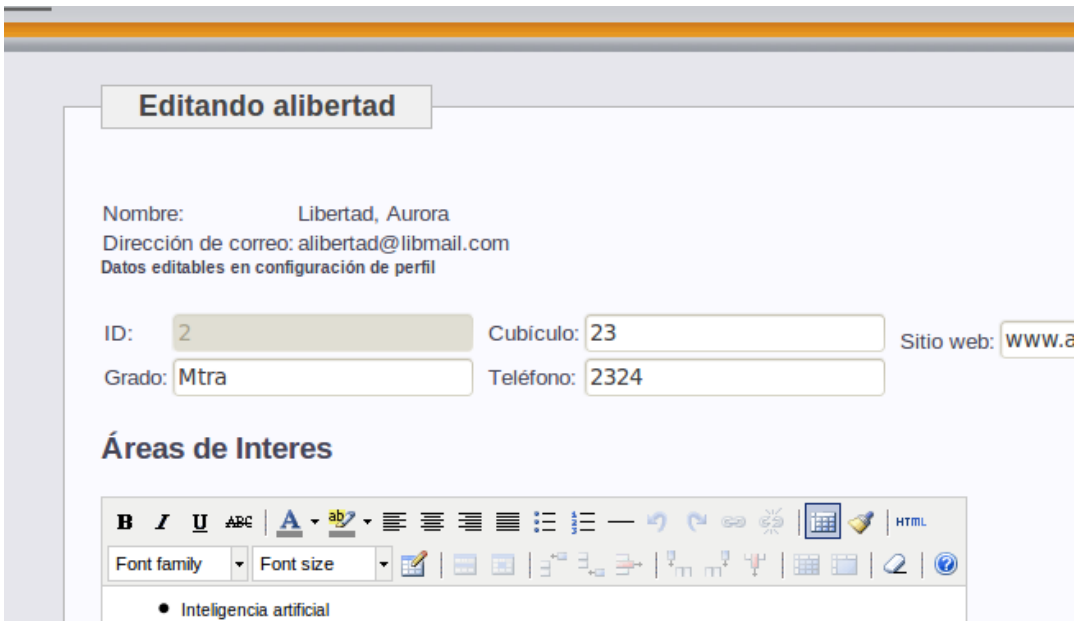

Fig. 4.27. Vista de la sección de edición, datos de perfil

Fig. 4.28. Vista de la sección de edición, editor de horarios

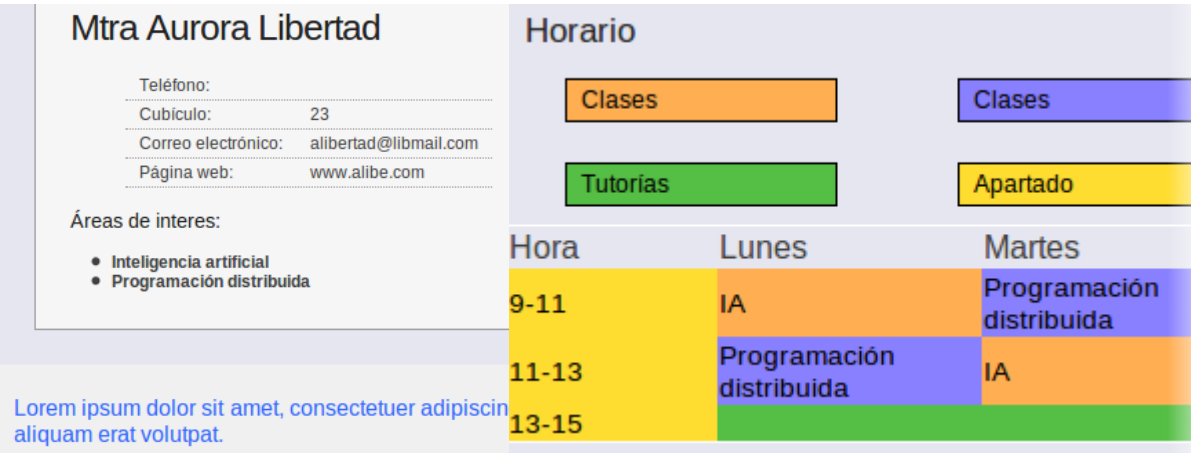

Fig. 4.29. Perfil desplegado en vista de aplicación, porción de datos generales y horario

API

Para consultar la interfaz, ir al apéndice 2 *API de módulos, Profesores*.

4.4.5 Módulo de micrositios

Este módulo de uso general se utiliza para crear los diferentes contenidos del sitio. Un micrositio es un pequeño conglomerado de páginas relacionadas. Cada micrositio además de estar compuesto de páginas puede contener otros microstios, este esquema se puede repetir indefinidamente, con lo que se puede crear la estructura del sitio actual desplegada en la Fig. 4.31.

Desde este módulo es posible modelar la mayoría de las secciones con las que cuenta actualmente el sitio de la Facultad de Ciencias de la Computación. Se puede crear el micrositio de de Información de la Facultad, Secretaría Académica, de las Áreas de Investigación, Bolsa de Trabajo, Vinculación, Becas, Diplomado, Servicio Social, Educación Continua, etc; y para casos en los que se prefiere que la página o sitio sea externo, se pueden crear páginas de redirección que generan una entrada en el menú del sitio como cualquier otra página, pero enlazan al sitio externo al dar clic. Otra funcionalidad es la de bloquear temporalmente páginas para evitar su despliegue sin tener que eliminarlas.

Diagrama de base de datos

El diagrama de la base de datos se expone en la Fig. 4.30.

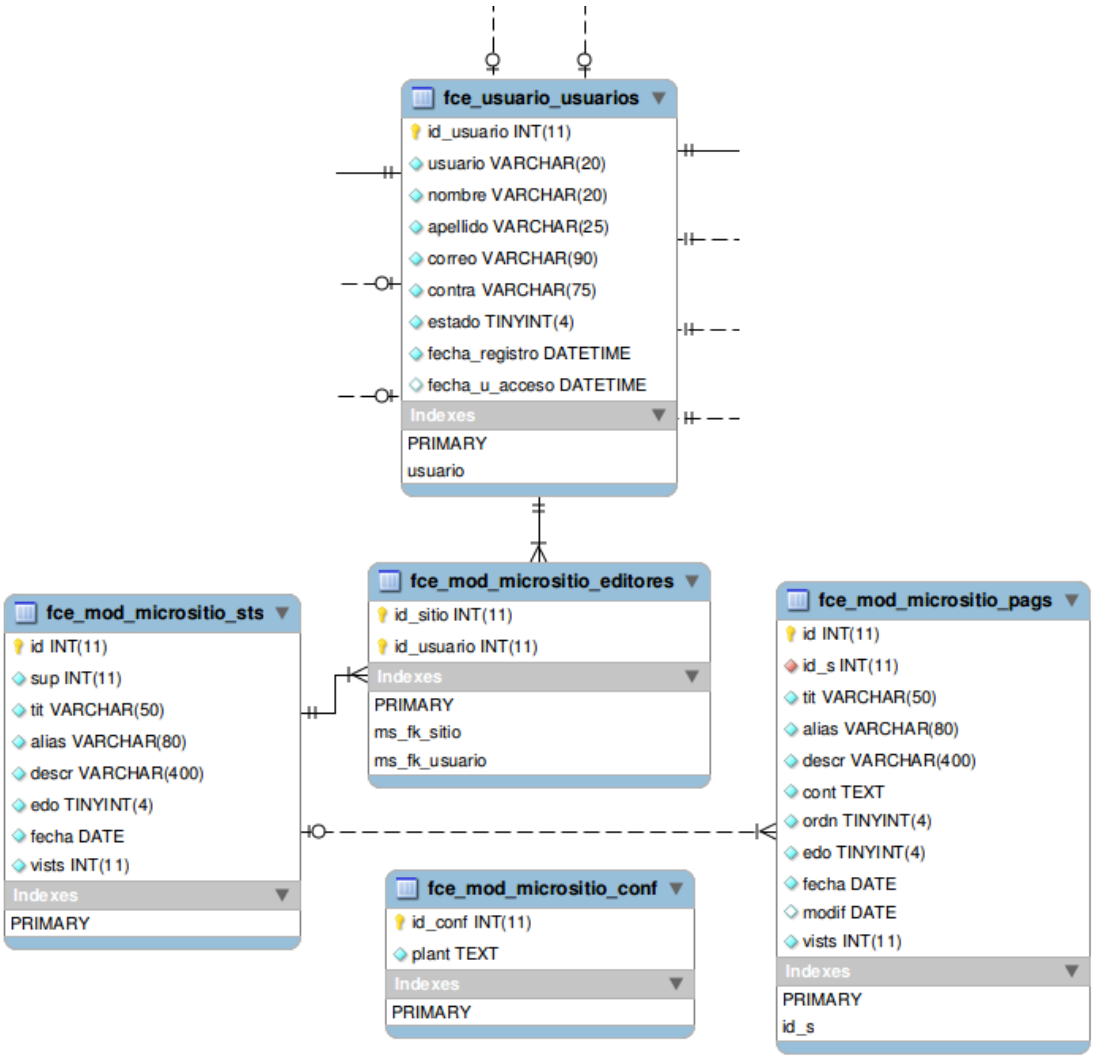

Fig. 4.30. Diagrama de base de datos del módulo de micrositios

Modos y funcionalidades

Este módulo contiene la funcionalidad básica para cualquier sitio, algo que evidentemente implica que debe tener programados los tres modos de funcionamiento. Como en otros casos los modos de administración y usuario son similares. Desde la administración se pueden gestionar las páginas y sitios, además de los usuarios que tendrán permisos para gestionar cuáles sitios; la plantilla de despliegue y el mensaje de error para páginas no encontradas.

En modo de usuario, cuando éste tiene permisos, puede gestionar páginas dentro del micrositio al que se le ha permitido y a los elementos contenidos en cascada, de modo

que, diferentes usuarios puedan ser responsables de la edición de diferentes secciones; tomando como base la Fig. 4.31, un usuario A puede encargarse de crear todos los contenidos de 'Oferta Educativa' y consecuentemente 'Áreas de Investigación' pero no tener acceso a los datos en modo edición de 'Secretaría Académica', además, un usuario B puede tener permiso de edición sobre 'Áreas de Investigación' pero no sobre 'Oferta Educativa'.
El modo de aplicación es finalmente la parte que permite navegar sobre todos los micrositios y sus páginas desde un menú desplegado que sigue la estructura con que se han creado sus elementos.

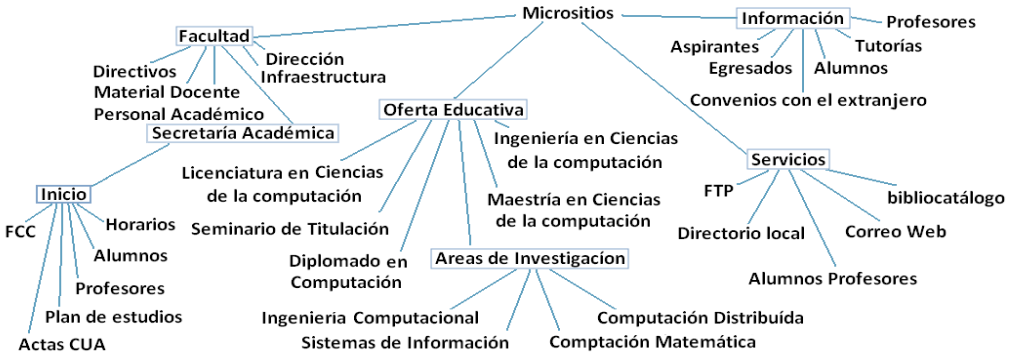

Fig. 4.31. Árbol con estructura actual del menú del sitio de la Facultad (sin ordenar)

Este modo tiene algunas particularidades. Primero, en el menú de módulos, éste no es desplegado (tiene un menú propio para desplegar las páginas); al acceder a algún contenido del módulo, como sólo existe una sección (resultado de la única acción) no se muestra el acceso a ninguna.

Fig. 4.32. Menú generado en la vista de aplicación y desplegado de una página

Por otro lado, la plantilla de la vista de aplicación a nivel general, contiene un llamado a la API de este módulo que regresa un menú con la estructura con la que se han creado los sitios e hipervínculos a sus páginas, este menú es el menú del sitio. Para una referencia se puede observar las Figs. 4.32, 4.33, 4.34 y 4.35, para mayor información se puede consultar el apéndice 1.2.3 *Módulo de micrositios.*

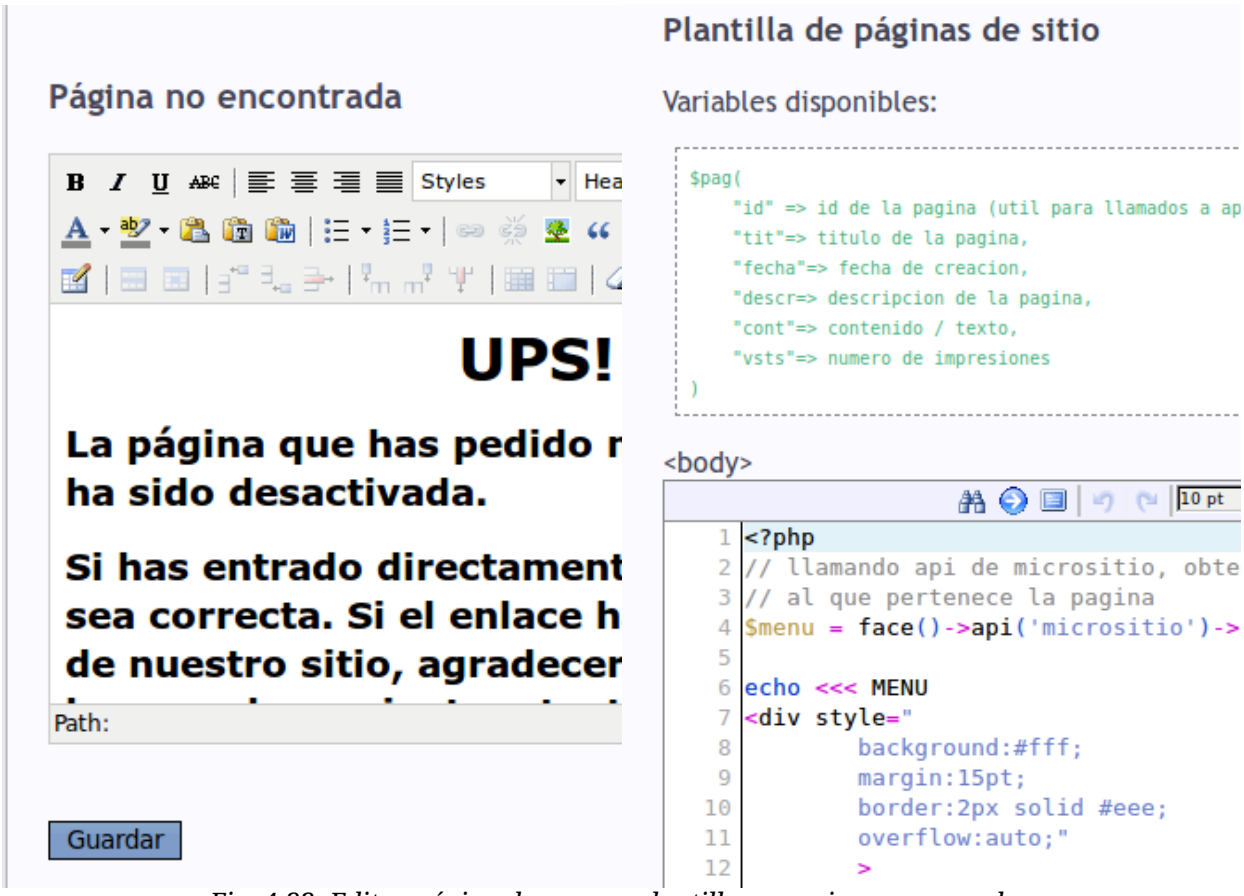

Fig. 4.33. Editor página de error y plantilla en opciones avanzadas

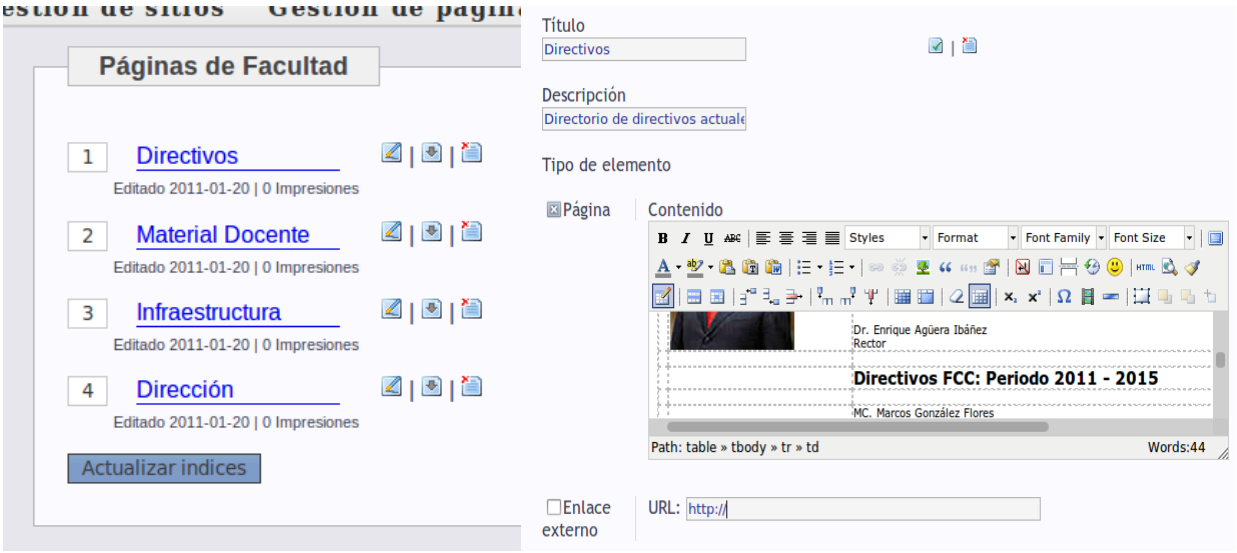

Fig. 4.34. Nueva página

Fig. 4.35. Gestión de sitios en modo de administración y de usuario

API

El API de este módulo está dividido en dos partes; la primera, de uso general (api.class.-php), contiene funciones para obtener información almacenada en el módulo. Particularmente esta API contiene una función de interés especial, `pon_menu()`, que genera un menú en un formato HTML listo para desplegarse preferentemente desde una plantilla de modo aplicación, en forma de lista no ordenada, de manera que semánticamente se utiliza una función natural de las etiquetas que funcionan aún sin estilos [36], [37].

Para consultar la interfaz, ir al apéndice 2 *API de módulos, Micrositios*.

4.4.6 Módulo de enlaces

Este módulo implementa el conjunto de funcionalidades para administrar los documentos que conforman un proyecto que es llevado colaborativamente. El módulo ha sido desarrollado con el fin de ofrecer herramientas que permitan llevar un control sobre los documentos generados al trabajar sobre un proyecto idealmente inter-facultativo entre diferentes instituciones y que a su ves permita publicar ciertos archivos que los integrantes del equipo consideren, de modo que la divulgación del trabajo sea sencilla y los conocimientos que han sido generados sean accesibles fácilmente.

Si por ejemplo, existe un proyecto entre la Facultad de Ciencias de la Computación y la Facultad de Ciencias de la Electrónica, ambas integrantes de la DESIT, se pueden crear cuentas para todos los usuarios que estarán involucrados para que tengan acceso y puedan crear documentos, subir diferentes versiones y hacer comentarios sobre estas y finalmente dar permisos de acceso público a los archivos que se deseen. Además, si existieran usuarios que pertenecen a otras instancias del CMS, estos pueden conectarse desde su propia instancia a servidor de manera remota y trabajar de la misma manera que en un servidor local, y en este caso, el módulo incluye la opción de que los archivos que se establezcan como públicos en la instancia remota sean listados también en su instancia propia.

Otra función del módulo genera un RSS que puede desplegar tanto información propia (generada o no de manera dinámica) como de otras instancias. De esta manera se pueden compartir entre diferentes sitios enlaces a archivos o contenidos. El RSS que se publica un sitio contiene tanto las entradas del sitio propio como las de los sitios a los que está subscrito.

La Fig. 4.36, contiene los diagramas de comunicación entre instancias para desplegar información. La parte de proyectos muestra como usuarios que colaboran en un proyecto 'P' desde diferentes instancias, despliegan desde sus sitios respectivos los mismos archivos públicos que el sitio que alberga al proyecto. La parte del RSS muestra como diferentes instancias que generan sus propios RSS (A, B, N) son enlazados para desplegarse de manera conjunta por el sitio que los ha agregado como RSS enlazados.

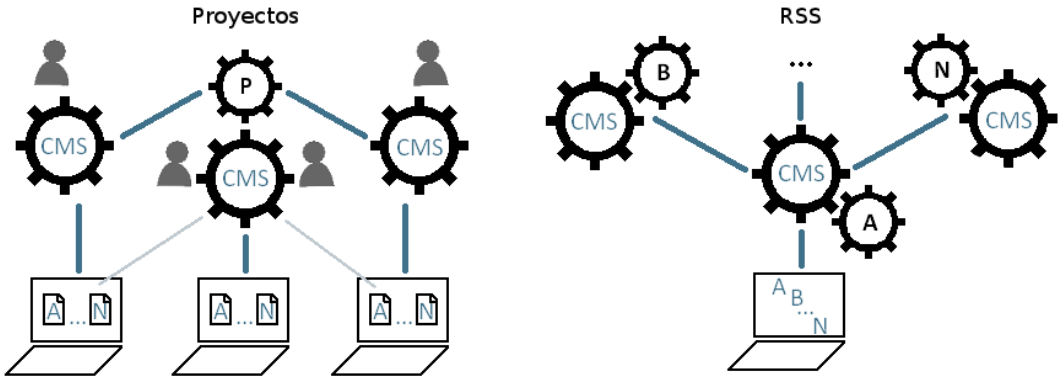

Fig. 4.36. Diagrama de conexión entre instancias compartiendo enlaces de un proyecto y RSS

La implementación utiliza la librería Snoopy (http://snoopy.sourceforge.net/) para hacer solicitudes HTTP entre servidores para sincronizar automáticamente la información entre servidores y el motor MagpieRSS (http://magpierss.sourceforge.net/) en una versión modificada no oficial mejorada (http://www.usefulsnippets.com/improved-magpierss.html) para leer RSS externos con algunos cambios para integrarla a al framework.

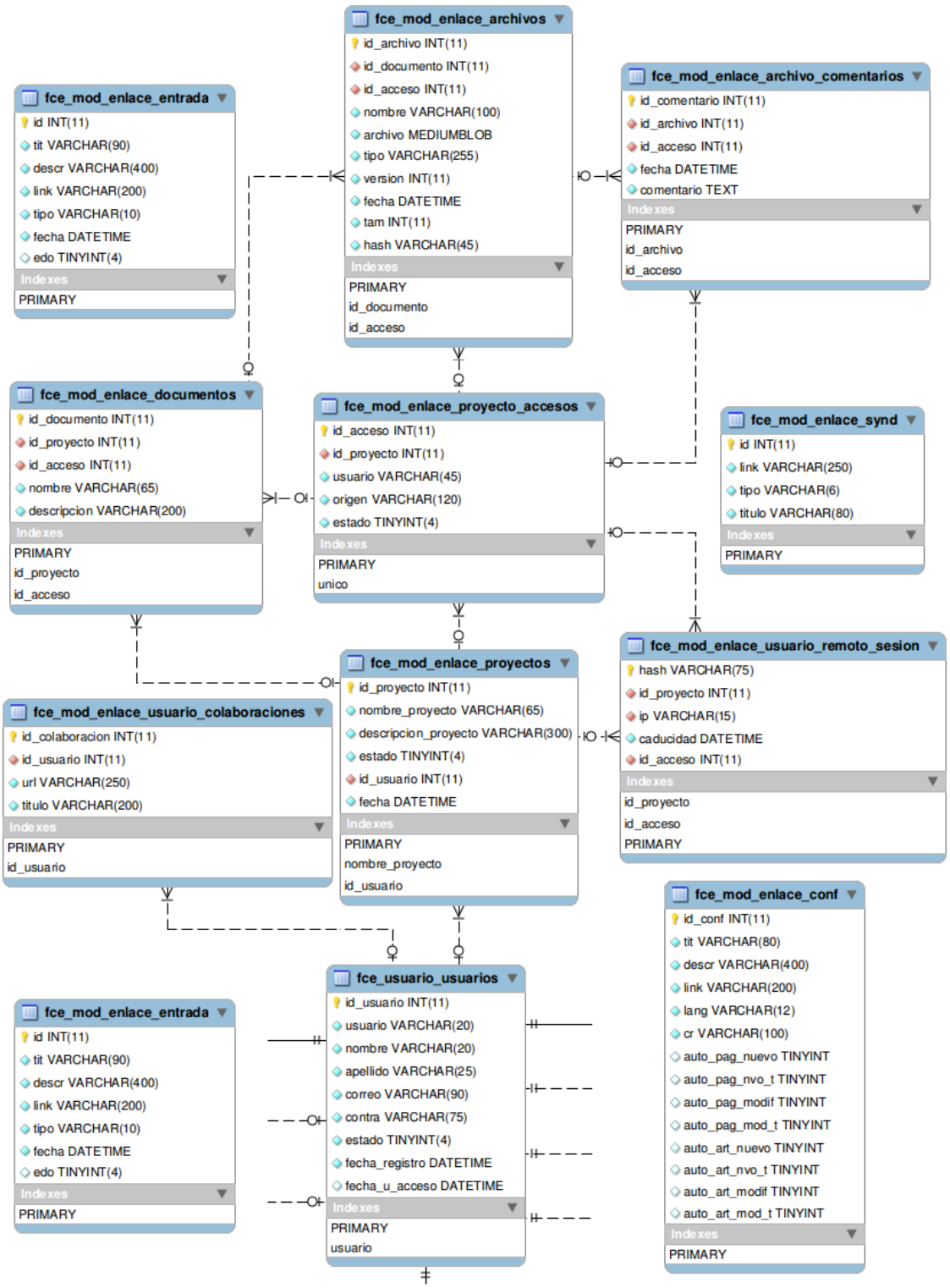

Fig. 4.37. Diagrama de base de datos del módulo de enlaces

Diagrama de base de datos

El diagrama de la base de datos se expone en la Fig. 4.37.

La estructura de las tablas almacena además de los elementos de proyectos, documentos y archivos y sus comentarios y un sistema básico de autenticación remota que almacena sesiones con caducidad después de verificar el acceso a través de otro servidor.

Modos y funcionalidades

Este módulo contiene los tres modos. El módulo de administración permite configurar y gestionar el RSS que se genera para el sitio, definiendo si generar nuevas entradas de manera automática al crear o modificar páginas o artículos; permite también agregar entradas manualmente; la Fig. 4.38 muestra imágenes de tales funcionalidades.

El administrador tiene también a posibilidad de agregar fuentes RSS externas para publicarlas como parte del RSS del sitio ayudando a difundir la información. Cuando un usuario tiene acceso a un proyecto que se encuentra en otra instancia se le permite desplegar los archivos del proyecto remoto, el administrador tiene la posibilidad de eliminar esos enlaces, o agregar nuevos manualmente en caso de desear publicar los archivos en el sitio sin colaborar sobre el proyecto, ésto puede observarse en la Fig. 4.39.

Fig. 4.38. Configuración de enlazador, sección para agregar elementos e indizador de enlaces

Fig. 4.39. Funciones para enlazar RSS y proyectos externos

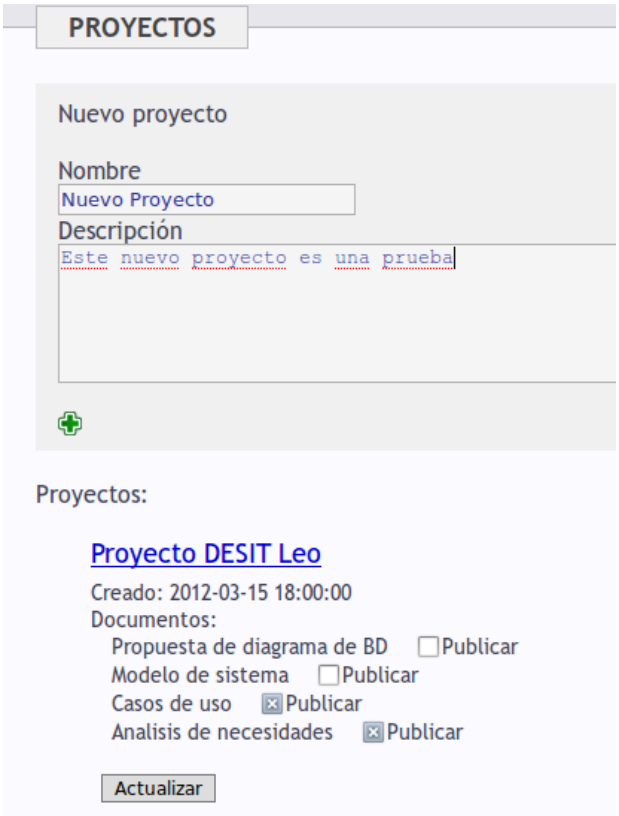

Fig. 4.40. Creación y gestión de proyectos

El modo de usuario contiene un conjunto de funcionalidades para permitir el trabajo colaborativo sobre documentos. Básicamente un usuario crea un proyecto (Fig 4.40) al que se le agregan documentos, que están conformados por archivos (Fig. 4.41). Los archivos pueden ser subidos por el administrador del proyecto o por cualquier colaborador autorizado (que puede pertenecer al mismo o a otro servidor usando una autenticación remota) (Fig. 4.42), además pueden hacerse comentarios sobre los archivos.

En el detalle de archivo se despliega toda su información, la liga de descarga y los comentarios escritos sobre el archivo. Las funciones de eliminación están solamente disponibles para el dueño del proyecto.

El modo de aplicación tiene una sección que despliega la información de los enlaces que se han definido en la creación de enlaces y la información obtenida de las ligas que se han indizado, desplegando los datos del RSS como una página del sitio, también cuenta con otra sección de proyectos donde se listan los proyectos públicos o remotos enlazados que cuentan con al menos un documento que se ha establecido como público; esto se muestra en la Fig. 4.43.

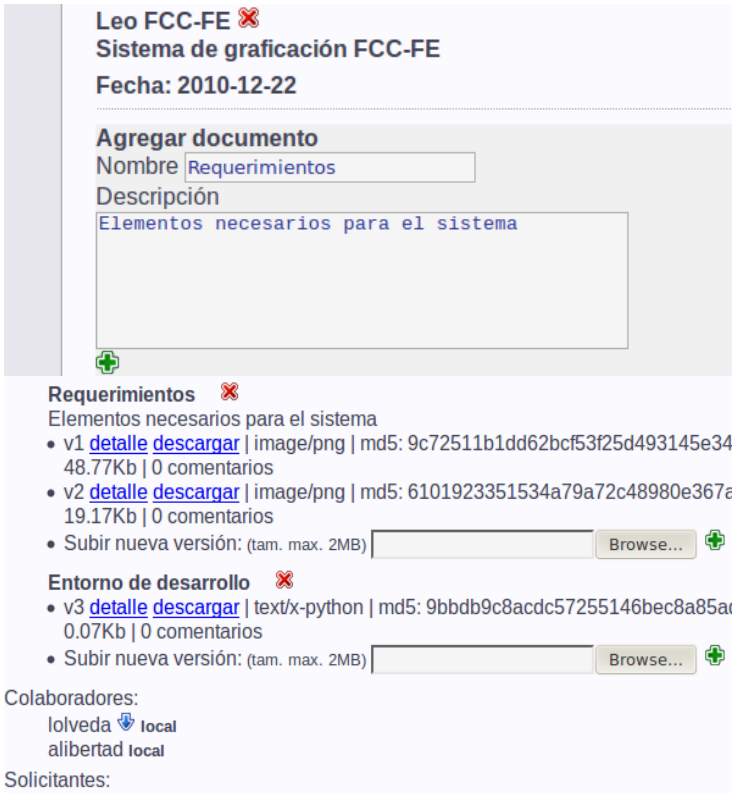

Fig. 4.41. Administración de documentos

Fig. 4.42. Detalle de archivo y sección de colaboraciones

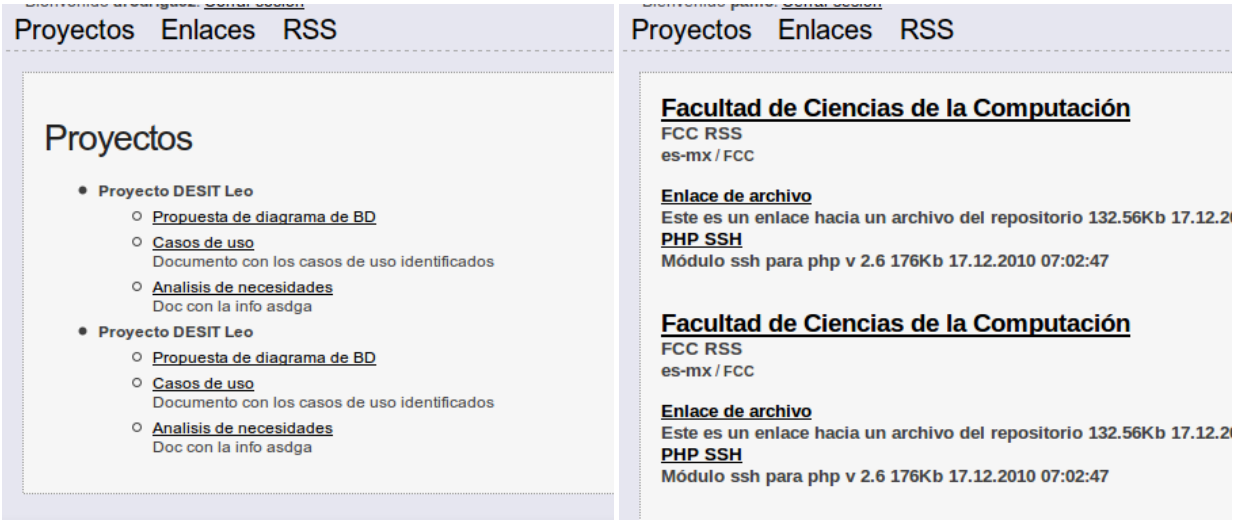

Fig. 4.43. Vista de aplicación, despliegue de documentos públicos de un proyecto y enlaces del RSS en modo web

API

Para consultar la interfaz, ir al apéndice 2 *API de módulos, Enlaces.*

4.4.7 Módulo de portada

Este módulo únicamente gestiona la portada del CMS.

Diagrama de base de datos

Este módulo no gestiona tablas directamente, sino que, accede a una tabla del módulo de administración de donde obtiene la plantilla configurada para usarse en la portada.

Modos y funcionalidades

Ese módulo no tiene modo de administración, sin embargo la plantilla es configurada en el modo de administración general. Esta función está en la sección de avanzadas. El módulo tampoco tiene modo de usuario.
El modo de aplicación despliega el contenido de la página principal. Esta página puede desplegar información estática y dinámica obtenida utilizando la API de cualquier módulo o inclusive código escrito en PHP.
Actualmente la configuración de la plantilla tiene comentada una sección que hace llamados a la librería del foro, que en caso de estar instalado puede descomentarse para agregar información de los mensajes escritos.

La vista de la portada del CMS (Fig. 4.46) está configurada para que inicialmente despliegue los seis últimos artículos publicados, desplegando el más actual en la parte principal. Se pueden sin embargo, cambiar la manera en que se despliegan los artículos, elegir directamente qué artículos desplegar de manera continua, o cualquier configuración que sea de interés.

API

Este módulo no tiene API.

4.5 Plantillas

En un CMS las plantillas se encargan de desplegar los contenidos de una manera uniforme, estructurada y consistente. Se puede imaginar lo tedioso que podría ser tomar una porción de registros de la base de datos para agregarlos al CMS y transformarlos en una página. Usando plantillas se decide una vez cómo se desea estructurar y dar formato al contenido, y se delega el trabajo de construir la publicación de páginas o secciones al sistema [38].

Para el desplegado se manejan distintas plantillas, que dependen del modo; una para la vista de la aplicación, otra para la vista de administración y una más para la vista de usuario, que son independientes entre ellas. Su uso facilita la aplicación de nuevos estilos y elementos en la vistas y permiten cambiar fácilmente la estructura visual, simplificando

la aplicación de nuevos diseños. La manera habitual para esto es definir parcialmente la vista de la plantilla, con un contenido de estilos mínimo, limitado a tamaños de texto y tipografías, contrastes entre fondos, para agregar en iteraciones sucesivas detalles cada vez más particulares.

Con las plantillas se puede estructurar la vista de manera sencilla con dos elementos: el código HTML que definirá la estructura de la información, como el orden de los menús contenidos, hipervínculos estáticos, etc.; y el código CSS que da el estilo a la plantilla y por consiguiente a las páginas resultantes.

Al crear una página web, se utiliza en primer lugar el lenguaje (X)HTML 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 tipografía, separación horizontal y vertical entre elementos, posición de cada elemento dentro de la página, etc.; con lo que se obtienen confortablemente documentos bien formados (semánticos), más accesibles y que son son fáciles de mantener [39].

La mayor parte del contenido de las páginas HTML habituales está formado por texto, llegando a ser más del 90% del código de la página. Por este motivo, es muy importante conocer los elementos y etiquetas que define HTML para el manejo del texto [40].

Las plantillas son manejadas por el elemento Vista del MVC. Después de que se han procesado los datos y es momento de desplegarlos, se obtiene desde la vista la plantilla correspondiente y se insertan dentro de ella el resultado de la solicitud. En el código de la plantilla se definen etiquetas especiales sobre las que se insertan datos.
Las plantillas además de cargar archivos habituales como imágenes, CSS o JS, también pueden incluir llamadas a cualquier librería del framework o API de algún módulo.

Plantilla del modo de administración. Esta plantilla (Fig. 4.44) usa una librería para obtener el listado de módulos con modo de administración junto con su información extra (provista por el archivo mod.info si es que existe). Con la información obtenida se genera un menú agregando un acceso por cada módulo administrable.

Plantilla del modo de usuario. La plantilla del modo de usuario (Fig. 4.45) es muy similar a la plantilla de administración siendo que el método para desplegar los módulos es el mismo con un mecanismo extra que filtra sólo los accesos permitidos por los grupos a los que pertenece el usuario.

Plantilla del modo de aplicación. La plantilla en modo de aplicación (Fig. 4.46) está basada en el template Puzzled de freecsstemplates.org (http://www.freecsstemplates.org/previews/puzzled/) que es de uso libre. Esta plantilla distribuye los conteni-

dos de una manera amigable introduciendo todos los elementos que el sistema genera mediante sus módulos.

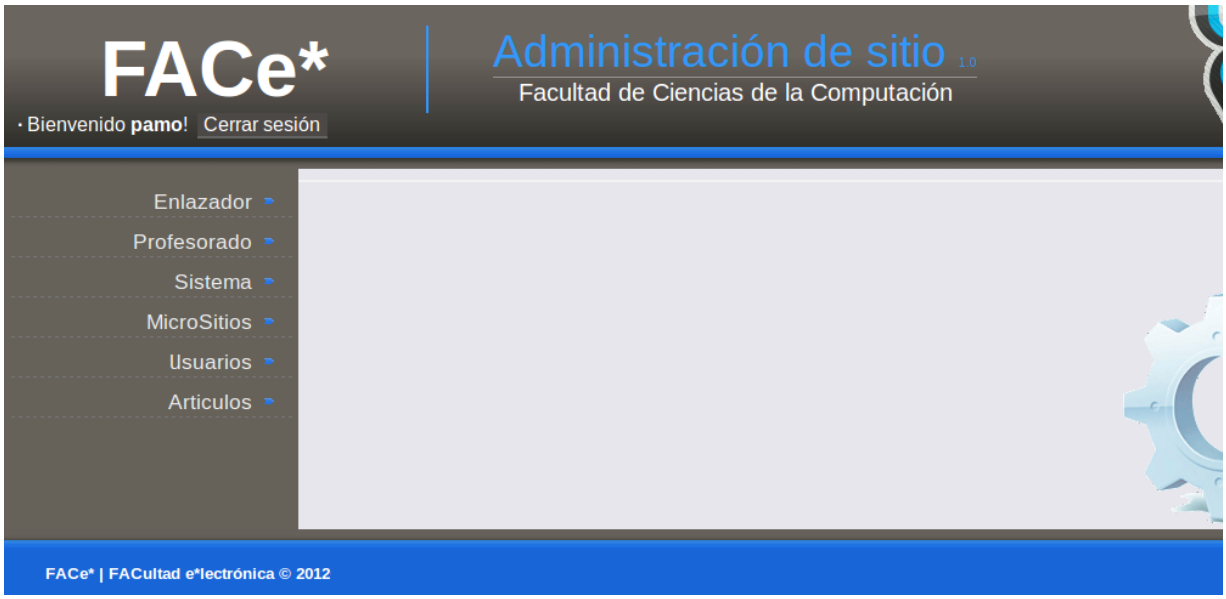

Fig. 4.44. Vista de la plantilla de administración

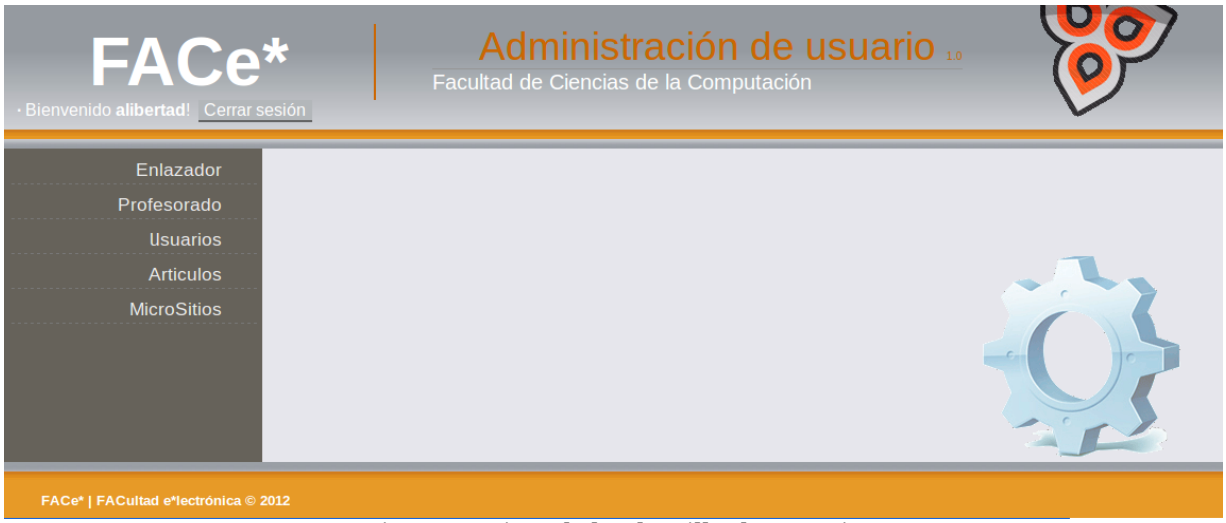

Fig. 4.45. Vista de la plantilla de usuario

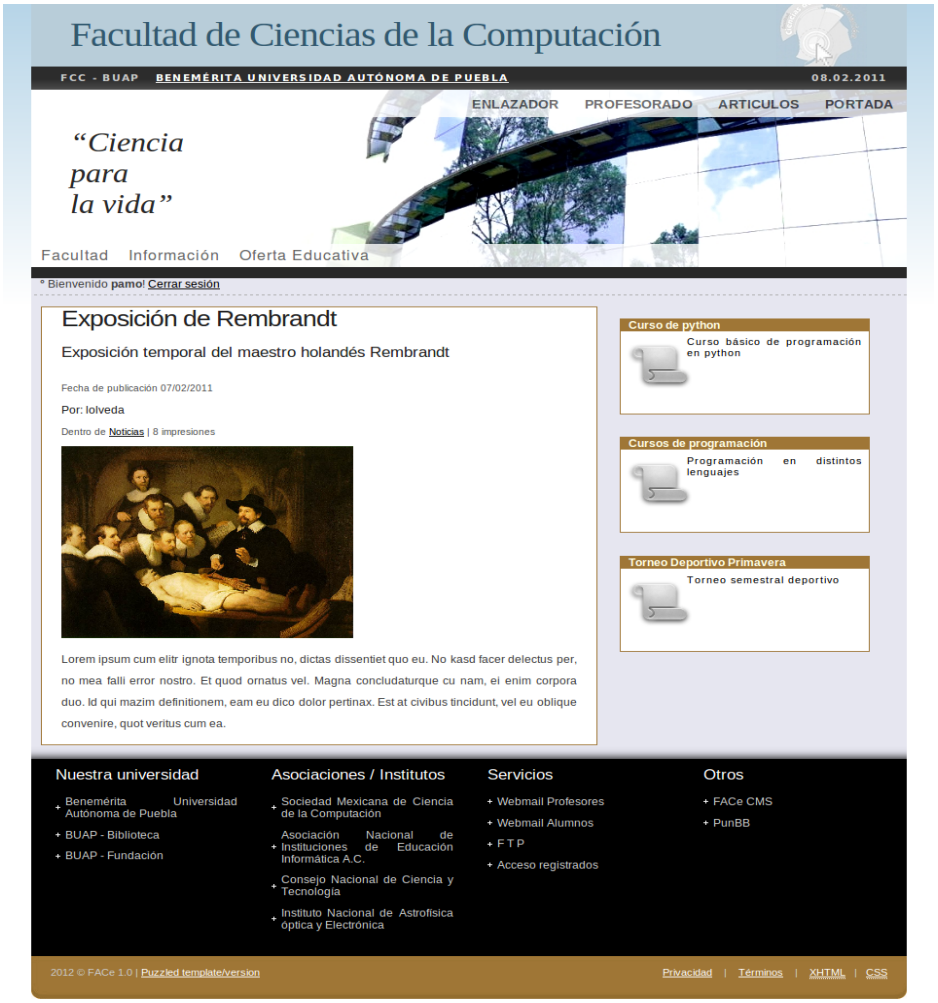

Fig. 4.46. Vista de la plantilla de aplicación desplegando la página principal

4.5.1 Presentación de contenidos – XHTML / CSS

Mientras HTML es un lenguaje de presentación con la estructura basada en SGML, XHTML es la visión de una presentación con estructura XML. Esta característica da orden, separa la presentación del contenido y posibilita la inserción de otros lenguajes basados en XML sobre el mismo documento.

El lenguaje elegido para la presentación es XHTML, pero, ¿Por qué elegir un lenguaje menos permisivo? XHTML da razones para ser elegido, dentro de las cuales podemos encontrar [41]:

Es más fácil de manipular

XHTML es una aplicación XML, lo que hace que sea sencillo crear herramientas que trabajen sobre documentos XHTML. También un número de estándares modernos se han desarrollado para manipular el XML como XSLT, Xpath, Xquery, DOM y SAX.

Funciona mejor sobre dispositivos móviles

Por razones históricas, los navegadores aún necesitan desplegar HTML aunque sea inválido. Sin embargo bajo XHTML las reglas son estrictas, esto hace a los navegadores de XHTML ser mucho más simples, pues no hay necesidad de crear analizadores complejos.

Es extensible

La mayoría de los lenguajes de programación usan una cantidad relativamente pequeña de tipos de datos y permiten agregar nuevas características a través de librerías. XHTML toma un acercamiento similar definiendo un grupo base de elementos y atributos.

Hay que considerar además que, siendo XHTML menos permisivo que otras versiones de HTML, esta puede cambiarse fácilmente modificando la configuración que indica al sistema servir como XTML, algo similar sucede cuando el navegador no reporta que puede desplegar XHTML.

4.6 Mejoras con Javascript

Un sitio se puede ver muy beneficiado al usar Javascript, sin embargo hay que tomar en cuenta que puede no estar disponible por distintas causas y que por tanto las funcionalidades no deben depender de él normalmente, de modo que al desarrollar la vista los esfuerzos deben dirigirse a conseguir un despliegue correcto en modo (X)HTML pero considerando que algunos de los elementos agregados deberán de poder extenderse de manera sencilla y no intrusiva con JS.

La importancia de tener o no Javascript se ve afectada por el rol que tiene el usuario. Es comprensible que las áreas de administración contengan funcionalidades dependientes completamente de esta tecnología, mientras que para un visitante, la ausencia de Javascript es enteramente justificable.

Se incluye también la versión 1.7 comprimida del framework prototype que tiene con una velocidad de acceso a objetos del DOM superior a los frameworks comunes como Jquery o Mootools (http://mootools.net/slickspeed/). Prototype es usado en scripts más complejos como el validador en Javascript.

Javascript para la administración y el usuario

El modo de administración y de usuario contienen Javascript para mejorar la usabilidad. Javascript puede mejorar significativamente la experiencia de uso del usuario e incrementar la productividad, e incluso disminuir la probabilidad de errores.

Un cuadro de texto puede verse muy favorecido agregando funcionalidades que faciliten el llenado como un selección de color en caso de necesitar ese tipo de valores (Fig. 4.51). Se ha visto en la descripción de los módulos que la mayoría de las áreas tienen plugins para facilitar la creación de código (X)HTML o PHP, en caso de que Javascript no esté disponible se despliegan áreas de texto comunes que no pierden la función de entrada de texto (Fig. 4.47, 4.48). Otro de los casos más comunes es la validación de campos del lado del cliente antes que de el servidor para ahorrar tiempo y envío de datos innecesarios (Fig. 4.49).

Algunas funcionalidades particulares pueden ser también implementadas de manera sencilla y consiguiendo efectos que bien valen su desarrollo, un ejemplo puede ser cargar imágenes de manera dinámica sobre la misma página que despliega su acceso (Fig. 4.50).

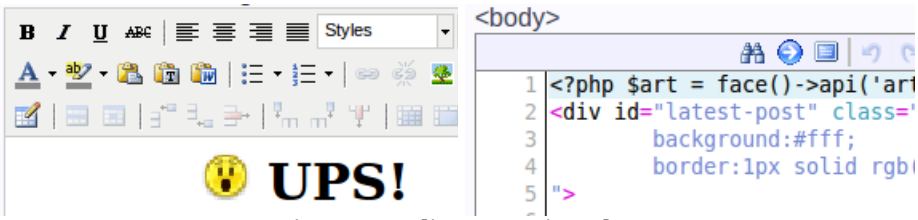

Fig. 4.47. Editores mejorados

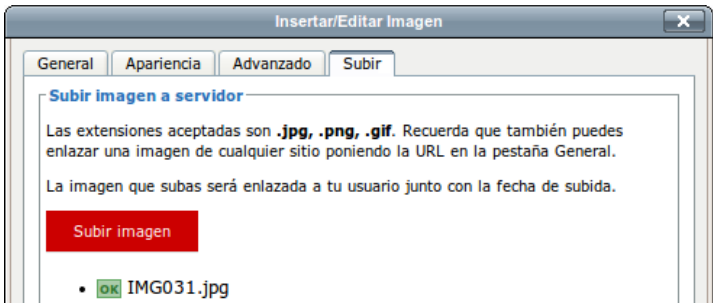

Fig. 4.48. Subiendo imágenes desde el editor de texto

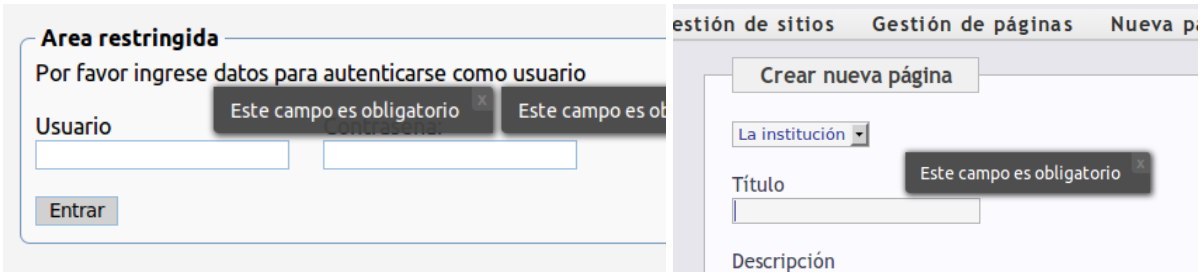

Fig. 4.49. Validación de campos

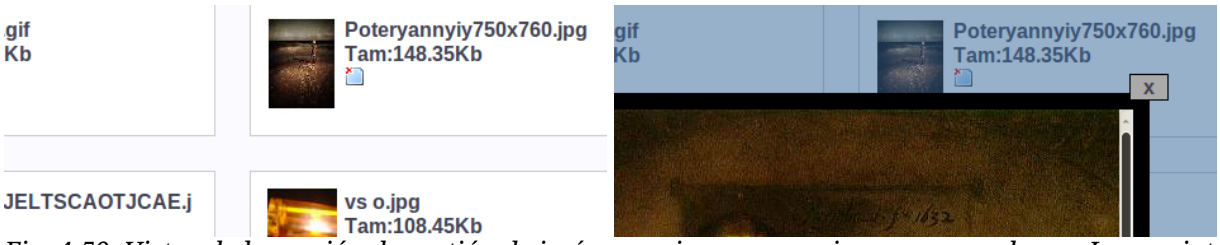

Fig. 4.50. Vistas de la sección de gestión de imágenes sin y con una imagen cargada por Javascript

Fig. 4.51. Selector de color en perfil de profesor

Javascript en la aplicación

El modo de aplicación no contiene prácticamente Javascript a excepción del menú principal que agrega los comportamientos ocultación y despliegue en los submenús con Javascript en el caso de navegadores IE antiguos, que no pueden desplegar correctamente algunos elementos de CSS; algo que ocasiona que el menú se despliegue en su totalidad sin ocultar las entradas que no tienen el puntero encima.

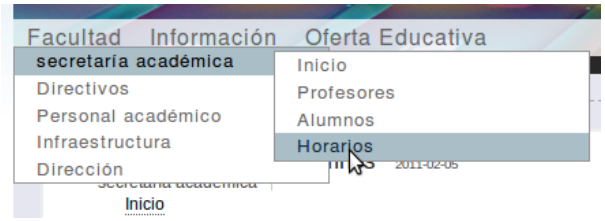

Fig. 4.52. Menú del sitio

CONCLUSIONES

Confirmación de objetivos

Se ha conseguido desarrollar el sistema que se planteó en un inicio, cumpliendo los objetivos propuestos.

Se ha obtenido un CMS que es capaz de generar un sitio que además de solucionar los requerimientos de la misma manera en que lo hace el sitio actual, provee un conjunto de herramientas y funcionalidades con las que la administración será más sencilla y agradable.

Usando el CMS desarrollado es posible generar enteramente el sitio de la Facultad de Ciencias de la Computación con todos sus contenidos, considerando que la forma en que se almacena y gestiona la información es diferente. Esto por supuesto no es una limitante sino un cambio de paradigma que da una mejor solución donde además de facilitar la administración desde diferentes perspectivas facilita la portabilidad.

Como cada módulo está pensado para solucionar principalmente las necesidades actuales, el CMS ofrece una solución integral que considera la administración de distintos tipos de datos, que aunque forman parte del mismo sitio, deben ser gestionados de diferente manera, e incluso por diferentes usuarios o roles. Acerca de los módulos, cabe mencionar que se ha implementado el mecanismo para que diferentes usuarios, de la misma u otra instancia puedan trabajar sobre los documentos de un proyecto de manera colaborativa, permitiendo además desplegar de manera selectiva los documentos que se deseen hacer disponibles a los visitantes en general, ofreciendo una herramienta explotable en proyectos inter-facultativos por las facultades que integran a la DESIT-BUAP.

El manejo de un CMS demuestra ser la mejor alternativa para administrar un sitio de manera consistente y eficaz, incrementando la productividad de una forma simple.

Limitaciones

El CMS es un gestor de sitio. Esto es suficiente para la administración del sitio de la Facultad de Ciencias de la Computación. Con esta premisa se justifica el desarrollo de un CMS especializado, sin embargo el sitio de una universidad puede tener mayores requerimientos y funcionalidades. Para usar el sistema deben considerarse las necesidades particulares sobre las funcionalidades disponibles.

El sistema sirve para administrar un sitio, no un conjunto de sitios. En caso de requerir más de un sitio, se pueden instalar más instancias del sistema sobre el mismo servidor, pero serán administradas de manera independiente.

Acerca del conjunto de funcionalidades para trabajos colaborativos, no se cuentan con editores avanzados como los administradores de documentos Google Docs que permiten editar directamente archivos de texto u hojas de cálculo, funcionalidades que pueden ser agregadas en caso de que sean necesarias.

Perspectivas

El sistema está diseñado contemplando diferentes tipos de cambios y mejoras.

Es posible la implementación de un sistema de idiomas. Los archivos con extensión *.lng.php* están reservados para poder ser usados como parte de la implementación de idiomas, conteniendo las etiquetas en un idioma en particular. Como la implementación separa la capa de la vista, es fácil crear archivos con distintos idiomas y hacerlos útiles dentro del sistema. Para implementar el idioma, se almacenaría el archivo en la carpeta mod de su módulo y desde el controlador o la vista cargar dicho archivo. Además de esto, se tendrían que cambiar las cadenas con los textos fijos en la vista por las constantes definidas en el archivo de idioma.

La conexión e interacción con la base de datos se hace únicamente a través de la interfaz para este propósito (la librería sql.lib.php), que siempre regresa la información solicitada en estructuras nativas de PHP. Este método de interacción con la base hace que sea posible crear diferentes implementaciones de la librería que utilicen otros manejadores de bases de datos.

Las funcionalidades que implementan Javascript son básicas y actualmente el sistema no hace mucho uso de este motor, sin embargo la estructura considera que en un futuro se puede desear agregar otros elementos, como librerías especiales de propósitos generales como jQuery, MooTools, etc.

La carga de diferentes hojas de estilo y carga de archivos .js es sencilla de implementar utilizando funciones del framework para este fin, de manera que se pueden hacer adaptaciones a la imagen y comportamiento sin romper el esquema de implementación del sistema.

La separación del código está establecida de forma que sea fácil identificar la ubicación de códigos en particular para facilitar la mejora y crear por terceros nuevas funciones dentro de las librerías existentes o creando nuevas, algo aplicable también a las API.

Las funcionalidades están organizadas en modos, estos modos separan las funciones por el tipo de usuario que hace uso de ellas, como el modo de usuario donde están las funciones administrables por un usuario en algún módulo; bajo este esquema pueden desarrollarse de manera ordenada nuevos modos para cubrir nuevas necesidades como el mencionado servicio web.

IMPRESIONES Y COMENTARIOS

Se ha enviado un resumen de este documento, presentando las funcionalidades que se han descrito a diferentes profesores que tienen a su cargo algún departamento que puede verse beneficiado con alguna funcionalidad implementada, ésto con el fin de tener la impresión acerca del sistema hecho.

M.C. Meliza Contreras (Educación Continua)

Tiene una opinión buena del sistema, haciendo el comentario de que en el diagrama de sitios (Fig. 4.31) no aparece el acceso a Educación Continua. El diagrama expuesto, presenta la estructura actual del menú del sitio, en el que esa entrada efectivamente no existe. La observación es importante pues demuestra que con una administración manual se dificulta el crecimiento y la adaptación de manera consistente, como en este caso, donde se ha agregado una liga con el acceso en la página de inicio, pero que no está disponible desde el menú por lo que no es posible acceder a esa ubicación si se está desplegando cualquier página fuera del inicio.

M.C. Consuelo Molina (Diplomados)

Opina que, con el sistema presentado, utilizando los mecanismos implementados pueden enriquecerse de manera más sencilla los contenidos, algo necesario frente a los constantes cambios en la información que contiene el sitio y conforme a la evolución de la institución, contrastando la manera en que actualmente se trabaja con el sitio de la Facultad, esto sin perder de vista que el alcance del sistema es para solucionar los problemas que tiene actualmente el sitio y su administración.

Acerca del mantenimiento rutinario hace el comentario de la importancia de tener la habilidad de poder realizar el mantenimiento de diferentes elementos y por diferentes personas; por un lado el poder trabajar elementos del estilo por un administrador general, pero dando la posibilidad de que sean los encargados de diferentes áreas o departamentos quienes administren los contenidos que les corresponden, haciendo más dinámico y sencillo el trabajo.

Considera que puede ser un sistema que puede trabajar en conjunto con otros, como el sitio de vinculación y convertirse en una solución más completa que de por resultado una mejor exposición de la Facultad de Ciencias de la Computación.

M.C. José Alfonso Garcés (Vinculación)

Los comentarios del profesor Alfonso Garcés, hondean sobre el estado actual y los problemas implicados, y la posibilidad de solventar las necesidades que no son resueltas con el sitio actual.

De los problemas más grandes observados por el profesor se encuentra la falta de una estructura a modo de columna vertebral que administre de una manera central los contenidos (y otros elementos) de diferentes coordinaciones y que de mismo modo permita un acceso común a esos y otros recursos similares, sin olvidar la presentación consistente, problema presente en diferentes áreas, donde el sitio de cada departamento tiene una vista diferente. Otro problema mencionado es el hecho de tener que solicitar modificaciones del sitio a un administrador en caso de necesitar alterar alguna sección que está al cargo propio.

La opinión general es positiva y considera que se pueden realizar las tareas necesarias adecuadamente con las funcionalidades ofrecidas. Encuentra que existe la flexibilidad faltante en el sistema anterior para acceder a los contenidos y gestionarlos de una manera controlada por permisos, de modo que fácilmente se puedan agregar nuevas páginas como 'Casos de Éxito'.

Con respecto a que otras funciones pueden ser consideradas, menciona que la generación de un esquema compatible con dispositivos móviles puede ser deseable. Otros comentarios refieren a otros módulos que permitan gestionar elementos internos, como una funcionalidad para chequeo remoto de clases directamente desde un dispositivo móvil por ejemplo en lugar de presentarse a firmar. Eso es interesante pues puede derivar en la creación de otros elementos como la generación de reportes tanto propios en caso de ser un profesor, como de la planta docente en general para un usuario con los permisos pertinentes.

REFERENCIAS

[1] *Web Portals: The New Gateways to Internet Information and Services*, p. 3
[2] *Content Management Systems*, cap. 1, The litany of pain
[3] *Content Management Systems*, cap. 2, Why Don't People Use Content Management?
[4] *Content Management Bible 2nd Ed*. What Is Content Management?, p. 76
[5] Ante, Spencer E. (2000-06-05). http://www.businessweek.com/archives/2000/b3684031.arc.htm Business Week. Ref. De 09 de Julio de 2011.
[6] *Content Management Systems*, The activities of a Content Management
[7] *Content Management Bible*, cap. 8: Knowing When You Need a CMS, p. 113
[8] *Professional Web 2.0 Programming*, cap. 2, p. 37
[9] *Content Management Bible*, cap. 3: Content Has Structure, p. 21
[10] *Web design, the Complete Reference*, Chapter 6: Site Types and Architectures , p. 167.
[11] *Content Management Bible*, Introducción p. 39
[12] *Content Management Systems*, cap. 4: CMS Map - Publishing
[13] *Software Engineering, A practitioners approach*, cap. 2: The Process, pp. 21,22
[14] *Scrum And Xp From The Trenches*, Splitting the productlog – or not?, p. 113
[15] *PHP 5 CMS Framework Development*, cap. 1: CMS Architecture – The Idea of a CMS, p. 9
[16] *PHP 5 CMS Framework Development*, cap. 1: CMS Architecture – Critical CMS Features, p. 10
[17] *Practical Web 2.0 Applications with PHP*, capítulo 1: Application Planning And Design, p. 8
[18] *Software Engineering, A practitioners approach*, cap. 10.5: Requirements Engineering, p. 256
[19] *PHP5 CMS Framework Development*: Model, View and Controller, p. 22
[20] *PHP5 Power Programming*, cap. 4: Design Patterns, p. 97
[21] *SQL for Smarties*, cap. 9: Normalization, p. 182
[22] *Database Modeling and Design*, cap. 5: Transforming the conceptual data model, p. 87
[23] *Modern Database Management*, cap. 5: Logical Database Design, p. 211
[24] *SQL for Smarties*, cap. 9: Normalization, p. 182
[25] *Architectural Blueprints – The "4+1" View Model of Software Architecture*, abstracto
[26] *PHP5 CMS Framework Development*, cap. 3: Organizing Code - Security, p. 58
[27] *Practical Web 2.0 Applications with PHP*, cap. 2: Setting Up the Application Framework, p. 16
[28] *PHP5 CMS Framework Development*, cap. 1: Security of a CMS, p. 26
[29] *Professional Web 2.0 Programming*, cap. 18: Security - Code Security, p. 469
[30] *Secure Applications of Low-Entropy Keys, Introduction — Why Stretch a Key?, Salt,* pp. 1,2
[31] *Cryptography in PHP*: use cases, diapositiva 15: bcrypt
[32] http://chargen.matasano.com/chargen/2007/9/7/enough-with-the-rainbow-tables-what-you-need-to-know-about-s.html, http://www.akkadia.org/drepper/sha-crypt.html
[33] http://www.usenix.org/events/usenix99/provos/provos.pdf, http://www.openbsd.org/papers/bcrypt-paper.pdf A Future-Adaptable Password Scheme, p. 6
[34] *Object Oriented Programming with PHP5*, cap. 3: More OOP, Autoloading Classes, p. 63
[35] *PHP 5 CMS Framework Development*, cap. 3: Organizing Code - Autoloading, p. 61
[36] *Introducción a CSS*, cap. 9: Listas, Estilos Avanzados, p. 131
[37] *Introducción a XHTML*, cap. 5: Listas, p. 69
[38] *Content Management Bible*, cap. 31: Designing Templates, p. 683
[39] *Introducción a CSS*, cap. 1: Introducción, p. 5
[40] *Introducción a XHTML*, cap. 3: Texto, p. 21
[41] *Professional Web 2.0 Programming*, cap. 2: Page Presentation, p. 44

BIBLIOGRAFÍA DETALLADA

Eric van der Vlist, A. Vernet, E. Bruchez, J. Fawcett, and D. Ayers. *Professional Web 2.0 Programming*. Wiley Publishing, Inc., 2007. 522 p. ISBN-13: 978-0-470-08788-6, ISBN-10: 0-470-08788-9

Hasin Hayder. *Object Oriented Programming with PHP5*. Packt Publishing Ltd., 2007. 255p. ISBN 978-1-847192-56-1

Phil Suh, D. Addey, D. Thiemecke and J. Ellis. *Content Management Systems*. Apress, 2003. 200 p. ISBN: 190415106X

Brampton, Martin. *PHP5 CMS Framework Development*. Packt Publishing, 2008. 328 p. ISBN: 978-1-847193-57-5

Gutmans, Andi. *PHP5 Power Programming*. Pearson Education, Inc, 2005. 689p. ISBN 0-131-47149-X

Bob Boiko. *Content Management Bible, 2nd Edition*. Wiley Publishing, Inc., 2005. 1122 p. ISBN: 0764573713

Arthur Tatnall. *Web Portals: The New Gateways to Internet Information and Services*. Idea Group Publishing, 2005. 364 p. ISBN 1-59140-438-X, 1-59140-439-8

Thomas A. Powel. *Web Design, the Complete Reference, 2nd Ed.* Mc Graw Hill, 2002. 901 p. ISBN: 0-07-222851-2

Eguíluz Pérez, Javier. *Introducción a XHTM* [en línea, publicación en formato pdf]. www.librosweb.es, 4 de octubre de 2008 [ref. de 17 abril 2010]. Disponible en http://www.librosweb.es/xhtml

Eguíluz Pérez, Javier. *Introducción a CSS* [en línea, publicación en formato pdf]. www.librosweb.es, 6 de octubre de 2008 [ref. de 17 abril 2010]. Disponible en http://www.librosweb.es/css

Philippe Kruchten. *Architectural Blueprints—The "4+1" View Model of Software Architecture* [en línea, publicación en formato pdf]. noviembre de 1995. [ref. de 17 de octubre de 2010]. Disponible en http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf

Quentin Zervaas. *Practical Web 2.0 Applications with PHP*. Apress, 2008. 572 p. ISBN-13 (electrónico): 978-1-4302-0474-9, ISBN-10 (electrónico): 1-4302-0474-5

Roger S. Pressman. *Software Engineering: A practitioner's approach*. Quinta edición. McGraw-Hill, . ISBN 0073655783

Henrik Kniberg. *Scrum And Xp From The Trenches* [publicación en formato pdf]. 2007 C4Media Inc. [ref. De 09 de septiembre de 2010]. Disponible en http://www.infoq.com/minibooks/scrum-xp-from-the-trenches ISBN: 978-1-4303-2264-1

Joe Celko. *SQL for Smarties, 4th Ed.* Morgan Kaufman Publishers, 2011. 787 p. ISBN: 978-0-12-382022-8

Toby Teorey, S. Lightstone, T. Nadeau, H. V. Jagadish. *Database Modeling and Design, 5th Edition*. Morgan Kaufman Publishers, 2011. 787 p. ISBN: 978-0-12-382020-4

Jeffrey A. Hoffer, M. B. Prescott, F. R. McFadden. *Modern Database Management, 8th Edition*. Pearsons Prentice Hall, 2007. 622 p. ISBN: 0-13-221211-0

J. Kelsey, B. Schneier, C. Hall, D Wagner, *Secure Applications of Low-Entropy Keys* [ref. De 5 de agosto de 2011]. Disponible en http://www.schneier.com/paper-low-entropy.pdf

Niels Provos, David Mazières. *A Future-Adaptable Password Scheme* [ref. De 5 de agosto de 2011]. Disponible en http://www.usenix.org/events/usenix99/provos/provos.pdf, http://www.openbsd.org/papers/bcrypt-paper.pdf

Enrico Zimuel. *Cryptography in PHP: Use Cases* (diapositivas). Zend/PHP Conference, Zend Technologies [ref. 20 de noviembre de 2011]. Disponible en http://www.zimuel.it/en/category/crittografia/