Universidad de las Ciencias Informáticas
Facultad 10

Título: Propuesta de procedimiento para la creación de portales web utilizando la tecnología Zope/Plone.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Aned corzo Leyva
Dayné Pérez Domínguez
Yuri Suárez Pérez

Tutor: Ing. Eneybis García Soto
Consultante: Lic. Hailem Dreis Carrasco Fuentes

Ciudad de La Habana

Junio, 2007

Los grandes trabajos no son hechos por la fuerza, sino por la perseverancia.
Ben Johnson

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de junio del año 2007.

_____ _____

Aned Corzo Leyva Ing. Eneybis García Soto

_____

Dayné Pérez Domínguez

_____

Yuri Suárez Pérez

AGRADECIMIENTOS

Agradecemos a la Revolución Cubana, por forjarnos como profesionales e inculcarnos las ideas revolucionarias que enseña nuestro Comandante en Jefe.

A Eneybis, nuestra tutora por su ayuda.

A Hailem, nuestro profesor consultante por guiarnos en la realización de este trabajo.

A todos los profesores que nos han ayudado: David, Leansy, Maidely, William por haber brindado tanto tiempo, por la gran ayuda que han ofrecido en la realización de este trabajo, y por su colaboración incondicional.

A nuestros profesores por formarnos como profesionales.

AGRADECIMIENTOS ESPECIALES

Agradecemos profundamente a todo el que de una forma o de otra ha contribuido a la realización de un sueño, pero especial, agradecemos a:

- Mi mamá por ser mi guía, mi amiga y mi tesoro más grande, por todo su apoyo incondicional en todos los momentos difíciles por los cuales atravesé en mis años de estudiante. A ti te debo mi vida y el haber llegado hasta aquí. Te amo!!!
- Mi papá por su amor y por haber depositado tanta confianza en mi.
- Mi hermanito David por quererme y soportarme como soy.
- Mi hermana Maylen por su amistad, apoyo, confianza, por estar siempre ahí.
- Roberto, mi novio por su amor, ayuda, comprensión, por estar a mi lado este tiempo soportando mis malcriadeces. Por soñar junto a mí.
- Mi familia toda que me ha brindado su cariño y apoyo.
- Todas mis amigas, amigos y compañeros de grupo y de proyecto por su apoyo y ayuda.
- Todas las personas que me han ayudado y que no puse, pero no por no acordarme de ellos, sino por el espacio.

…les estoy muy agradecida.

Aned Corzo Leyva

Agradecimientos

- Neida Domínguez Martínez, mi mamita linda, por todo su amor, comprensión y ayuda en todos los momentos difíciles que pase en mi carrera. Por querer con tantas ganas que me gradúe y sea una profesional más completa. Por regañarme cuando más lo necesitaba aunque yo en ese momento no la entendiera. Por llorar, ponerse nerviosa conmigo con esta tesis. Por darme la vida, te quiero mucho mami.
- Mi papito por apoyarme todos los días, porque se que esta nervioso aunque no me lo diga. Papi, te agradezco que hayas sido tan duro conmigo, porque me enseñaste que en la vida hay que luchar para conseguir lo que uno quiere. Te quiero mucho papi.
- Luismi, mi neni, por darme tanto amor y alentarme cuando estaba desesperada, cuando tenía tantas cosas en la cabeza. Por soportar mis majaderías, mis malcriadeces y mis nerviosismos con la pre-defensa. Por toda su atención y comprensión. Por tenerme siempre presente. Te Amo!!!!
- Mi doctora, Gloria Lara, por su apoyo, su guía constante, la seguridad y confianza que hizo tener en mi misma, muchas gracias por todo.
- Mis amigos, compañeros de aula y de proyecto por la ayuda prestada, Joel, Reinier, Raydel, Ailin, Maikel.

…a todos muchas gracias.

Dayné Pérez Domínguez

- Mis padres por haber brindado todo su apoyo y dedicación durante los cinco años de la carrera.
- Mis amigos especialmente a Gilberto y Yeisel por su ayuda desinteresada.

…muchas gracias

Yuri Suárez Pérez

RESUMEN

En la nueva y cada vez más rápida era digital en la que se está viviendo los Sistemas Manejadores de Contenido cobran gran importancia y son cruciales para que una página web opere eficientemente. Estos sistemas en la actualidad se están volviendo imprescindibles para todos los sitios web dinámicos que tienen gran cantidad de contenidos y funcionalidades ya que contribuyen a que el sitio pueda ser modificado de forma rápida y segura desde cualquier computadora conectada a Internet. En el proyecto "Informatización de la prensa" perteneciente la Universidad de las Ciencias Informáticas se están desarrollando actualmente portales web utilizando al Sistema Manejador de Contenidos, Plone y a su servidor Zope, pero no se cuenta con una guía que permita orientar al equipo que va a trabajar en el desarrollo de los portales para una mejor organización y uniformidad de su trabajo. En la presente investigación se propone un procedimiento para la creación de portales web utilizando la tecnología Zope/Plone con el fin de indicar al equipo de desarrollo qué hacer y qué obtener en cada flujo de trabajo por los que van a pasar los portales durante su ciclo de vida de desarrollo. Este procedimiento sirve de guía a los equipos de desarrollo, mostrando lo que se debe hacer y obtener en cada flujo de trabajo, para así lograr la uniformidad y un producto final de acuerdo a los requerimientos de los clientes, de forma eficiente, con calidad y en tiempo.

PALABRAS CLAVES

Informática, tecnología Zope/Plone, portales web

Índice

ÍNDICE GENERAL

INTRODUCCIÓN 1

Capítulo 1 Fundamentación teórica 5

1.1 Portal web 5

1.2 Tendencias y tecnologías actuales 6

1.2.1 Content Management System 6

1.2.2 Zope 10

1.2.3 Python 14

1.2.4 CSS 16

1.2.5 XHTML 17

1.2.6 Plone 18

1.3 Metodologías para el desarrollo de aplicaciones web 19

1.3.1 WSDM 20

1.3.2 WebML 22

1.3.3 UML Based Web Engineering 24

1.3.4 OOWS 25

1.3.5 Rational Unified Process 26

1.4.6 eXtreme Programming 28

Capítulo 2 Propuesta de procedimiento para la creación de portales web utilizando la tecnología Zope/Plone 33

2.1 Análisis de las metodologías existentes. 33

2.2 Bases de la propuesta 36

2.3 Propuesta de procedimiento 36

2.3.1 Especificación de los requisitos 37

2.3.2 Análisis 42

2.3.3 Diseño 48

2.3.4 Implementación 52

2.3.5 Prueba 55

Capítulo 3 Puesta en práctica del procedimiento 61

3.1 Catálogo de requisitos 61

Índice

3.2 Documento del análisis 72
3.3 Diseño 74
3.4 Implementación 80
CONCLUSIONES 83
RECOMENDACIONES 84
REFERENCIAS BIBLIOGRÁFICAS 85
BIBLIOGRAFÍA 87
GLOSARIO DE TÉRMINOS Y SIGLAS 90

ÍNDICE DE FIGURAS

FIGURA 1: ZOPE MANAGEMENT INTERFACE 11
FIGURA 2: ARQUITECTURA DE ZOPE 12
FIGURA 3: ESQUEMA DE FASES DE WSDM 21
FIGURA 4: ESQUEMA DEL CICLO DE VIDA DE WEBML 23
FIGURA 5: PROCESO DE DESARROLLO DE UWE 24
FIGURA 6: FASES Y FLUJOS DE TRABAJO DE RUP 27
FIGURA 7: FLUJO DE TRABAJO PARA LOS CONTENIDOS 46
FIGURA 8: FLUJO DE TRABAJO PARA LAS CARPETAS 47
FIGURA 9: EJEMPLO DE MODELO CONCEPTUAL 48
FIGURA 10: EJEMPLO DEL DISEÑO ESTRUCTURAL DE UN SITIO WEB 51
FIGURA 11: EJEMPLO DE DIAGRAMA DE DISEÑO 52
FIGURA 12: EJEMPLO DIAGRAMA DE IMPLEMENTACIÓN 54
FIGURA 13: MODELO DE CASOS DE USO 64
FIGURA 14: FLUJO DE TRABAJO PARA LOS CONTENIDOS 72
FIGURA 15: MODELO CONCEPTUAL 73
FIGURA 16: PROTOTIPO DE INTERFAZ DE LA PÁGINA PRINCIPAL 74
FIGURA 17: PROTOTIPO DE INTERFAZ DE LA TEMÁTICA “ACTUALIDAD” DEL MENÚ PRINCIPAL 75
FIGURA 18: PROTOTIPO DE INTERFAZ DE LAS NOTICIAS 76
FIGURA 19: ESTRUCTURA GENERAL DEL SITIO 77
FIGURA 20: ESTRUCTURA DE LAS SECCIONES 78
FIGURA 21: ESTRUCTURA DE LAS NOTICIAS A TRAVÉS DE LAS CARPETAS PERSONALES 79
FIGURA 22: DIAGRAMA DE DISEÑO 80
FIGURA 23: DIAGRAMA DE IMPLEMENTACIÓN 81
FIGURA 24: DIAGRAMA DE IMPLEMENTACIÓN DEL COMPONENTE PORTAL_SKINS 81

ÍNDICE DE TABLAS

TABLA 1: PATRÓN PARA LA DESCRIPCIÓN DE LOS ACTORES DEL SISTEMA 40
TABLA 2: PATRÓN PARA LA DESCRIPCIÓN DE LOS CASOS DE USO 41
TABLA 3: DESCRIPCIÓN DE LOS ACTORES DEL SISTEMA 65
TABLA 4: DESCRIPCIÓN DEL CASO DE USO: AUTENTICAR 66
TABLA 5: DESCRIPCIÓN DEL CASO DE USO: GESTIONAR INFORMACIÓN 67
TABLA 6: DESCRIPCIÓN DEL CASO DE USO: GESTIONAR SECCIÓN 68
TABLA 7: DESCRIPCIÓN DEL CASO DE USO: PUBLICAR CONTENIDOS 70
TABLA 8: DESCRIPCIÓN DEL CASO DE USO: REVISAR CONTENIDOS 71

INTRODUCCIÓN

Los portales web están dirigidos a satisfacer las necesidades específicas de un grupo de usuarios, ofreciéndoles de forma fácil el acceso a una serie de recursos y servicios como foros, buscadores, compras y correo electrónico, etc. En un mundo tan competitivo y cambiante como es el mundo de hoy el contenido constituye una parte muy importante en cualquier portal web, por tanto mantenerlo actualizado y amigable de forma eficiente es imprescindible.

Al principio de Internet, los administradores web (webmasters) eran los encargados de proveer de información a los portales, pero con el desarrollo de la red esta forma de suministro se ha vuelto obsoleta, puesto que la cantidad de datos a publicar ha aumentado y variado grandemente. Los portales web han ido evolucionando y se ha pasado de modelos basados en páginas estáticas a aplicaciones web muy complejas que gestionan gran cantidad y multiplicidad de contenidos en diferentes idiomas, integran aplicaciones de colaboración entre los usuarios, etc.

La necesidad de gestionar la información haciéndola accesible y de forma ágil para los usuarios, es una tarea primordial para todas las empresas y organizaciones, pero a su vez puede ser un trabajo arduo y complicado si no se tienen las herramientas adecuadas. Para solucionar esto en la actualidad, debido a la lógica evolución de la web, existen los Sistemas de Gestión de Contenidos (Content Management System o CMS). Estos son herramientas que permiten la creación, mantenimiento, publicación, presentación y actualización de sitios web rápida y fácilmente, sin la necesidad de expertos. Los gestores de contenidos ofrecen un grupo de ventajas a los usuarios, estas se mencionarán a continuación y fueron expuestas por Cuerda y Minguillón en su artículo “Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto” (Cuerda y Minguillón, 2004).

- Inclusión de nuevas funcionalidades en la web.
- Mantenimiento de gran cantidad de páginas.
- Reutilización de objetos o componentes.
- Páginas interactivas.
- Cambios del aspecto de la web.
- Mecanismos de control de acceso.

Introducción

La Universidad de las Ciencias Informáticas (UCI) no está al margen de todos los cambios que se están produciendo a nivel mundial en las TICs y fundamentalmente en el desarrollo que se está manifestando en las herramientas para crear y mantener portales web. En la UCI se están empleando actualmente varios tipos de CMS como entornos colaborativos de trabajo, aulas virtuales y para la creación de portales web. En el proyecto "Informatización de la prensa" perteneciente a la Facultad 10 se están desarrollando portales para la prensa cubana y se trabaja con Plone que es uno de los tantos gestores de contenidos de código abierto que existen en el mundo informático y con Zope (Z Object Publishing Enviroment) que es el framework sobre el cual está construido. Hasta el momento están en fase de terminación tres de estos portales, tras un arduo esfuerzo de los desarrolladores que se han encontrado gran cantidad de problemas, entre los más notables se encuentran:

- Poco conocimiento de la tecnología debido a que es relativamente nueva.
- Escasez de documentación y por tanto los conocimientos se van adquiriendo a partir del estudio del código de todo lo que está implementado en Plone y en los foros de discusión internacionales.
- Carencia de expertos locales (personas a las que se les pueda consultar una duda).
- Falta de una guía que permita desarrollar productos con esta tecnología.

Uno de los problemas mencionados es que no cuentan con una guía que permita guiar a los estudiantes y profesores participantes en dicho proyecto para que sepan cómo organizar su trabajo de manera efectiva, y esto trae consigo varias consecuencias que de una forma o de otra pueden frenar el buen desenvolvimiento del equipo de trabajo y el desarrollo de los portales. La inexistencia de un método a seguir para crear aplicaciones web utilizando las tecnologías antes mencionadas imposibilita:

1. Uniformidad en el trabajo.
2. Capacitación y asimilación de la tecnología por parte del personal.
3. El logro de un producto final de forma eficiente, en tiempo y con calidad.

Luego de un análisis de la situación actual en el proyecto, impera el siguiente **problema**: ¿Cómo diseñar un procedimiento de trabajo para el desarrollo de portales web utilizando la tecnología Zope/Plone que facilite la organización del trabajo, que se pueda cumplir con los compromisos de tiempo y que se satisfagan los requerimientos del cliente? Este problema está enmarcado en el **objeto de estudio**: el desarrollo de portales web, y dentro de él, como **campo de acción** se define: el desarrollo de portales

web utilizando la tecnología Zope/Plone.

Se plantea la siguiente **hipótesis** para llevar a cabo la investigación: si se confecciona una propuesta de procedimiento para el desarrollo de portales web usando la tecnología Zope/Plone, entonces el desarrollo de los proyectos que se realicen utilizando esta tecnología, mejorará en cuanto a la organización del trabajo, cumplimiento de los compromisos de tiempo y en la satisfacción de los requerimientos del cliente.
Para darle solución al problema existente, en el presente trabajo se plantea como **objetivo general**: diseñar un procedimiento para el desarrollo de portales web utilizando la tecnología Zope/Plone.
Del objetivo general antes mencionado se derivan los siguientes **objetivos específicos**:

1. Investigar sobre las diferentes tendencias y tecnologías actuales más usadas para el desarrollo de portales web.
2. Estudiar la herramienta Zope.
3. Estudiar al CMS Plone.
4. Realizar un estudio de las diferentes metodologías que se utilizan para el desarrollo de portales web.
5. Diseñar una propuesta de procedimiento para el desarrollo de portales web utilizando la tecnología Zope/Plone.

Para alcanzar los objetivos propuestos, se llevarán a cabo las siguientes **tareas**:

- Estudio de las tecnologías y estándares que componen a la tecnología Zope/Plone.
- Análisis y definición de las características, funcionalidades y beneficios de Zope.
- Definición de las características y beneficios de Plone.
- Análisis y crítica de las diferentes metodologías usadas para el desarrollo de portales web.

Se utilizaron los **métodos científicos** siguientes para darle cumplimiento a las tareas propuestas con anterioridad:

- Teórico
  - Analítico-sintético en el análisis de diversas metodologías y documentos de los cuales se extraerán los rasgos distintivos relacionados con el desarrollo de portales web.
  - Modelación, porque en este trabajo se creará una propuesta de procedimiento de trabajo.

❖ Empírico
- Observación, debido a que se pudo ver y constatar los problemas que existían debido a la carencia de un instrumento que guiara el proceso de desarrollo de portales web utilizando la tecnología Zope/Plone, tras estos problemas es que surge la idea de realizar este trabajo.
- Entrevista, porque se realizaron diversas entrevistas a diferentes personas que fueron muy útiles para el enriquecimiento del conocimiento adquirido.

Estructuración del contenido

El presente trabajo consta de 3 capítulos:

Capítulo 1 *Fundamentación Teórica***:** se abordan aspectos generales sobre los CMS y se analizarán las tendencias y tecnologías actuales que componen la tecnología Zope/Plone. Además se abunda y caracteriza algunas de las metodologías que existen en el mundo para la creación de portales web.

Capítulo 2 *Propuesta de procedimiento para la creación de portales web utilizando la tecnología Zope/Plone:* se realiza un análisis crítico de las metodologías abordadas. Se propone el procedimiento para la creación de portales web utilizando la tecnología Zope/Plone; se brindan las características generales del mismo así como algunas recomendaciones que debe de cumplir el equipo de desarrollo. Se dará también una explicación de cada flujo de trabajo con los que cuenta el procedimiento.

Capítulo 3 *Puesta en práctica del procedimiento*: se realiza la puesta en práctica del procedimiento utilizando un caso de ejemplo en el cual se obtendrán todos los artefactos a través de la realización de las tareas que propone el procedimiento para los flujos de trabajo de: especificación de requisitos, análisis, diseño y parte de la implementación.

Capítulo 1 Fundamentación teórica

Hablar del mundo informático de hoy y de las tecnologías más actuales utilizadas para crear portales web es algo complicado. Este capítulo está enfocado a sentar las bases en lo referente a las tendencias más novedosas y utilizadas que complementan la tecnología Zope/Plone para el desarrollo de portales web. Con el objetivo de facilitar la comprensión de las terminologías y herramientas utilizadas se profundiza en aspectos necesarios y características de las tendencias y tecnologías que serán usadas en el desarrollo este trabajo. Además se analizan diversas metodologías que existen para la creación de sitios web a nivel mundial que son las que guiarán este trabajo para alcanzar el objetivo propuesto.

1.1 Portal web

El término portal varía su significado entre las personas por lo que se pueden encontrar muchas definiciones de este término. Hay quién opina que un portal web es "(…) *una aplicación web que gestiona de forma uniforme y centralizada, contenidos provenientes de diversas fuentes, implementa mecanismos de navegación sobre los contenidos, integra aplicaciones e incluye mecanismos de colaboración para el conjunto de usuarios (comunidad) a los que sirve de marco de trabajo. Todo esto en un entorno web*". (Díaz) Otros lo entienden como "*una plataforma de despegue para la navegación en el web*." (Informática Milenium) Sin embargo para otras personas es "(…) *un punto de entrada a Internet donde se organizan sus contenidos, ayudando al usuario, y concentrando servicios y productos, de forma que le permitan a éste hacer cuanto necesite hacer en Internet a diario* (…)" (García y Saorín).

Lo cierto es que en un portal lo que impera es el volumen de información que se quiere transmitir al usuario mediante texto, imágenes, banners, enlaces y siempre tratando de que la navegación sea cómoda y fácil; por lo que uno de los objetivos más importantes de los portales web es distribuir información, publicarla, ponerla a disposición de los usuarios interesados en ella. En los portales es usual encontrarse servicios de foro, buscadores, chat, documentos, aplicaciones, entre otros.

A medida que Internet va adquiriendo popularidad las organizaciones y empresas ven una posibilidad de publicidad efectiva y de facilitar la comunicación con sus clientes, provocando la inclusión de bases de datos y de grandes cantidades de información en la red. Esto dio paso a las páginas interactivas en las

que se incluye información de los múltiples medios y con las que el usuario puede interactuar. Lo anteriormente dicho trajo como consecuencia, que los administradores de red se volvieran un mecanismo inadecuado, para la gestión de la gran cantidad de contenidos y datos que comenzaron a poblar las páginas web, debido a la gran velocidad de incorporación de información a los portales y a que los datos comenzaron a requerir más atención de los que el administrador podía ofrecerles; surgieron entonces, los Sistemas Gestores de Contenidos que facilitan el trabajo permitiendo distribuir la gestión de los contenidos entre varias personas y proporcionando de alguna manera que estas pudieran incluirlos en el portal.

1.2 Tendencias y tecnologías actuales

En este epígrafe se enfoca en abordar las diversas tecnologías y estándares que componen la tecnología Zope/Plone.

1.2.1 Content Management System

Los sistemas de gestión de contenidos son un tipo de software utilizados principalmente para facilitar la gestión de webs, ya sea en Internet o en una Intranet, y por eso también son conocidos como gestores de contenido web (WCM o Web Content Management).

Desde el punto de vista del usuario del sistema los CMS gestionan accesible y cómodamente un sitio web dinámico, que se puede actualizar periódicamente y en el cual pueden trabajar más de una persona; desde el punto de vista del cliente se trata de un sitio web dinámico con apariencia e interfaz uniforme que permite desarrollar fácilmente las tareas para las que ha sido diseñad.

El sistema consiste en una interfaz que controla una o varias bases de datos donde está ubicado el contenido. El sistema permite manejar por separado el contenido del diseño, por lo que es posible hacer cambios en el diseño del sitio y sin cambiar el formato dado al contenido del mismo.

1.2.1.1 CMS de código abierto y comercial

Los CMS pueden dividirse en CMS libres y CMS comerciales. Los CMS libres son desarrollados por un grupo de personas o empresas, tienen libre acceso para copiar, modificar o redistribuir y son de código abierto. Los CMS comerciales, son desarrollados por empresas que consideran al código como un activo más que tienen que mantener bajo propiedad y no se permite al acceso a terceras personas.

La posibilidad de ver el código fuente brinda la oportunidad de modificar, copiar y distribuir libremente el producto respetando los términos planteados en la licencia a la que pertenece, de esta forma se garantiza que el mismo pueda evolucionar aun si la compañía o el desarrollador que le dieron origen dejaran de existir.

Otra de la ventajas de los CMS de código abierto es el coste, generalmente todas las herramientas de software libre son gratis, excepto en algunos casos, pero siempre el coste va a ser menor que los CMS comerciales.

Con respecto al soporte los CMS comerciales brindan soporte profesional pero es muy caro, los de código abierto tienen la posibilidad de contar con una extensa comunidad de usuarios reunidos en foros donde se discuten problemas y se brindan soluciones.

Es posible encontrar CMS de código abierto en empresas que ofrecen servicios de valor añadido y con activas comunidades de usuarios. En el caso comercial también sucede, pero el coste de las licencias hace que el gran público se decante por otras opciones y por lo tanto las comunidades de soporte son más pequeñas.

Un problema que presenta el software de código abierto es la documentación, que es escasa, está dirigida a usuarios con conocimientos técnicos y muchísimas veces esta mal redactada.

1.2.1.2 Clasificación de los CMS

Si se busca en la Internet se puede encontrar una gran multitud de CMS, las cuales se agrupan en diferentes categorías. Debido al bloqueo que sufre el país, como apuesta por la filosofía del software libre y a las ventajas que supone el trabajo con los CMS de código abierto, en este epígrafe se presentan algunos de los CMS libres que existen. Para clasificarlos se utiliza la propuesta de tipología dada por Tramullas por ser la clasificación más completa de acuerdo a la opinión del equipo de trabajo, y es la siguiente (Tramullas, 2005):

- Plataformas para desarrollo de gestión de contenidos

La importancia de este tipo de CMS está en la posibilidad que ofrecen para construir soluciones adaptadas a cada caso, estas pueden ser de gestión de contenidos, de comercio electrónico, entre otras. Ofrecen un entorno y unas herramientas de desarrollo que posibilitan la creación e integración de módulos realizados por terceros. Algunas de las plataformas más extendidas son:

- OpenCMS
- Apache Lenya
- Zope

- Plataformas para desarrollo de portales

Una de las funciones principales es la creación y mantenimiento de portales, como herramienta básica de los servicios de información web. Su funcionalidad, administración y mecanismos de control están orientados a ofrecer al usuario portales con diferentes tipos de contenidos y de servicios, desde la publicación de noticias, repositorio de documentos, foros, creación de perfiles y de grupos de usuarios, etc. Siguen una arquitectura modular porque se componen de módulos, encargados de diferentes funciones, que son administrados desde una interfaz centralizada. Los más destacados son:

- PHP Nuke
- Drupal
- Plone (requiere Zope)

- Aula virtual

Son entornos que ofrecen las prestaciones necesarias para la creación de contenidos para el aprendizaje en línea, y ciertos mecanismos de interacción, como foros, chats, evaluación interactiva, etc. La mayoría de ellos siguen esquemas de aula clásica, traspasados al entorno web. Este tipo de plataformas han ajustado las prestaciones de gestión de contenidos a las características del material docente en formato digital. Es altamente recomendable que la plataforma que se utilice sea capaz de empaquetar sus contenidos en formatos SCORM y/o IMS. Los más conocidos son:

- Claroline
- Moodle

- Bibliotecas digitales

El desarrollo de colecciones digitales, la organización y creación de los mecanismos de acceso o la gestión de metadatos se benefician de la combinación de los procesos de publicación digital, y de los principios de gestión de información. Tienden a configurar espacios de colaboración, mediante el acceso y la gestión de colecciones distribuidas. El estado actual de las principales herramientas es diverso, así

como sus enfoques, ya que van desde repositorios federados de documentos hasta herramientas de usuario final. Este tipo de herramientas son más exigentes, en sus requerimientos, que los otros tipos indicados en este texto.

- ❖ Fedora
- ❖ Dspace
- ❖ Greenstone

- Publicaciones digitales

Son plataformas especialmente diseñadas teniendo en cuenta las necesidades de las publicaciones digitales, tales como periódicos, revistas, etc. Sus prestaciones están orientadas al control de los procesos de edición, creación y publicación de contenidos, más que al desarrollo de aplicaciones o portales.

- ❖ Cofax
- ❖ Open Journal Systems

- Entornos para colaboración

Corresponde a la definición de groupware, herramientas para trabajo en grupo. Su objetivo es dar a grupos de usuarios especializados las prestaciones necesarias para llevar a cabo trabajos y proyectos en común. En este tipo de entornos, los flujos de trabajo, los usuarios, los puntos de control y los documentos entregables son los contenidos claves a gestionar. Además, se necesitan herramientas de control de tiempos y actividades junto a posibilidades de comunicación síncrona y asíncrona. Se trata, entonces, de una especialización de la gestión de contenidos que se relaciona con la gestión de proyectos.

- ❖ eGroupware
- ❖ phpCollab
- ❖ Wiki

- Blogs o bitácoras

Son sitios periódicamente actualizados que recopilan cronológicamente textos o artículos de uno o varios autores. Frecuentemente los usuarios escriben sus comentarios y el autor los responde estableciendo un diálogo. Muestran un modelo de gestión de contenidos bastante simplificado, ya que suelen ser

monousuario, y con un sencillo flujo de trabajo, lo que ha facilitado su expansión entre amplios grupos de usuarios sin conocimientos técnicos profundos.

- WordPress
- Blosxom
- Serendipity

1.2.2 Zope

Zope es un servidor de aplicaciones web de código abierto escrito en Python, un lenguaje script interpretado orientado a objeto. Zope, de acuerdo a sus iniciales "Z Object Publishing Enviroment" es un entorno para publicar objetos y se utiliza también como una plataforma de desarrollo rápido de aplicaciones que son ricas en contenido y funcionalidad.

La licencia de Zope (ZPL) es una licencia de software libre y aunque es incompatible con la GPL ha permitido que miles de desarrolladores en todo el mundo puedan descargarlo, modificarlo, agregarle funcionalidades, productos sin ningún coste.

Zope corre en las plataformas de sistemas operativos más difundidas: GNU/Linux, Windows NT/2000/XP, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X. Permite a los desarrolladores crear aplicaciones web con el solo uso de un navegador web, puede ser Mozilla, Internet Explorer, Netscape, Opera, todos compatibles para mostrar y manejar el entorno de desarrollo de Zope.

Zope cuenta con una interfaz web llamada ZMI (Zope Management Interface), mostrada en la figura 1 que permite la creación y manipulación de objetos de una manera sencilla, rápida y amigable. Se torna muy útil para la realización de tareas de administración, gestión de contenidos, creación de usuarios, asignación de permisos, etc.

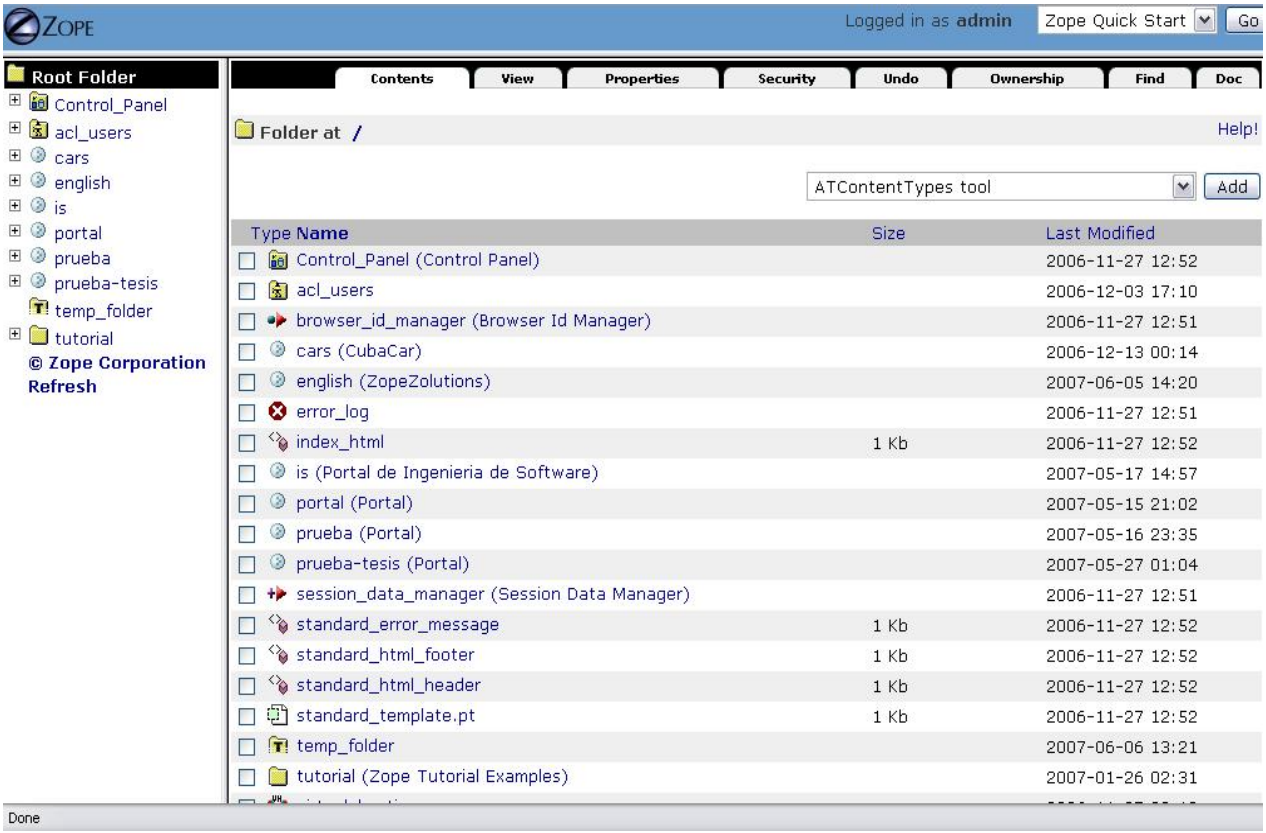

Figura 1: Zope Management Interface

1.2.2.1 Base de datos

Una de las características fundamentales de Zope es que en su arquitectura todo son objetos. Estos se almacenan en la base de datos transaccional orientada a objetos con que cuenta. Puede almacenar contenido, datos personales, plantillas dinámicas HTML, scripts, un motor de búsqueda, y conexiones con bases de datos relacionales. La potencia de este almacenamiento es enorme, sobre todo cuando se modifica dinámicamente y busca datos de los objetos que forma el sitio web.

Álvaro del Castillo en su artículo “Zope, Python y la programación web” señala que la ZODB (Zope Object Data Base) se destaca por ser casi transparente para el programador y los objetos que deban ser persistentes requieren unos cambios mínimos (heredar de una clase Python). Da soporte a las transacciones y ello permite que casi cualquier acción que realicemos en Zope pueda ser deshecha. Destaca también su alto rendimiento, que permite alcanzar un buen rendimiento incluso con bases de

datos de giga-bytes (Del Castillo). La base de datos proporciona que se pueda interactuar mediante adaptadores con otros manejadores de bases de datos como MySQL, PostgreSQL, Oracle, Sybase, entre otros.

1.2.2.2 Arquitectura de Zope

La flexible arquitectura de Zope permite dar soluciones a varios tipos de problemas. Está basada en una serie de mecanismos de intercambio de datos con el servidor Zope y un conjunto de herramientas de apoyo a ese servidor. Lo cierto es que Zope está compuesto por distintos componentes que trabajan conjuntamente manteniendo una arquitectura cliente/servidor muy completa, figura 2.

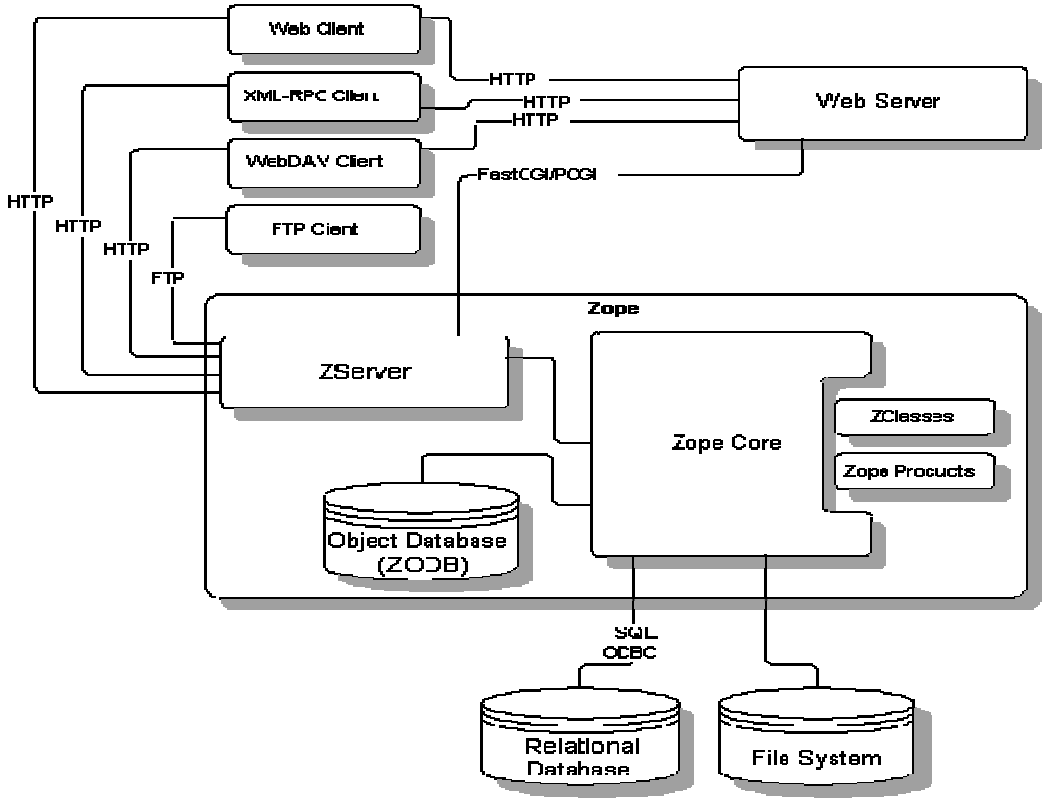

Figura 2: Arquitectura de Zope

- **ZServer**: Zope tiene incluido un servidor web que provee una alta conectividad. Tiene incluido un servidor FTP, Web, XML-RPC, WebDAV para el intercambio de información. Es posible también que interactúe con otros servidores web a través de los protocolos FastCGI, PCGI y HTTP.
- **Web server** (servidor web): Zope trabaja con otros servidores como es el caso de Apache si no se desea usar el que trae por defecto.
- **Zope Core**: Conocido como el ORB, es el que gestiona las peticiones de objetos e interacciona con las clases, la base de datos y los distintos productos. También está compuesto por un motor de búsqueda y una capa para la seguridad, permisos, etc.
- **Object database** (base de datos de objetos): La base de datos de objetos es la ZODB. Esta soporta transacciones, lo que permite que casi cualquier acción en Zope se puede deshacer. Otra característica es que los objetos que deban ser persistentes requieren unos cambios mínimos.
- **Relational database** (base de datos relacional): Zope trabaja con otras bases de datos relacionales como Oracle, PostgreSQL, Sybase, MySQL y otras, en caso de que no se quiera usar la ZODB.
- **File system** (sistema de archivos): Zope puede trabajar con documentos y otros archivos almacenados en su sistema de archivos.
- **ZClasses**: Las ZClasses extienden el núcleo de Zope añadiendo nuevos objetos diseñados desde el ZMI. Lo que permite crear productos online de manera sencilla.
- **Zope products**: Los productos de Zope extienden el núcleo de Zope añadiendo nuevos objetos diseñados con Python en su sistema de archivos.

1.2.2.3 Lenguajes de plantillas

En Zope los usuarios pueden crear su propia presentación, la misma es definida como “(...) *la tarea de definir dinámicamente el diseño de páginas web y otros datos visibles al usuario* (… (Escalona Cuaresma, 2001))” (Del Castillo), debido a que las aplicaciones van a generar páginas web, el lenguaje a utilizar es HTML, que es muy sencillo pero carece de algunas características que permiten darle dinamismo a los contenidos.

Zope ofrece dos posibilidades para la "Presentación": Zope Page Templates (ZPT) y Document Template Markup Language (DTML).

Ambos ZPT y DTML son lenguajes de script que están del lado del servidor, como SSI, PHP, o JSP. Esto significa que los comandos DTML y ZPT son ejecutados por Zope en el servidor, y el resultado de esta ejecución es enviado al navegador del cliente (usuario).

Objetos DTML

DTML es el lenguaje más básico que ofrece Zope para crear plantillas y el que más en desuso está quedando debido a la potencia que ofrece ZPT.
Es recomendado para la realización de plantillas simples, sobre todo en aplicaciones que no requieran grandes dosis de colaboración entre desarrolladores y cuando se quiere dar dinamismo a objetos que no contienen texto HTML o XML, como puede ser CSS, sentencias SQL, etc.

ZPT

Las ZPTs son usualmente usadas para crear HTML dinámico en las páginas web. En ellas se utiliza un lenguaje denominado TAL. Las ZPT crean el dinamismo añadiendo atributos especiales a las marcas ya existentes de HTML.
Cualquier plantilla de una página ZPT es usualmente vista en un navegador con contenido por "defecto" (estático) siempre que estos comandos están incrustados en él, lo cual hace más fácil para ambos programadores y diseñadores ver su trabajo.

1.2.3 Python

Python es un lenguaje de script interpretado, (lo que significa que no se necesita compilar el código fuente para poder ejecutarlo) y orientado a objetos. Implementa estructuras de datos muy avanzadas (lista, tuplas, diccionarios) que se pueden combinar para crear otras estructuras realmente complejas. Permite además mantener de forma sencilla interacción con el sistema operativo y resulta adecuado para manipular archivos de texto, característica esta que hace que muchas distribuciones de GNU/Linux lo utilicen para sus herramientas de configuración.

Características

Python es un lenguaje que está en auge en el mundo, debido a que presenta una serie de características que lo hacen muy atractivo.

- Propósito general:

Se pueden crear todo tipo de programas. No es necesario un lenguaje creado específicamente para la web.

- Multiplataforma

Es posible utilizar Python en plataformas como Unix, Windows y otros. Puede correr en cualquier sistema siempre y cuando exista un intérprete programado para él

- Interpretado.

No se debe compilar el código antes de su ejecución. La compilación que hace el código, se realiza de manera transparente para el programador.

- Interactivo

Python dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias.

- Orientado a Objetos

La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

- Funciones y librerías

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos.

- Sintaxis clara

Python tiene una sintaxis muy visual, gracias a una notación con márgenes de obligado cumplimiento. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

- Libre

Es un lenguaje gratuito. Actualmente, se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

1.2.4 CSS

Las hojas de estilo en cascada (Cascading Sheet Style o CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML, separando el contenido de la presentación. La información de estilo especifica cómo se ha de mostrar una etiqueta determinada. Esta información puede ser adjuntada como un documento separado o en el mismo documento HTML.

1.2.4.1 Ventajas

Las ventajas de utilizar CSS se mencionan a continuación:

- El documento HTML en sí mismo es más claro de entender y se consigue reducir considerablemente su tamaño.
- Pueden usarse con otros lenguajes de programación (como JavaScript) para conseguir efectos dinámicos en las páginas.
- Se pueden especificar hojas de estilo para diferentes navegadores y tipos de medios (impresos, braille, auditivos, etc.).
- Separando contenido de diseño mediante XHTML y CSS se podrán realizar cambios en el sitio en cuestión de segundos, sin importar la cantidad de páginas.
- Una página puede disponer de diferentes hojas de estilo según el dispositivo que la muestre o incluso a elección del usuario.
- Alta velocidad de carga. Usando CSS se aminora la cantidad de código.

1.2.4.2 Navegadores que lo soportan

Esta tecnología es bastante nueva, por lo que no todos los navegadores la soportan, ni implementan las mismas funciones de hojas de estilos, por ejemplo, Microsoft Internet Explorer 3 no soporta todo lo relativo a capas. Los navegadores que no soportan hojas de estilos por lo menos, leerán el código de la página y visualizarán todos los datos aunque descolocados y sin estilo porque no se mostrará ningún posicionamiento.

1.2.5 XHTML

Ante la llegada al mercado de un gran número de dispositivos, XHTML surge como el lenguaje cuyo etiquetado (más estricto que HTML) va a permitir una correcta interpretación de la información independientemente del dispositivo desde el que se accede a la misma.

XHTML, es el acrónimo inglés de eXtensible Hypertext Markup Language (Lenguaje Extensible de Marcado de Hipertexto). Es el lenguaje de marcado que va a sustituir a HTML como un estándar para las páginas web ante la limitante que presenta con el uso de las herramientas basadas en XML. Los tipos de documentos de la familia XHTML están basados y diseñados fundamentalmente para trabajar en conjunto con aplicaciones de usuario basados en XML.

XHTML es una familia de tipos de documentos y módulos actuales y futuros que reproducen, subdividen y extienden HTML 4. Algunas mejoras que pueden ser percibidas para quienes migran a XHTML son:

- Documentos fácilmente visualizados, editados y validados con herramientas XML estándar.
- Los documentos pueden escribirse para que funcionen igual o mejor que como lo hacían antes, tanto en los agentes de usuarios conformes a HTML 4 como en los conformes a XHTML.
- Los documentos pueden usar aplicaciones (scripts y applets) que se basen ya sea en el Modelo del Objeto de Documento (DOM) de HTML o XML.

La familia XHTML es el próximo paso en la evolución de Internet. Adaptándose a XHTML hoy, los desarrolladores de contenido entrarán en el mundo XML con todos sus beneficios relacionados, y al mismo tiempo pueden sentirse seguros de la compatibilidad pasada y futura de sus contenidos.

A medida que la familia XHTML evolucione, los documentos escritos en XHTML estarán más preparados y tendrán mejor rendimiento para interactuar con las actuales herramientas web, que aquellos escritos en HTML.

Navegadores que lo soportan

La mayoría de los navegadores actuales más populares soportan XHTML y aunque es posible encontrar navegadores antiguos que tienen cierta compatibilidad con este lenguaje donde la información se visualiza, pero sin formato, muchos de estos navegadores no son totalmente compatibles con el estándar, lo que provoca que las páginas no siempre se muestren totalmente.

1.2.6 Plone

Plone es un CMS construido sobre la sólida base de Zope, programado en Python y publicado bajo la licencia GNU. La importancia de que sea de código abierto posibilita descargar y utilizar gratuitamente así como también hacer modificaciones, agregándole funcionalidades y productos.

Desde el nacimiento de Plone en 1999, rápidamente se convirtió en uno de los sistemas gestores de contenido de código abierto más populares y poderosos, por ello en el 2004 se creó la Fundación Plone sin fines de lucro, para proteger y promover el uso de Plone, así como asegurar su perfeccionamiento y evolución. Este conocido CMS cuenta con muchísimos seguidores por todo el mundo. Plone agrupa herramientas y utilidades que proveen de un sistema para la gestión de contenido web ideal para proyectos en grupo, comunidades, sitios web, intranets y extranets.

Ventajas de Plone

- Fácil utilización

Es muy fácil de instalar, configurar y poner en marcha. Puede ser utilizado fácilmente tanto por usuarios experimentados como también aquellos usuarios que se están iniciando en el uso de gestores de contenidos web.

- Internacional

Su interfaz se ha traducido a más de 30 idiomas. Detecta la configuración de su navegador, activando el idioma correspondiente.

- Estandarizado

Cumple con estándares de usabilidad y accesibilidad. Las páginas cumplen con las pautas de accesibilidad del W3C para utilizar estándares como XHTML y CSS (lo que permite que la interface sea muy ligera).

- Soporte internacional extendido

Numerosas organizaciones a nivel mundial ofrecen servicios de consultoría, personalización y de soporte en el entorno Plone.

- Extensible

Existen muchos productos que permiten incorporar nuevas funcionalidades y tipos de contenidos. Esta cantidad aumenta debido a que muchos programadores continúan desarrollando productos para Plone. Está construido sobre una arquitectura de componentes reutilizables.

- Tecnológicamente neutro

Es compatible con diferentes plataformas incluyendo GNU/Linux, Windows, Mac OS X, Solaris y BSD. Además puede operar con muchos de los sistemas de bases de datos relacionales, tanto libres y como comerciales; sin embargo por defecto almacena todo su contenido en la base de datos de Zope (ZODB).

- Seguridad

Para manejar la seguridad de los usuarios dispone de roles y de grupos, por lo que los administradores pueden establecer un control, para que los diferentes usuarios accedan a las áreas según su nivel. También es posible manejar localmente la seguridad.

- Está protegido

La Fundación Plone se creó para proteger y promover el uso de Plone. Existen muchas comunidades con seguidores por todo el mundo que contribuyen al desarrollo y evolución de este CMS.

- Todos los elementos del sistema de información pueden tener workflows (flujos de trabajo), incluyendo los usuarios.
- Cuenta con un completo y excelente motor de búsqueda.

1.3 Metodologías para el desarrollo de aplicaciones web

Actualmente lo más importante en el desarrollo de aplicaciones web han sido las herramientas. La facilidad de crear sitios web ha traído como consecuencia, que para el desarrollo de estas aplicaciones no se utilicen métodos y técnicas formales de análisis y diseño. El desarrollo de Internet y de los medios de comunicación ha originado un creciente interés por el desarrollo de propuestas metodológicas que ofrezcan un marco de referencia idóneo al desarrollar portales web.

"*Durante los últimos años han surgido diferentes metodologías que se mueven en entornos concretos dando más importancia a los aspectos más importantes de los sistemas que tratan*" (Escalona, 2001). A raíz de esto varios estudiosos han trabajado y propuesto metodologías que ofrecen modelos y técnicas apropiadas para el desarrollo de aplicaciones web. Dentro de las metodologías que se pueden encontrar es posible citar:

- HDM (Hypermedia Design Method)
- OOHDM (Object-oriented Hypermedia Design Methodology)
- SOHDM (Scenario-based Object-oriented Hypermedia Design Methodology)

- WSDM (Web Site Design Method)
- WAE-Process Conallen (Web Application Extension for UML – Process Conallen)
- OOTD (Object Oriented Text Descomposition)
- OOTDD (Object Oriented Top Down Design)
- OO-Method, con su adaptación a interfaces hipertextuales.
- WebML (Web Modelling Language)
- UML Based Web Engineering (UWE)
- OOWS (Object Oriented Web Solutions)

De todas las metodologías mencionadas sólo se van a tratar cuatro de ellas, debido a que son los métodos más conocidos para la creación de portales web y son las que se emplean como guía para el procedimiento que se propone en este trabajo. También se abunda en el método XP (eXtreme Programming), debido a que es aplicable a cualquier proceso de desarrollo, para un grupo de trabajo relativamente pequeño, proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad. Además se trata la metodología RUP (Rational Unified Process) puesto que es un "*marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación*, (...)" (Jacobson y otros, 2004). No se tratan las metodologías orientadas a hipermedia debido a que están orientadas sobre todo a la optimización del programa desde el punto de vista informático, pero no tanto a la organización de los contenidos, y esto es algo importante para la tecnología Zope/Plone.

1.3.1 WSDM

Web Site Design Method (WSDM) o "Método para diseño de sitios web" (su traducción al español) "*es una propuesta para el desarrollo de sitios web, en la que el sistema se define en base a los grupos de usuarios*" (Appelmans, 2004).

El proceso de desarrollo se divide en cuatro fases: modelo de usuario, diseño conceptual, diseño de la implementación e implementación. En la figura 3 se muestran las fases de este modelo.

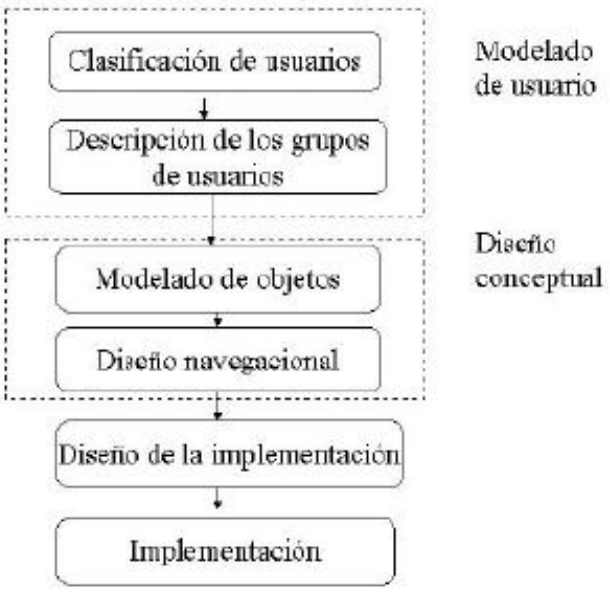

Figura 3: Esquema de fases de WSDM

La fase más importante es la primera, puesto que se intenta detectar los perfiles de usuarios para los cuales se construye la aplicación. Para ello, se deben realizar dos tareas:

1. Clasificación de usuarios: se deben identificar y clasificar a los usuarios que van a hacer uso del sistema.
2. Descripción de los grupos de usuarios: se describen con más detalles los grupos de usuarios detectados en la etapa anterior. Para esto hay que elaborar un diccionario de datos, en el que indican los requisitos de almacenamiento de información, requisitos funcionales y de seguridad para cada grupo de usuarios.

El resto de las fases del proceso de WSDM se hace en base a la clasificación de usuarios que se realiza en la primera etapa.

La fase de diseño conceptual se divide en otras dos subfases:

1. Modelado de objetos: se modela la información y los requerimientos funcionales de las diferentes clases.
2. Diseño navegacional: se modela la estructura del sitio web.

La fase de diseño de la implementación se divide en tres subfases:

1. Diseño de página: se agrupan la información en páginas, siguiendo el diseño navegacional realizado.
2. Diseño de presentación: se especifica la presentación.
3. Diseño de datos lógicos: se diseña la base de datos.

En la fase de implementación se implementa el sitio web.

El primer paso que plantea el método WSDM es definir “the mission statement” (la declaración de la misión) del sitio web. Esta declaración debe responder las siguientes preguntas:

¿Cuál es el objetivo del sitio web?

¿Cuál es el tema?

¿Quién es el beneficiado?

Sin la declaración de la misión no existirá una base apropiada para tomar decisiones o evaluar la efectividad del sitio y los usuarios no tendrán ninguna idea del objetivo del sitio web.

1.3.2 WebML

Web Modelling Language (WebML) es una notación visual para el diseño de aplicaciones web en la que se especifica el contenido, composición y las características de navegación para las aplicaciones de hipertexto utilizando UML. Provee especificaciones gráficas formales para un proceso de diseño completo que puede ser asistido por herramientas de diseño visuales.

El ciclo de vida que propone este modelo comienza con el modelado conceptual del sistema en el que mediante algún lenguaje de modelado como UML (WebML no exige ninguno en concreto) se representa la estructura estática del sistema. Luego, se realiza el modelo de hipertexto en el que se describen uno o más hipertextos que pueden ser publicados en el sitio web. Cada uno de estos hipertextos define una vista del sitio. La descripción de los hipertextos se realiza mediante dos modelos:

1. Modelo de composición, que define las páginas que componen el sistema.
2. Modelo de navegación, que describe cómo se podrá navegar a través de ellas.

En el siguiente paso del proceso se describe el modelo de presentación, que define la apariencia física de las páginas. Y por último, el modelo de personalización define cómo debe adaptarse el sistema a los diferentes roles de usuario. En la figura 4 se muestra el ciclo de vida de este modelo.

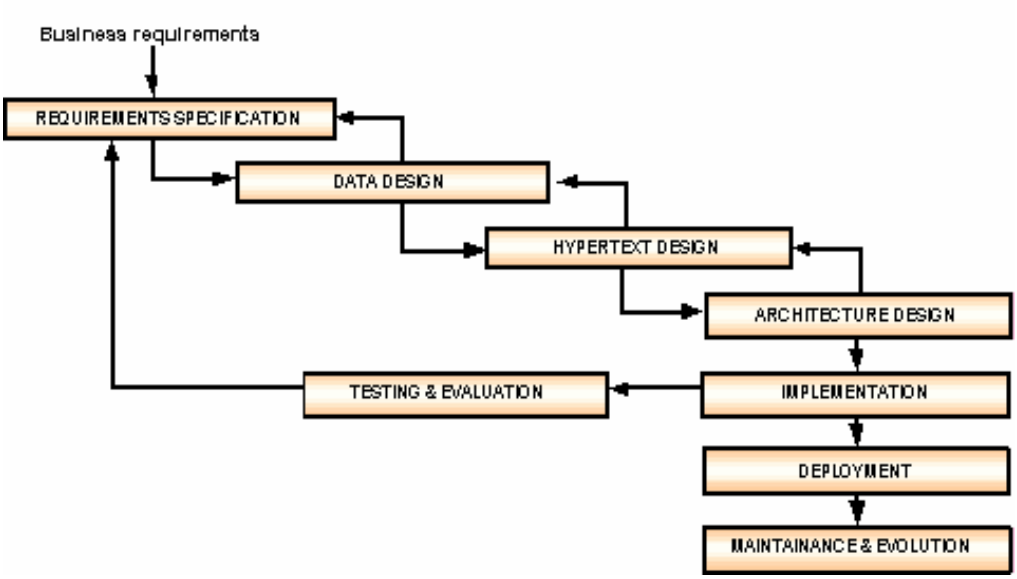

Figura 4: Esquema del ciclo de vida de WebML

En la primera etapa, es donde se identifican los diferentes usuarios y sus necesidades de personalización, además se capturan los requisitos funcionales y no funcionales.

WebML se enfoca en el diseño de la interfaz. Para esto provee una serie de estereotipos que pueden ser implementados usando XML. Sin duda WebML es una buena opción que permite a los modeladores expresar de forma conceptual los datos y la navegación de una aplicación Web utilizando diagramas Entidad-Relación.

Una de las facilidades de este modelo es que disponen de una herramienta CASE llamada WebRatio, que genera automáticamente aplicaciones completamente funcionales de los diagramas WebML. La herramienta es libre, está disponible para usos no comerciales.

1.3.3 UML Based Web Engineering

UML-Based Web Engineering (UWE) es una propuesta metodológica basada en el Proceso Unificado y UML para el desarrollo de aplicaciones web. UWE cubre todo el ciclo de vida de este tipo de aplicaciones, y consta de un proceso de desarrollo de tres fases, que es mostrado en la figura 5: captura de requisitos, análisis y diseño, e implementación.

Figura 5: Proceso de desarrollo de UWE

UWE propone las siguientes tareas a realizar:

Fase de requisitos: Comenzar con "*la identificación de los usuarios y la elicitación de los requisitos*" (Escalona y Koch). El resultado final de la captura de requisitos es un modelo de casos de uso acompañado de documentación que describe los usuarios del sistema, las reglas de adaptación, los casos de uso y la interfaz.

Fase de análisis y diseño: Distingue entre diseño conceptual, de modelo de usuario, de navegación, de presentación, de adaptación, de la arquitectura, en el diseño detallado de las clases y en la definición de los subsistemas e interfaces.

Fase de implementación: Implementación de la arquitectura, implementación de la estructura del hiperespacio, implementación del modelo de usuario, implementación de la interfaz de usuario, implementación de los mecanismos adaptativos y las tareas referentes a la integración de todas estas implementaciones.

UWE considera los requisitos de navegación como un tipo de requisito funcional y, aunque no propone técnicas específicas para el tratamiento de los mismos, principalmente se basa en los casos de uso, los que separa con idea de identificar mejor los aspectos que influirán en el modelo navegacional que se realiza en la fase de análisis y diseño.

1.3.4 OOWS

Object Oriented Web Solutions o "Método de Producción de Soluciones Web Orientada a Objeto" es una extensión del método OO-Method con capacidades navegacionales y de presentación.

Siguiendo la aproximación OO-Method, la especificación de los requisitos de usuario debe realizarse en la fase de modelado conceptual. De acuerdo a lo que plantea María J. Escalona en su tesis "Metodologías para el desarrollo de sistemas de información global: análisis comparativo y propuesta", para modelar la navegación asociada al sistema deseado se propone un proceso de desarrollo de soluciones web con dos pasos principales: Especificación del problema y desarrollo de la solución (Escalona, 2001).

1. Especificación del problema: se deben capturar las peculiaridades y el comportamiento que debe ofrecer el sistema para satisfacer los requisitos de usuario identificados. En esta fase se realiza un estudio de los tipos de usuarios que pueden interactuar con el sistema, indicando qué visibilidad sobre el sistema tendrán (qué atributos y qué operaciones podrán ver y/o activar), cómo se podrán conectar (requerirán o no identificación), y se organizarán en jerarquías de especialización.

La fase de modelado conceptual se considera como una única fase. Usando lo que podría llamarse una aproximación de modelado conceptual OO, se introduce la expresividad necesaria en el modelo para especificar correctamente los requisitos navegacionales y las características de presentación. Toda esta información es utilizada para proporcionar una guía metodológica precisa para ir del espacio del problema (modelado conceptual) al espacio de la solución (representado por el producto software final), soportado por un proceso de generación automática de código.

En el modelado conceptual, las abstracciones que se derivan del problema son especificadas en términos de clases y de su estructura, comportamiento y funcionalidad, construyendo los siguientes modelos:

- Modelo de objetos define la estructura y las relaciones estáticas entre clases identificadas en el dominio del problema.

- Modelo dinámico se describen las posibles secuencias de servicios y los aspectos relacionados con la interacción entre objetos.
- Modelo funcional captura la semántica asociada a los cambios de estado entre los objetos motivados por la ocurrencia de eventos o servicios.
- Modelo de navegación define cómo se le proporcionará a cada usuario del sistema el acceso a la información y la funcionalidad que le es relevante para llevar a cabo su tarea dentro del sistema y qué secuencias de caminos deberán seguir para conseguirlo. Está compuesto por un conjunto de mapas de navegación que representan y estructuran la visión global del sistema para cada tipo de usuario, definiendo su navegación permitida.
- Modelo de presentación captura los requisitos básicos de presentación de información, orientado hacia ambientes web. Está fuertemente basado en el modelo de navegación y permite definir, de una manera abstracta, la estructura lógica de presentación de los objetos navegacionales en la interfaz de usuario.

2. Desarrollo de la solución*:* se propone una estrategia de generación de código basada en componentes para integrar la solución propuesta en ambientes web. En esta etapa se obtiene una aplicación web, con una funcionalidad equivalente a la especificación inicial según una visión operativa. Esta fase "*se desarrolla de manera totalmente automática por un compilador de modelos conceptuales que, en función de unos patrones arquitectónicos y dependiendo de la plataforma destino, construye un sistema software que recoge los requisitos de la aplicación modelada, siguiendo uno de los principales objetivos de la metodología propuesta por OO-Method*" (Fons, y otros).

1.3.5 Rational Unified Process

RUP o "Proceso Unificado de Desarrollo" es un proceso iterativo e incremental porque propone que cada fase desarrolle en iteraciones; con esto se logra reducir los riesgos del proyecto y tener un subsistema ejecutable tempranamente. El proceso unificado está dirigido por casos de uso, quiere decir que "(…) *el proceso de desarrollo sigue un hilo y avanzan hacia una serie de flujos de trabajo que parten de los casos de uso, estos se especifican, diseñan y los casos finales son la fuente a partir de los cuales (…) se construyen los casos de prueba*". (Jacobson y otros, 2004) Está centrado en la arquitectura porque muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar

de acuerdo. Describe además, los cimientos del sistema que son como base para comprenderlo, desarrollarlo y producirlo económicamente. A medida que los casos de uso se especifican y maduran se conoce algo más acerca de la arquitectura, este proceso continúa hasta que se distinga que la arquitectura es estable.

RUP divide el proceso de desarrollo en ciclos, cada ciclo se divide en cuatro fases: inicio, elaboración, construcción y transición; cada fase concluye con un hito. Los principales elementos del proceso unificado son: trabajadores (quién), actividades (cómo), artefactos (qué) y flujo de actividades (cuándo). En RUP se han agrupado las actividades definiéndose nueve flujos de trabajo, los primeros seis son conocidos como los flujos de ingeniería y los tres últimos como de apoyo. En la figura 6 mostrada a continuación se muestran los nueve flujos de trabajo y las cuatro fases que define RUP.

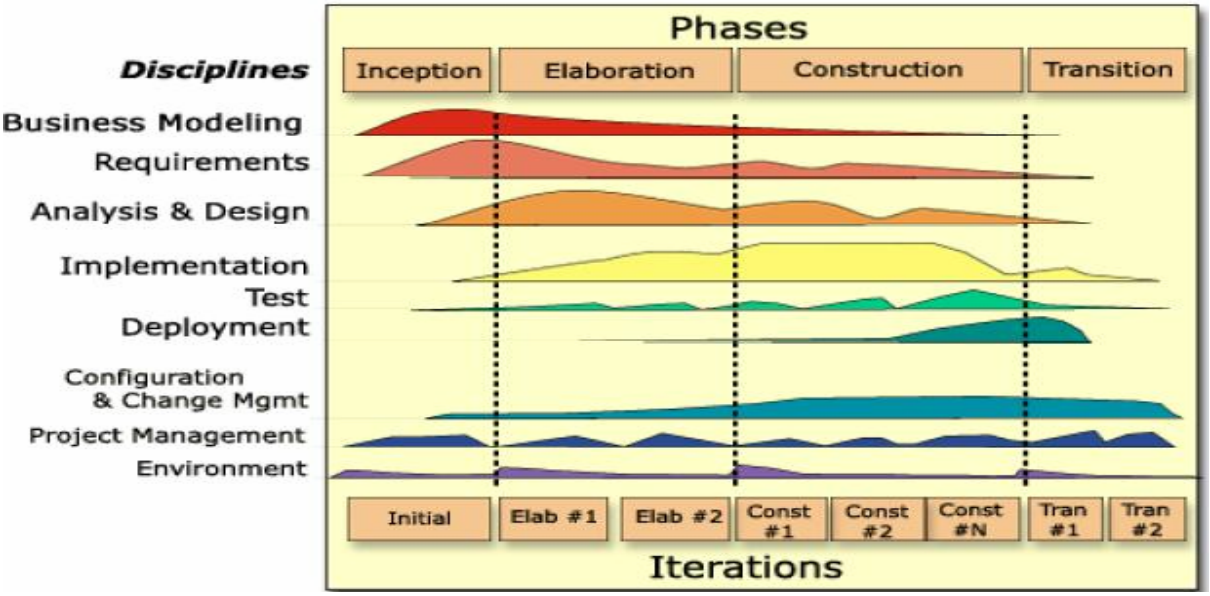

Figura 6: Fases y flujos de trabajo de RUP

Características de los flujos de trabajo de ingeniería

- Modelamiento del negocio: describe los procesos del negocio indicando quiénes participan y las actividades que requieren automatización.
- Requerimientos: define lo que el sistema debe hacer, para lo que se identifican las funcionalidades requeridas y las restricciones que se imponen.

- Análisis y diseño: describe cómo el sistema será realizado a partir de los requerimientos, indica con precisión lo que se debe programar.
- Implementación: define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, y la estructura de capas de la aplicación.
- Pruebas: busca los defectos a lo largo del ciclo de vida, las pruebas pueden ser de integración, de sistema, de regresión y de unidad.
- Instalación: produce la publicación (release) del producto y realiza actividades para entregar el software a los usuarios finales.

1.4.6 eXtreme Programming

XP "eXtreme Programming" o "Programación Extrema" (su traducción al español) es una de las tantas metodologías que están dentro del grupo de "metodologías ágiles", de todas es la que ha recibido más atención.

XP está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en una serie de valores que se aplican una y otra vez en las distintas prácticas utilizadas en la programación extrema: realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

"*Muchas de las prácticas que propone XP son técnicas antiguas, tratadas y probadas, aunque a menudo olvidadas por muchos, incluyendo la mayoría de los procesos planeados. Además de resucitar estas técnicas, la XP las teje en un todo sinérgico dónde cada una refuerza a las demás*" (Fowler, 2000).

Fases y prácticas XP

De acuerdo con Don Wells en su portal llamado "The Rules and Practices of Extreme Programming" para cada regla (como él le llama) de este método se utilizan diferentes prácticas, como se ve a continuación (Wells, 2001):

- Planificación del proyecto

  - *Historias de usuario*. El primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: constan de tres o cuatro líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre una y tres semanas.

  - *Release planning*. Después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones, en inglés “release plan”, donde se indican las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Un “release plan” es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa.

  - *Iteraciones*. Todo proyecto que siga la metodología XP se ha de dividir en iteraciones de aproximadamente tres semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las historias de usuario definidas en el “release planning” que serán implementadas. También se seleccionan las historias de usuario que no pasaron el test de aceptación que se realizó al terminar la iteración anterior. Estas historias de usuario son divididas en tareas de entre uno y tres días de duración que se asignarán a los programadores.

  - *Velocidad del proyecto*. Medida que representa la rapidez con la que se desarrolla el proyecto. Para estimarla hay que contar el número de historias de usuario que se pueden implementar en una iteración, de esta forma se sabrá la cantidad de historias que se pueden desarrollar en las distintas iteraciones. Usando la velocidad del proyecto se controla que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración.

- *Reuniones diarias*. Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

- Diseño
  - *Diseños simples.* Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
  - *Metáfora*. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema.
  - *Funcionalidad extra.* Nunca se debe añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada.
  - Spike solutions. Programa muy simple para explorar soluciones potenciales y para arreglar las respuestas a los problemas técnicos o del diseño que sean resistentes.
  - *Refactorizar*. Es una técnica para reestructurar el código, alterando su estructura interna sin alterar el comportamiento externo con el objetivo de remover duplicación de código. *"Su corazón es una serie de pequeños comportamientos que preservan las transformaciones, cada transformación (llamada refactorización) hace poco, pero una secuencia de transformaciones puede producir una reestructuración significativa"* (Fowler).
  - *Tarjetas C.R.C*. El uso de las tarjetas C.R.C (Class, Responsabilities and Collaboration) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedimental clásica. Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

- Codificación
  - *Disponibilidad del cliente.* El cliente real(es el que usará el sistema) tiene que estar con el equipo de trabajo, debe responder preguntas, resolver disputas, establecer prioridades, discutir mejoras. La comunicación oral es más efectiva que la escrita.

- *Estándares de programación*. XP plantea que es indispensable que se sigan ciertos estándares de programación para mantener un mismo estilo de código: homogéneo y legible. Son fundamentales cuando los programadores cambian de pareja o hacen refactorización del código de otros.
- *Programación en parejas*. Todo el código se escribe en parejas, se produce código de mayor calidad, se extiende el conocimiento y se realiza el trabajo de una persona en casi la mitad del tiempo y mejor.
- *Pruebas*. La producción de código está dirigida por las pruebas unitarias. Estas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.
- *Integración continua*. Cada pieza de código es integrada en el sistema una vez que esté lista y se prueba. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
- *Propiedad colectiva del código*. Cualquiera puede modificar el código en cualquier momento. Se evitan cuellos de botella en la codificación, todos asumen las responsabilidades sobre el conjunto del sistema.

- Prueba
  - *Pruebas*. La producción de código está dirigida por las pruebas unitarias. Estas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema
  - *Pruebas de aceptación*. Son un sistema de prueba de caja negra. Cada prueba de aceptación representa algún resultado esperado del sistema. Los clientes son los responsables de verificar que las pruebas estén correctas para decidir cuál prueba fallida tiene la prioridad más alta.

Otras prácticas que propone XP, según los autores Canós, Letelier & Penadés serían (Canós, y otros):

- *El juego de la planificación*. Comunicación frecuente entre cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
- *Entregas pequeñas*. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Una entrega no debería tardar más de tres meses.

- *40 horas por semana*. Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas debido a que el exceso de trabajo puede convertirse en un serio problema.

En este capítulo se realizó un estudio de las tendencias actuales y estándares que componen a la tecnología Zope/Plone y que son utilizadas en el desarrollo de la propuesta de procedimiento. De ellas se estudiaron características y funcionalidades con el fin de comprenderlas mejor. Aprovechando las ventajas que ofrecen son empleados: Python porque es el lenguaje en el que está escrito Zope, CSS para separar el contenido del diseño y XHTML para escribir las páginas web. También fueron estudiadas las características, funcionalidades y ventajas de Plone para la creación, mantenimiento y actualización de los sitios y de Zope como el servidor de aplicaciones web sobre el cual está montado Plone. Se estudiaron y analizaron además diversas metodologías que alumbrarán el camino para el procedimiento que se realizará en el próximo capítulo. Una vez conocidas las herramientas y los conceptos se puede empezar a desarrollar la propuesta de procedimiento.

Capítulo 2 Propuesta de procedimiento para la creación de portales web utilizando la tecnología Zope/Plone

En el presente capítulo se expone la propuesta de procedimiento para la creación de portales web utilizando la tecnología Zope/Plone. Primeramente se realiza una crítica a las metodologías que fueron abundadas en el capítulo anterior. Se explica la filosofía de trabajo implantada en el procedimiento, y dentro de ella se mencionan las características que tiene así como recomendaciones para el equipo de desarrollo. Se puntualizan además cada flujo de trabajo con que va a constar el mismo y dentro de él se precisan los actores, artefactos y actividades que lo componen.

2.1 Análisis de las metodologías existentes.

Si se analiza la utilización de las fases clásicas de todo proceso de desarrollo: especificación de requisitos, análisis, diseño, implementación y prueba en los métodos analizados, se verá cuál de ellas son tratadas en cada uno y cuál puede ser útil para la realización de la propuesta de procedimiento.

La fase de especificación de requisitos es tratada en todas las metodologías que fueron analizadas, aunque en cada una de forma diferente. Dentro de las técnicas que son utilizadas para la definición de requisitos se encuentran: la ambigua técnica de lenguaje natural que es usada por WSDM; la de casos de usos que son utilizados por WebML, OOWS, RUP y UWE este último además utiliza las técnicas de escenarios y glosarios, y XP define historias de usuario que básicamente tienen la misma finalidad que los casos de uso pero con algunas diferencias. De lo anteriormente visto es posible inferir que en la fase especificación de requisitos existe una tendencia a usar la técnica de casos de uso.

La fase de análisis es abordada en casi todas las metodologías estudiadas, de ellas WSDM centra en el análisis del estudio de los grupos de usuarios, mientras que el resto se centran en realizar modelos. En UWE se unen las fases de análisis y la de diseño en una sola, análisis-diseño, dentro de ella se realizan el modelo conceptual, de usuario, de navegación, de presentación, de adaptación y de la arquitectura, que son los que corresponden a la fase tratada y el resto de los modelos que se construyen, pertenece al diseño. WebML permite el desarrollo de modelos conceptual y el de hipertexto. OOWS dentro del

modelado conceptual se construyen los modelos de objeto, dinámico, funcional, navegación y presentación. RUP al igual que UWE tiene una fase de análisis-diseño, el resultado final del flujo de trabajo del análisis es el modelo del análisis que es un modelo de objetos conceptual, el modelo incluye los paquetes, clases y realizaciones de casos de uso del análisis, y la vista de la arquitectura del modelo de análisis. Con lo mencionado anteriormente se muestra que la mayoría de los métodos convergen en el empleo de un modelo conceptual en el análisis.

La fase de diseño es tratada en todos los métodos estudiados, pero de forma diferente. De las prácticas y tendencias que estas metodologías utilizan para elaborar este flujo las que más se adecuan a este trabajo son las utilizadas por WSDM, RUP y XP. En la metodología WSDM se hace énfasis en la importancia de la modelación de la información y los requerimientos funcionales, así como la modelación estructural del sitio web. En RUP se expone un concepto importante y es que el diseño describe cómo el sistema será realizado a partir de los requerimientos, indica con precisión lo que se debe programar, se basa en los artefactos y las actividades para dar cumplimiento a los requisitos obtenidos. En XP se tienen en cuenta varias prácticas de las cuales las más importantes son: diseños simples, porque es posible, mediante esta práctica, conseguir diseños sencillos, fáciles de entender; funcionalidad extra, no se debe agregar una funcionalidad extra aunque se piense que se puede utilizar en el futuro, pues puede llegar a resultar un desperdicio de tiempo y recursos; y refactorizar, esta práctica es importante, porque su objetivo es eliminar la duplicación de código, esto se logra alterando la estructura interna del mismo sin alterar el comportamiento externo.

La fase de implementación es abordada en las metodologías WSDM, OOWS, RUP y XP. En WSDM es tratada esta fase como simplemente implementar el sitio, lo cual no es la mejor solución para emplear en el proyecto “Informatización de la prensa”; en OOWS se propone la estrategia de generar el código basándose en componentes, se desarrolla de manera totalmente automática. En la fase de implementación de la metodología RUP se define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes, y la estructura de capas de la aplicación; en XP se promueve la simplicidad de las soluciones implementadas y aplica varias prácticas, como son: disponibilidad del cliente, expresa que el cliente debe formar parte del equipo de trabajo, debe responder preguntas, establecer prioridades; programación en parejas, se produce un código de mayor calidad y se realiza el trabajo en casi la mitad del tiempo y mejor; pruebas, deben ser ejecutadas constantemente ante cada modificación del sistema, logrando una corrección de errores en

tiempo real; integración continua, cada pieza de código debe ser integrada en el sistema cada vez que esté lista y se prueba, así el sistema puede ser integrado y construido varias veces en un mismo día; esta última es las que más se adecua a este trabajo por lo anteriormente explicado.
La fase de prueba está presente en WebML, aunque no hay mucha información respecto al tipo de pruebas que realizan, Baresi, Fraternali, Tisi & Morasca proponen un acercamiento a la caja-negra basado en la gramática formal de WebML (Baresi, y otros). En XP se realizan pruebas unitarias y de aceptación. Mientras que en RUP se realizan pruebas de unidad, estas son: pruebas de especificación o caja negra y prueba de estructura o caja blanca; pruebas de integración, de sistema, de regresión y de aceptación.

A partir del estudio realizado se muestra que RUP es una metodología en la que se documenta muy bien todo el trabajo que se va realizando y se ajusta a todo tipo de proyecto, tanto web como aplicación de escritorio. Pero como se deben realizar tantas actividades en cada flujo de trabajo y tiene tantos artefactos y trabajadores entonces existirá demora en el tiempo de entrega, lo cual no cumple con la necesidad de rapidez que se requiere en el proyecto; además los requisitos deben estar bien definidos desde el inicio.
Debido a la negativa de la utilización de RUP, se pensó que XP podría ser de gran utilidad porque es una metodología ágil, para grupos pequeños de trabajo en los que se mantiene una constante comunicación con los clientes y está especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, pero su dificultad radica en que cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves) y además nada se documenta debidamente durante el desarrollo del sistema; estos son los motivos por los que no es utilizada completamente. Al mismo tiempo las metodologías UWE, WebML, WSDM y OOWS podrían utilizarse pero son demasiado extensas, algunas no tienen el flujo de trabajo de prueba, el cual es importante realizar al final de cada versión de los sitios web, y existe muy poca documentación respecto a ellas, la información que se puede encontrar son criterios, valoraciones y algunas características que brindan otros autores de las mismas.
Por todo lo dicho antes es que se decide crear una solución, que para este caso sería un procedimiento que tuviera la agilidad de XP, la documentación, algunos artefactos y actividades de RUP, así como algunos principios y conceptos que se abordan en las metodologías usadas para el diseño y la creación de sitios web.

2.2 Bases de la propuesta

Tomando las buenas ideas de las propuestas que se han analizando, se define un procedimiento para portales web utilizando la tecnología Zope/Plone, en el que se cubre por completo el ciclo de vida del proyecto y en el que, para cada fase, el grupo de trabajo de acuerdo al rol asignado tiene una guía de cómo hacer las cosas, qué presentar al cliente y qué obtener después de cada flujo de trabajo.

La filosofía de trabajo que se desea implantar es muy simple, se pretende crear portales web de gran calidad, soluciones muy eficientes que satisfagan la necesidad de los clientes en el tiempo previsto y que permitan la incorporación de nuevas características y funcionalidades con facilidad. Para lograr esto, se necesita una retroalimentación continua del cliente y el equipo de desarrollo.

Analizadas las diferentes propuestas metodológicas, se han adquirido diversas ideas generales de todas ellas que serán asumidas en la propuesta. El éxito que estos modelos han tenido, ampara la decisión de tomarlas como adecuadas para la propuesta que se realiza. Por ello el procedimiento cumple las siguientes características:

- Cubrir todo el ciclo de vida del proyecto.
- Indicar qué hacer en cada momento del ciclo de vida.
- Orientar en cada fase qué obtener.
- Sencillez y claridad en el código y el diseño.

Además propone las siguientes recomendaciones para el equipo de trabajo:

1. El equipo de trabajo debe ser pequeño, tener una buena formación y deseos de aprender.
2. Debe existir una constante comunicación y retroalimentación entre clientes y equipo de desarrollo.

A partir de estas características, en el próximo epígrafe se presenta una visión global de todo el procedimiento.

2.3 Propuesta de procedimiento

La propuesta de procedimiento consta de un ciclo de vida que se divide en cinco flujos de trabajo: especificación de requisitos, análisis, diseño, implementación y prueba. Estos flujos de trabajo se realizan de forma consecutiva.

Para un mejor esclarecimiento y comprensión de cada flujo de trabajo se emplean términos como trabajadores, artefactos y actividades.

Trabajadores: es toda aquella persona que interviene en el desarrollo de un portal. "U*n tipo de trabajador es un papel que un individuo puede desempeñar en el desarrollo de un software* (…)" (Jacobson y otros, 2004). Un trabajador puede representar un grupo de personas que trabajan juntas como por ejemplo un trabajador arquitecto puede ser un grupo de arquitectura. Una persona puede ser varios trabajadores durante el ciclo de vida de un proyecto.

Artefactos*:* término generalizado dado a cualquier tipo de información que es creada o utilizada por los trabajadores en el desarrollo del portal.

Actividades: son todas aquellas actividades que de forma organizada se realizan para darle cumplimiento a los flujos de trabajo. Las actividades están divididas en una o varias tareas.

2.3.1 Especificación de los requisitos

Un requisito es la representación documentada de una condición o capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal; este concepto es uno de los tantos que existen sobre los requisitos en el mundo y que aparece en el glosario de términos de la IEEE.

Se debe tener en cuenta a la hora de identificar y definir requisitos que la descripción de los mismos debe ser lo más precisa posible, debe garantizar que la definición de estos en un documento tiene que ser conocida y aprobada por los clientes y el equipo de desarrollo y además debe expresar la conformidad de ambas partes con esa descripción de la solución.

Trabajadores

Jefe de proyecto

Es el responsable de organizar y guiar las reuniones que se realizarán con los clientes.

Analista

Es el encargado de realizar la captura de requisitos, modela estos en casos de uso y encuentra los actores del sistema asegurando que el catálogo de requisitos y el modelo de casos de uso estén completos.

Artefactos

Catálogo de requisitos

Para la captura de los requisitos el equipo de trabajo llevará a cabo varias entrevistas y reuniones con el cliente en la cual se identificarán sus necesidades, luego de que estén definidas, se especifican en un documento llamado catálogo de requisitos, el mismo debe englobar:

- Los requisitos funcionales
- Los requisitos no funcionales
- Modelo de casos de uso
- La descripción de los actores del sistema
- Descripción de los casos de usos
- Glosario: El glosario se empleará para definir términos que les son comunes a los analistas y otros desarrolladores para describir el sistema, y conceptos para reducir al máximo el riesgo de confusiones que se pudieran crear durante el ciclo de desarrollo del portal y para lograr mayor comprensión entre los integrantes del equipo de desarrollo.

Actividades del flujo de trabajo de especificación de requisitos

Esta fase para su mejor análisis y utilización está dividida en actividades y estas, a su vez, en tareas.

Actividad 1: Adquirir información del entorno.

Tarea 1.1: Obtener información del entorno de trabajo.

Tarea 1.2: Preparar y realizar entrevistas y reuniones.

Actividad 2: Identificar y definir los requisitos.

Tarea 2.1: Definir los requisitos funcionales.

Tarea 2.2: Identificación y definición de los requisitos no funcionales.

Actividad 3: Realizar modelo de casos de uso.

Actividad 4: Identificar y definir los actores del sistema.

Tarea 4.1: Definir los actores del sistema.

Tarea 4.2: Asignar permisos a los actores del sistema.

Actividad 5: Describir los casos de uso.

Actividad 1: Adquirir información del entorno

La primera actividad que hay que realizar es estudiar y llegar a conocer el entorno de trabajo para el cual se va a trabajar, para esto se deben estudiar folletos, el funcionamiento y características del portal que hasta este ese momento existe; así como estableciendo entrevistas y reuniones que le permita al equipo de desarrollo obtener toda la información que necesita para identificar y definir correctamente los actores y requisitos del nuevo portal.

Actividad 2: Identificar y definir los requisitos

Para dar cumplimiento a esta actividad se debe de conocer bien el entorno.

Los requerimientos funcionales detallan las funciones que el sistema será capaz de realizar. Para definir los requisitos se emplean los casos de uso, los cuales se muestran en la cuarta y sexta actividades: realizar modelo de casos de uso y describir los casos de uso.

Los requerimientos no funcionales tienen que ver con las cualidades que el producto debe tener, cualidades que son las que lo limitan. Debe pensarse en estas cualidades como las características que hacen al producto atractivo, usable, rápido, confiable. Algunos de estos requisitos serán de usabilidad, confiabilidad, hardware, rendimiento, etc.

Actividad 3: Realizar modelo de casos de uso

Ahora lo que se debe realizar es el modelo de casos de usos de acuerdo a los requisitos funcionales que se recogieron. Se deben relacionar los actores y los casos de uso que estos realizan, además de mostrar las relaciones que puedan existir dentro de varios casos de uso y entre los actores. Para representar este modelo se emplearán los diagramas que propone UML.

Actividad 4: Identificar y definir los actores del sistema

Un actor es una abstracción de una persona, hardware o sistemas que interactúe con el sistema. "*Varios usuarios pueden actuar como diferentes ocurrencias del mismo actor*" (Jacobson y otros, 2004).

En el sistema aparecerán varios actores, cada uno va a definir un rol que los usuarios asumen cuando interactúan con el sistema, por lo que se hace necesario definir qué roles son los se utilizan. Luego que estén bien definidos todos los actores del sistema se procede a asignar los permisos de acuerdo al rol que

desempeña en el portal, se debe tener en cuenta que en ocasiones un mismo actor puede tomar diferentes roles dentro del sistema.

Los roles básicos que serán empleados son los siguientes:

Miembros: son todos los usuarios que se han unido, es decir que tienen cuenta de usuario en el portal.

Revisor: tiene muchos más permisos que los miembros pero menos que los administradores. Son usuarios que pueden editar o revisar el contenido entrado por los miembros. Ellos pueden cambiar la configuración de un sitio o alterar una cuenta de usuario.

Administrador: pueden hacer cualquier cosa como eliminar o editar contenido, eliminar usuarios, alterar la configuración e incluso eliminar sitios web.

Usuario anónimo: usuario que no se ha autenticado.

En dependencia de los requisitos solicitados por el cliente se adicionarán a los portales nuevos roles y a los cuales le serán asignados diversos permisos.

Para la identificación y descripción de los actores del sistema se empleará el patrón que se muestra en la tabla 1 que aparece a continuación.

Actores del sistema	**Descripción**
(1)	(2)

Tabla 1: Patrón para la descripción de los actores del sistema

(1) Nombre del actor

(2) Descripción de la función que realizan en el sistema, aquí también se incluyen los permisos que le son asignados de acuerdo a su rol.

Actividad 5: Describir los casos de uso

Al describir los casos de uso y debido a la necesidad de que su definición sea completa se utilizará el patrón que muestra la tabla 2.

Caso de uso	
(1)	(2)
Propósito	(3)
Actores: (4)	
Resumen: (5)	
Referencias	(6)
Precondiciones	(7)
Poscondiciones	(8)
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
(9)	(10)
Flujo alternativo (11)	
Acción del actor	**Respuesta del sistema**
(12)	(13)

Tabla 2: Patrón para la descripción de los casos de uso

(1) Código del caso de uso.

(2) Nombre del caso de uso.

(3) Objetivo del caso de uso.

(4) Actor o actores del caso de uso.

(5) Descripción del caso de uso.

(6) Requerimientos funcionales o casos de uso con los que se relacione el caso de uso, ya sea porque es un requerimiento que le da origen, o por ser un caso de uso que se incluye o extiende.

(7) Condiciones que se deben cumplir para que se realice el caso de uso.

(8) Condiciones que se cumplen luego que se realice el caso de uso.

(9) Listado de acciones que realiza el actor.

(10) Listado de respuestas del sistema ante las acciones del actor

(11) Nombre o número del flujo alternativo

(12) Listado de acciones que realiza el actor.

(13) Listado de respuestas del sistema ante las acciones del actor.

2.3.2 Análisis

Los objetivos del análisis son conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de entender y mantener, y que ayude a estructurar el sistema. En esta fase se puede razonar más sobre los aspectos internos del sistema. En el análisis se deberá conseguir el modelo conceptual, para lograr esto definen las plantillas que se utilizarán y las entidades o contenidos, así como el flujo de trabajo que sigue cada uno.

Trabajadores

Arquitecto de información

La función del arquitecto de información es realizar la arquitectura del sistema. Este modelo se crea conociendo previamente y según los requerimientos de diseño especificados por los clientes, las vistas que van a existir y/o las que se pudieran crear, y los contenidos que participan y que se van a relacionar directamente con las diferentes plantillas. Tiene que además definir los campos y el flujo de trabajo de contenidos nuevos que se pudieran crear.

Artefactos

Documento de la arquitectura

El documento de la arquitectura recogerá el modelo conceptual del análisis. El modelo conceptual permite modelar y describir la información que maneja el sistema además de su estructura. El modelo va a constar de un modelo que mediante la notación UML represente la estructura conceptual de la aplicación.

Actividades del flujo de trabajo de análisis

Actividad 1: Construir el modelo conceptual del análisis.

Tarea 1.1: Identificar y definir las plantillas.

Tarea 1.2 Definir las entidades.

Tarea 1.3: Definir flujo de trabajo de los contenidos.

Tarea 1.4: Creación del modelo.

Actividad 1: Construir el modelo conceptual del análisis

Las clases del análisis según RUP encajan en uno de tres estereotipos básicos: de interfaz, entidad o control. En este trabajo sólo se va a utilizar la clase entidad debido a que las clases control están implementadas en el propio Plone y las clases interfaz no van a existir como clases propiamente porque en Plone todos los usuarios se encuentran la misma interfaz, lo único que cambian son las plantillas o vistas de acuerdo al rol, por tal motivo para este procedimiento las clases interfaz se convertirán en plantillas (son vistas de las entidades).

Clase entidad: "*se utilizan para modelar información que posee larga vida y que es con frecuencia persistente*" (Jacobson y otros, 2004). Se usará la clase entidad puesto que los portales van a constar con una serie de contenidos (pueden llamarse también recursos de información documental), es decir, todos los contenidos con los que cuenta un portal desarrollado con Plone van a ser entidades.

Vistas de contenidos: son templates (plantillas) que van a representar la forma en la que se va a presentar un contenido.De acuerdo con los requisitos funcionales que especifican el diseño de la interfaz del sitio solicitados por los clientes, es que se conocen qué plantillas se van a utilizar, estas van a ser diseñadas e implementadas en los próximos flujos de trabajo, por tanto no es necesario conocer cuál es el diseño de cada una.

Definición de entidades o contenidos

Archivo: Es cualquier archivo que pueda ser cargado desde el disco duro. Este puede ser un archivo de sonido, de textos, hoja de cálculo, un programa ejecutable, entre otros. Los campos con los que cuenta para su inserción son: *título*, que es obligatorio ponérselo puesto que es su identificador; *descripción*, es un pequeño resumen que es opcional; el *archivo* que se quiera insertar hay buscarlo y cuenta con un botón para esto; y unas *opciones* que pueden permitir o no, en dependencia del creador un debate del archivo que se insertó.

Documento: Un documento es una página de contenidos, usualmente un pedazo autónomo de texto que presenta información estática al usuario. Es el tipo de contenido agregado más común y el que más estrechamente representa una página web típica. Consta de varios campos necesarios para insertarlo como un *nombre corto* el cual se volverá parte de la URL del documento, un *título*, una *descripción* que va a ser utilizada más tarde por las páginas que muestran resúmenes de documentos, cuenta además con una caja de texto donde se va escribir el cuerpo del documento utilizando un formato el cual puede ser

texto estructurado, HTML o texto llano, y un *subir documento* para subir algún documento que se tenga guardado.

Carpeta: Este contenido es como una carpeta en el disco duro, es un lugar para contenido y que sea fácil de localizar luego. Las carpetas se utilizan para guardar cualquier tipo de contenidos, incluyendo carpetas; tienen solamente dos campos, *título* y *descripción*.

Carpetas inteligentes: Permiten recoger el contenido de diferentes lugares del sitio de Plone y proporciona una ubicación para ellos. Las carpetas inteligentes trabajan creando un criterio que sea común para todos los objetos que se quieran recolectar en la misma ubicación. Debido a que este tipo de contenido es algo complicado solo los administradores pueden agregarlos inicialmente luego los usuarios experimentados. Si un usuario no ve en la lista de contenidos a este es porque no tiene permiso para llegar a él. Los campos del contenido son: *título*, *descripción*, un *límite para mostrar los contenidos de búsqueda*, es decir si se pone el número cuatro cuando se vaya a buscar el contenido indicado, sólo mostrará cuatro contenidos, aunque existan más. Cuenta también con una *tabla* que muestra una lista de criterios para localizar los objetos que se deseen, para esta localización cada contenido cuenta con un campo en sus metadatos al que debe de poner una palabra que sirva de identificador y que trae por defecto Plone en una tabla y entonces la carpeta inteligente busca de acuerdo a esas palabras que hay que agregarle a su lista.

Evento: Es un acontecimiento próximo, puede ser una conferencia, reunión, u otro acontecimiento.

Imágenes: Las imágenes más comunes que se visualizan en todos los sistemas operativos y que pueden ser cargadas, son los archivos ".gif", ".jpg" o ".png". Los campos con los que cuenta son: *título*, *descripción*, *fechas de inicio y fin del evento*, es obligatorio establecerlas; lugar del evento. Cuenta también con una caja de texto para escribir información del evento (la que el usuario desee) en formato HTML (de aquí en adelante el llenado de los campos es opcional aunque es importante llenarlos para lograr un mayor grado de especificidad del evento) *asistentes*, para escribir el nombre de los asistentes al evento; la *dirección web* del evento, el nombre, teléfono y correo electrónico para contactar con el o los creadores del evento; y con unas *opciones* que pueden permitir o no, en dependencia del creador, el debate del evento.

Link: vínculo a otro ítem o artículo o cualquier recurso a la que se quiera acceder. Se puede hacer un vínculo a un recurso que esté en Internet o en una intranet. Para insertar links los campos a llenar serían

el *título*, *descripción*, la URL y las *opciones* pueden permitir o no, en dependencia del creador, un debate del contenido del link.

News item: documento que será mostrado bajo la etiqueta de las noticias. Normalmente se usan en los sitios web para desplegar noticias que son de interés al lector. News ítem puede ser llamado también artículo de noticias. Realmente un artículo de noticias contiene la misma información que un documento, la única diferencia real es que un artículo de noticias se presentará cuando un visitante pulsa el botón de noticias.

Contenido: Agrupa todas las entidades mencionadas anteriormente, estas son además los contenidos que trae por defecto Plone.

De acuerdo con los requerimientos de los clientes pueden definirse otros contenidos a través de productos que se instalen. De cada contenido deben indicarse todos los campos de los mismos, tanto los obligatorios como los opcionales.

Definir flujo de trabajo de los contenidos

Un flujo de trabajo para los contenidos se define como las posibles transiciones de un contenido de un estado a otro, Andy McKay creador de “The Definitive Guide to Plone” plantea que es la capacidad de aplicar diversos estados a los contenidos (McKay, 2005).Los diversos estados por defecto son:

Visible: cuando el contenido está mostrado de una forma visible todos los usuarios pueden encontrarlo a través de la función búsqueda y pueden acceder a él directamente visitando el objeto URL. Un contenido visible no se muestra en el árbol de navegación y es editable únicamente por su(s) dueño(s) y los administradores del sitio.

Pendiente: el contenido marcado en pendiente es un contenido que necesita ser revisado aunque desde el punto de vista de los usuarios el contenido pendiente se comporta como un contenido en estado visible. En este estado los contenidos sólo pueden ser editados por los administradores y revisores.

Publicado: los contenidos publicados son visibles para todos los visitantes del sitio. Ellos aparecen en los resultados de las búsquedas y en el árbol de navegación. Los contenidos publicados sólo pueden ser editados por los administradores pero los dueños pueden retractarlos para editarlos (retractado revierte un artículo al estado proyecto público).

Privado: los contenidos en el estado privado son visibles y editables sólo por sus dueños y otros con el acceso a la carpeta en que ellos existen, como los administradores. Estos no parecen en los resultados de las búsquedas ni en los árboles de navegación para los otros usuarios. Los contenidos en este estado son editables por los administradores solamente.

Para todos los contenidos excepto las carpetas y las carpetas inteligentes después de ser publicados deben retractarse (llevará al contenido al estado visible) para ser editado, una vez en el estado visible puede se editado; luego de esto se coloca en la cola para se revisado (pendiente). Este paso, aunque un poco molesto, es necesario para asegurarse de que todo el contenido pasa por la revisión. En la figura 7 se muestra el flujo de trabajo que propone Plone por defecto para ellos. Los dueños sólo pueden poner el contenido visible o privado y el revisor pendiente o publicado.

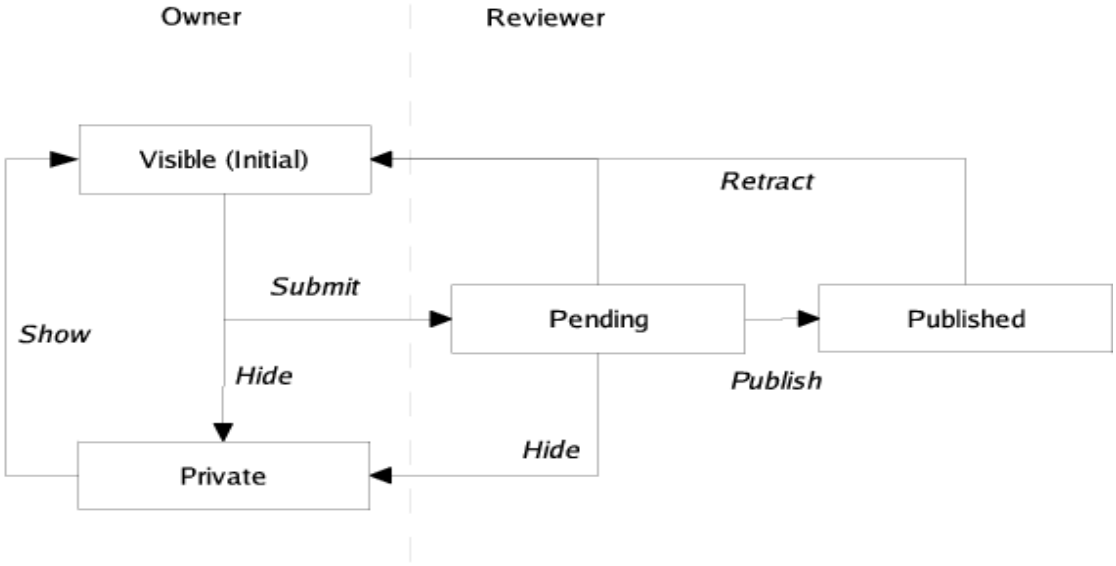

Figura 7: Flujo de trabajo para los contenidos

Carpeta

Las carpetas tienen un flujo de trabajo más sencillo que el resto de los contenidos. Debido a estas contienen contenidos pero no tienen ninguno propio, no pasan por el estado revisado (pendiente), cualquiera puede directamente publicarlas o hacer carpetas privadas; por todo esto sólo tienen tres estados: privado, visible y publicado como se muestra en la figura 8. Tanto el dueño como el revisor pueden ponerlas en cualquier estado de los mencionados.

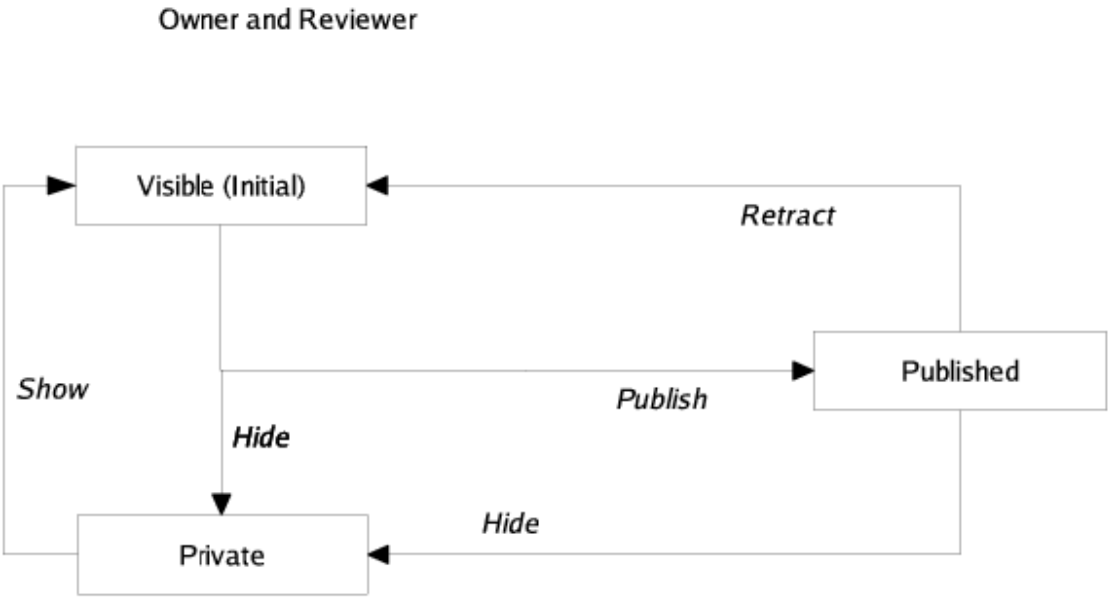

Figura 8: Flujo de trabajo para las carpetas

Cuando se agreguen contenidos al portal que sean solicitados por los clientes hay que definir el flujo de trabajo del mismo, es decir los estados por los que va a pasar y los actores que van a trabajar en ellos.
Una vez que han sido definidas las clases entidades empleadas y conociendo qué vistas van a existir de acuerdo a los requerimientos de los clientes, hay que crear el modelo de navegación. En el ejemplo que a continuación se muestra, figura 9, van a existir 3 vistas y 2 clases entidades. De la plantilla de la Página principal se puede ir hacia las vistas Galería de fotos e Imágenes, y de estas se puede retornar a la principal. En la plantilla Galería de fotos puede haber una o varias carpetas y la plantilla Imagen puede existir una o varias imágenes, todo esto, según deseen los usuarios.

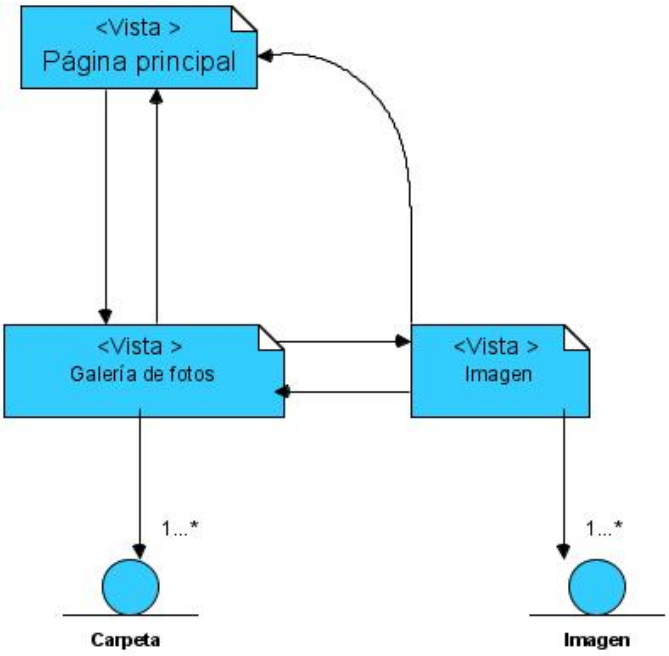

Figura 9: Ejemplo de modelo conceptual

2.3.3 Diseño

El diseño trata de inventar un enfoque práctico para el problema, dentro de los límites impuestos por las restricciones de estructuras de datos, algoritmos y partes ya existentes del sistema. El objetivo del diseño es montar la interfaz de usuario y tener el sitio de administración del sistema. Se dejará bien claro qué trabajadores, artefactos y actividades estarán involucrados en el desarrollo de este flujo de trabajo.

Trabajadores

Montador de Interfaz

Es el responsable de montar las páginas Web estáticas (sólo XHTML y CSS) logrando que se vean iguales en todos los navegadores y que se ajuste 100% al diseño gráfico recibido.

Arquitecto

Es el encargado de crear el sitio de administración del sistema, mediante la definición de cómo se realiza la gestión de los contenidos en el sistema, la estructura de almacenamiento de los contenidos, cómo agruparlos, cómo moverlos, etc., qué productos desarrollar, ya sea desde cero o modificando algunos existentes, estableciendo sus campos y métodos. Establece, además, la visibilidad de los contenidos para cada tipo de usuario en el sistema. Es el responsable de definir el flujo de trabajo en el sistema.

Artefactos

Prototipo de interfaz de usuario

Para realizar este artefacto el montador de interfaz debe montar las páginas web estáticas, que se ajusten al 100% al diseño gráfico recibido.

Diseño estructural

El objetivo del diseño estructural es crear el sitio de administración del sistema, para lograr esto el Arquitecto modelará la estructura del sitio web teniendo en cuenta los requisitos que se capturaron en la primera fase. Se debe estructurar el sitio mediante carpetas, carpetas inteligentes (smart folders) y si se requiere referencias (section). Definirá según los requisitos que productos utilizará y de esos cuales se van a desarrollar.

Diagrama de diseño

Se realizará un diagrama el cual represente la relación existente entre los componentes que conforman la interfaz de usuario, estos componentes pueden ser:

- CSS: son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Estas hojas de estilos están relacionadas con cada vista y cada contenido.
- Portlets: son templates que van a incluir funcionalidades específicas en el sitio, pueden verse como las partes fijas de cada template. Son componentes reusables de la web que muestran informaciones relevantes a los usuarios, por ejemplo: el calendario, el estado del tiempo, etc.
- Vistas de contenidos: (fue explicada en el flujo de trabajo anterior)

Actividades del flujo de trabajo de diseño

Actividad 1: Prototipar la interfaz de usuario.

Actividad 2: Definir la estructura del sitio.

Tarea 2.1: Revisar los requisitos funcionales.

Tarea 2.2: Entender cada requisito funcional.

Tarea 2.3: Realizar la estructura del sitio.

Actividad 3: Establecer permisos de usuarios.

Actividad 4: Realizar diagrama de diseño.

Tarea 4.1: Definir cuales son los elementos que conforman el sitio.

Tarea 4.2: Realizar el diagrama de diseño.

Actividad 1: Prototipar la interfaz de usuario

La primera actividad que hay que realizar es estudiar HTML y CSS, para esto se deben estudiar folletos y libros, consultar sitios en Internet. Se debe tener en cuenta que las clases e identificadores de las etiquetas que estén relacionas sean los mismos. No se debe usar tablas con fines de diseño sólo para mostrar información de forma tabular.

Actividad 2: Definir la estructura del sitio

Para definir la estructura del sitio el Arquitecto debe revisar los requisitos funcionales capturados en el flujo de trabajo de especificación de requisitos con el objetivo de agrupar información en común, por ejemplo las noticias, las cuales pueden estar relacionadas con diferentes temáticas, por lo que hay que crear esas temáticas, generalmente a través de carpetas inteligentes, para poder crear criterios de búsquedas que encuentren esa noticia y le hagan especie de un acceso directo a ella y no tener la misma noticia repetida en el sitio. Además se debe definir qué contenidos va a utilizar, (entiéndase por estos contenidos las carpetas, las carpetas inteligentes, y las referencias), así como, los productos que se deberán desarrollar en el próximo flujo de trabajo. Teniendo en cuenta todo esto realiza la estructura del sitio. En la figura 10 que se muestra a continuación se presenta un ejemplo de una estructuración de un sitio.

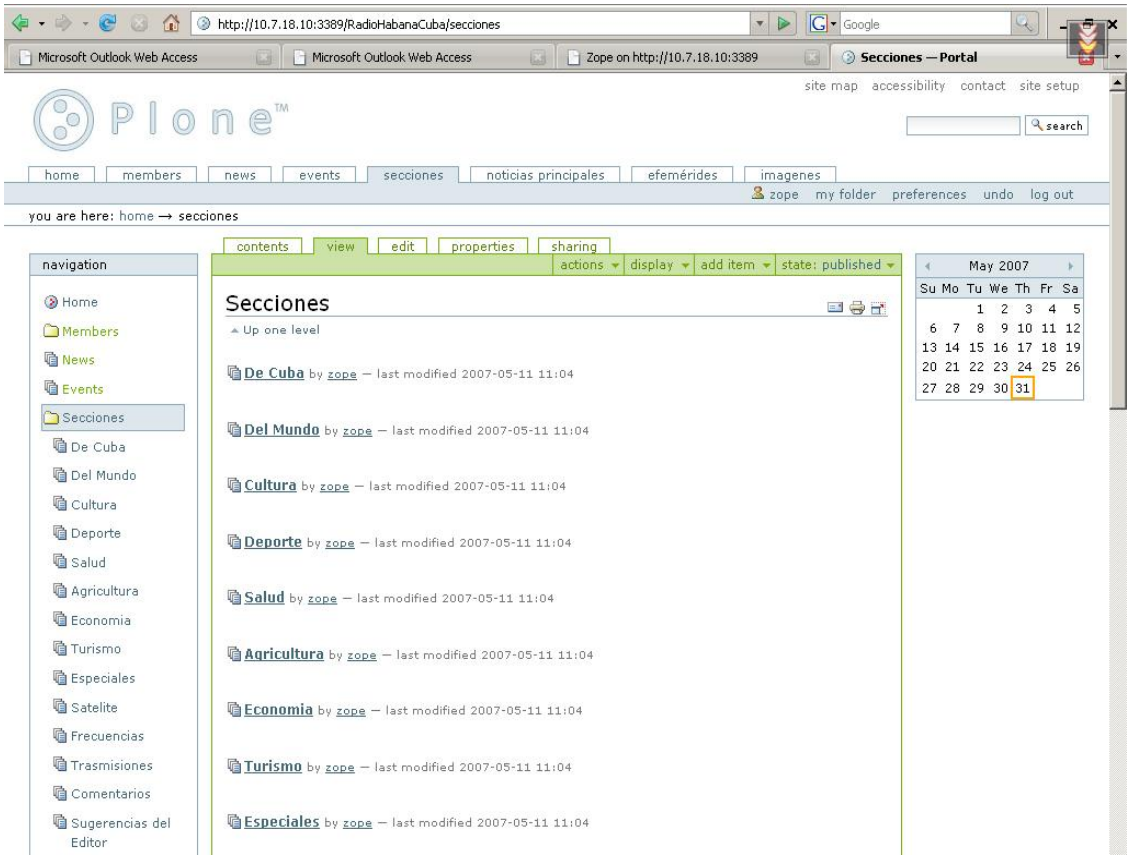

Figura 10: Ejemplo del diseño estructural de un sitio web

Actividad 3: Establecer permisos de usuarios

El Arquitecto debe definir los permisos y roles que tendrá el sitio administrativo, para así poder definir las diferentes vistas que tienen los usuarios. Esta actividad es opcional sólo se realizará si está prevista en los requisitos funcionales.

Actividad 4: Realizar diagrama de diseño

El Arquitecto teniendo en cuenta el prototipo de interfaz de usuario definirá cuáles son los componentes que conforman dicha interfaz, relacionándolos en un diagrama de diseño. Un ejemplo de este tipo de diagrama se muestra a continuación en la figura 11.

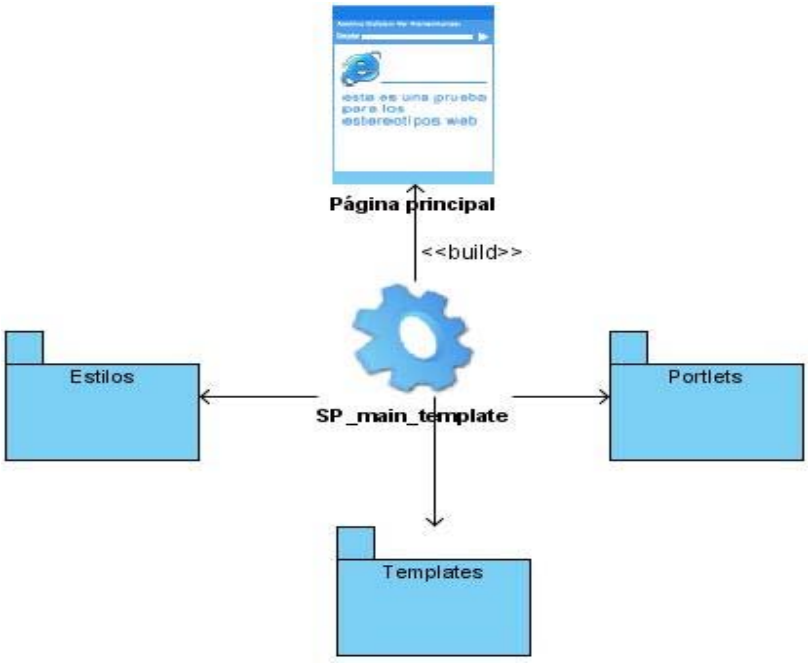

Figura 11: Ejemplo de diagrama de diseño

2.3.4 Implementación

En la implementación se realiza la construcción de la solución de un problema en un lenguaje o medio ejecutable (tal como una base de datos o un cierto hardware digital). Según RUP en la implementación se empieza con el resultado del diseño e implementamos el sistema en términos de componentes, es decir, ficheros de códigos fuentes, scripts, ficheros de códigos binarios, ejecutables y similares. En el caso de este procedimiento y teniendo en cuenta lo que plantea RUP, esta fase comenzará con el diseño y se implementará el sistema en términos de componentes.

Para llevar a cabo este flujo de trabajo es importante tener en cuenta cuatro conceptos, los cuales son de suma importancia para poder implementar un sitio web.

1. Disponibilidad del cliente: el cliente tiene que estar con el equipo de trabajo, debe responder preguntas, resolver disputas, establecer prioridades, discutir mejoras. La comunicación oral es más efectiva que la escrita.
2. Pruebas: la producción de código está dirigida por las pruebas unitarias. Estas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

3. Integración continua: cada pieza de código es integrada en el sistema una vez que esté lista y se prueba. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
4. Programación en parejas: todo el código se escribe en parejas, se produce código de mayor calidad, se extiende el conocimiento y se realiza el trabajo de una persona en casi la mitad del tiempo y mejor.

El objetivo de este flujo de trabajo es implementar el portal, esto se deberá realizar recursiva e incrementalmente, teniendo en cuenta la opinión de los clientes y mostrándoles el avance del sitio constantemente.

Trabajadores

Programador

Es el encargado de implementar el portal a partir del sitio de administración creado por el Arquitecto y de las páginas estáticas creadas por el Montador de Interfaz, de crear nuevos contenidos para Plone y Zope, y de establecer nuevos flujos de trabajos para el Plone.

Especialista en Bases de Datos

Es el que define si se utiliza la base de datos del Zope o se utiliza una base de datos externa. Integrar bases de datos externas con Plone, creación de herramientas que permitan automatizar el proceso para migrar toda la información existente en el sistema viejo para la base de datos del nuevo sistema.

Artefactos

Diagrama de implementación

En este diagrama el Programador dejará bien claro cuáles son los componentes a tener en cuenta y cuáles son las relaciones existentes entre ellos, para un mejor entendimiento de lo que se desea.

Componentes

Los componentes son cada parte que conforma un sitio web, los cuales pueden ser:

- *Templates (Plantillas)*: son elementos del lenguaje HTML que permiten definir y guardar características concretas para la presentación de textos en la pantalla que podrás aplicar en cualquier parte del documento.
- *Productos*: es un paquete que va a incluir la unión de ciertos módulos de Python y uno o más contenidos, estos módulos van a ser ficheros con extensión “.py” que contienen código de Python.

- *Scripts en el cliente o en el servidor*: los scripts en el cliente van a ser códigos de JavaScript, los cuales se van a ejecutar en el cliente y los scripts en el servidor van a ser códigos de Python que se van a ejecutar en el servidor.

Otros componentes como los que se mencionan a continuación fueron explicados anteriormente:

- Contenidos
- CSS
- Portlets
- Vistas de contenidos

Actividades del flujo de trabajo de implementación

Actividad 1: Realizar el diagrama de implementación.

Actividad 2: Crear la base de datos.

Tarea 2.1: Definir la base de datos a utilizar.

Actividad 3: Implementar los contenidos.

Actividad 4: Implementar el portal.

Actividad 1: Realizar el diagrama de implementación

El Programador debe, a partir del diagrama de diseño realizado en el flujo de trabajo de diseño, hacer un diagrama el cual represente las relaciones existentes en cada uno de los componentes descritos anteriormente en este flujo de trabajo. En la figura 12 se muestra un ejemplo de este tipo de diagrama.

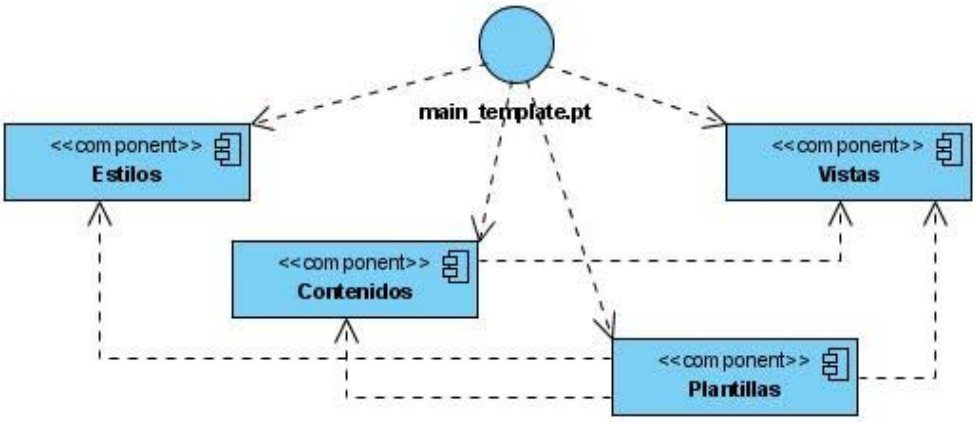

Figura 12: Ejemplo diagrama de implementación

Actividad 2: Crear la base de datos

Esta actividad se realiza si el cliente solicita agregar una base de datos extra. El Especialista en Bases de Datos se planteará las estrategias a seguir según los requisitos funcionales para decidir si usa la base de datos de Zope o utiliza una base de datos externa, en dependencia del tamaño, magnitud del sitio, si este debe conectarse a otra BD, etc.

Actividad 3: Implementar los contenidos

El Programador teniendo en cuenta los productos y contenidos que se obtuvieron en el flujo de trabajo anterior que se debían crear, los implementa con el uso de las herramientas ArgoUML y el ArchGenXML. Crea también, el flujo de trabajo perteneciente a ese contenido, incluyendo roles y permisos.

Actividad 4: Implementar el portal

El Programador debe, a partir de las páginas estáticas y del sitio administrativo resultantes del flujo de trabajo anterior, implementar el portal; para lograr esto debe utilizar los portlets, macros, y las expresiones TAL.

2.3.5 Prueba

Probar es la actividad dedicada a encontrar posibles defectos en un producto, *no es* determinar que un producto funcione. En el flujo de trabajo de prueba se verifica el resultado de la implementación probando cada construcción, incluyendo tanto construcciones internas como intermedias, así como las versiones finales del sistema a ser entregadas a terceros. Con esto se asegura que el producto final funciona correctamente como una unidad y que todo está integrado con éxito.

Trabajadores

Diseñador de prueba

Es el responsable de planificar cada inspección que se vaya a realizar y de la planificación y realización de las auditorías. Este trabajador debe conocer el funcionamiento del portal en profundidad para saber dónde pueden estar los posibles errores y así proponer las pruebas.

Probador

Es el encargado de realizar las pruebas, de entregarle el “documento Resultado de las pruebas” al Diseñador de prueba y de informar al Jefe de proyecto y a los Jefes de equipos de trabajo el resultado de las pruebas. No tiene por qué conocer el funcionamiento del portal.

Artefactos

Plan de Pruebas

Este documento recoge la planificación de las pruebas que se realizarán (tipo de prueba y fecha) y a qué elementos (productos, funcionalidades) del portal se le harán, así como una estimación del plazo de entrega del sitio web a los clientes. Incluye además todos los requerimientos que serán probados en las pruebas de sistema, cualquier requerimiento no incluido en esta lista estará fuera del alcance de las pruebas.

Resultado de pruebas

Es un documento en el que se da un reporte de la prueba que se realizó, los errores encontrados. Luego que el documento esté terminado hay que darle una copia del mismo al Jefe de proyecto y al Jefe de del equipo de desarrollo del portal que se está probando para que tenga constancia de los errores.

Actividades del flujo de trabajo de prueba

Actividad 1: Planificar las inspecciones.

Actividad 2: Realizar prueba de caja negra.

- Tarea 2.1: Probar que el sitio cumpla con los requisitos de diseño especificados por los clientes.
- Tarea 2.2: Realizar pruebas negativas.
- Tarea 2.3: Verificar la seguridad de la base de datos externa.

Actividad 3: Realizar pruebas de sistema.

Actividad 4: Realizar auditoría.

- Tarea 4.1: Planificar auditoría.
- Tarea 4.2: Realizar auditoría.

Actividad 1: Planificar las inspecciones

La primera actividad que se tiene que realizar es la planificación de las inspecciones que se harán. En cada inspección se llevan a cabo las pruebas que hayan sido planificadas y que estén recogidas en el plan de pruebas.

Actividad 2: Realizar prueba de caja negra

Para la realización de esta actividad primeramente se comprobará que el sitio realizado (puede estar completo o no) cumpla los requisitos de diseño especificados por los clientes; aquí se verificará que cada contenido contenga los campos y metadatos que exigen los diferentes sitios web, de acuerdo a sus características. De los campos y metadatos a llenar se tiene que ver si cuando no se llenan los campos obligatorios no se guarde el contenido y brinde la posibilidad de llenarlos. Se comprobará además que no existan errores ortográficos en el diseño, que las vistas de los contenidos, sean específicamente como fue solicitada, se harán pruebas de liga, es decir que cada hipervínculo vaya a la sección que le corresponda y muestre el contenido que le corresponde. También se comprueba que el flujo de trabajo sea correcto, es decir como lo solicitaron los clientes, que se puedan realizar todas las operaciones que se propusieron en la captura de requisitos, como son: insertar, eliminar, modificar; también se verifican los metadatos de cada contenido y el flujo de estos.

Al realizar las pruebas al sitio se verificará que exista homogenización en el idioma escogido y que cada actor entre al portal de acuerdo a sus privilegios.

Luego se realizan una serie de pruebas negativas en las que se van a introducir una serie de datos erróneos para intentar provocar que el sistema falle, y así, poder revelar sus debilidades. Los Probadores utilizan el sistema en formas para los que no ha sido diseñado, alguna de las pruebas más importantes son:

- Pruebas de campos, donde se solicite la entrada de una cadena de caracteres, entrar números o caracteres especiales, y viceversa.
- Prueba para verificar si el portal acepta más de un usuario con el mismo alias.

La tarea de verificar la seguridad de la base de datos (BD) externa se realiza en caso de que los clientes soliciten la utilización de una, que puede ser MySQL, PostgreSQL, Oracle, entre otras. En este caso se pueden hacer varios tipos de pruebas, como por ejemplo:

- Pruebas de concurrencia, para ver cuántas conexiones simultáneas se pueden manejar sin errores. Deben ejecutarse varios procesos de lectura y al menos dos de escritura constantemente sobre la base de datos por un período prolongado de tiempo.
- Pruebas de carga, para cuántas peticiones puede manejar el servidor sin colgarse. Deben ejecutar se un número considerable de procesos de lectura durante un período de tiempo
- Inyecciones SQL (SQL injection) para tener conocimiento de las vulnerabilidades del portal.
- Pruebas de Tiempo de respuesta (eficiencia), para tener una noción de la velocidad que tiene el gestor al manejar grandes volúmenes de datos, de esta forma se conoce el tiempo que se demora en realizar una consulta. Se deben que ejecutar varios procesos de lectura con peticiones de datos extensas durante un cierto tiempo.

Las pruebas realizadas a una BD, nunca pueden tomarse como resultados reales, aunque dan un aproximado al mismo, dependiendo del grado de acercamiento que se utilice a los procesos de la vida real. Aunque las pruebas realizadas brinden resultados que puedan parecer alentadores, no pueden ser motivo de confianza, todo lo contrario, la mejor prueba que se le puede realizar a cualquier resultado informático lo constituye la interacción directa del usuario, de aquí la importancia del proceso de mantenimiento de los portales para buscar y resolver, posibles errores que se detecten durante su vida útil.

Actividad 3: Realizar pruebas de sistema

Las pruebas del sistema se emplean para probar que el sistema (en este caso el sitio web) funcione correctamente como un todo. El objetivo de estas pruebas es buscar diferencias entre los requerimientos y el funcionamiento del portal. Para evitar sorpresas a la hora de entregar el portal, es conveniente especificar claramente qué se va a hacer para determinar que el sistema satisface los requerimientos.

Para esto se van a realizar los siguientes pasos:

1. Revisar la verificabilidad del requerimiento.
2. Especificar el criterio de verificación.

3. Hacer visible las propiedades o elementos del portal necesarios para verificar el cumplimiento del requerimiento.
4. Reportar los resultados en el documento Reporte de prueba.

Ejemplo: tomando un requerimiento de un portal, tal como un atributo de desempeño y aplicando los pasos antes mencionados se obtiene:

1. Suponiendo que el atributo restringe el tiempo de respuesta del software.
2. Este puede darse en términos absolutos (el sistema debe responder cualquier hipervínculo en menos de dos segundos) o relativos a un modelo de desempeño.
3. Para hacer observable la propiedad que interesa (tiempo de respuesta) el portal debe de estar en funcionamiento para poder obtener los tiempos de ejecución que les interesan al equipo de trabajo y a los clientes.
4. El reporte de los resultados se va a escribir en el documento de resultado de pruebas.

Actividad 4: Realizar auditoría

Una vez realizadas las pruebas, si existe un error, el Probador da un plazo de tiempo para corregir los mismos. Cuando el plazo ha sido cumplido se realiza una auditoría, esta determinará si el producto o una determinada funcionalidad del portal que se está realizando carecen de errores; y se hace con el objetivo de evaluar la corrección de los errores que fueron encontrados en las diferentes pruebas que se realizaron.

Para dar cumplimiento a esta actividad hay que planificar cuándo se va a realizar la auditoría, esta es una función que realiza el Diseñador de pruebas y va a estar en dependencia de los tipos de errores que fueron encontrados y la cantidad.

Cuando se termine la implementación y antes de entregar al cliente el portal, es necesario hacerle una prueba de caja negra para verificar si el arreglo de un error detectado no afectó funcionalidades que estaban bien; si no se encuentra ningún error, entonces el portal puede ser entregado a los clientes con confianza.

En el capítulo se realizó una propuesta de procedimiento para el desarrollo de portales web utilizando la tecnología Zope/Plone, en el cual se dieron a conocer las bases del mismo así como las características que va a tener y algunas recomendaciones para el equipo de trabajo. Se abarcó todo el ciclo de vida para el desarrollo de los portales explicando detalladamente cuáles son los trabajadores que participan, los artefactos que se obtienen y el orden de las actividades que hay que realizar en cada flujo de trabajo.

Capítulo 3 Puesta en práctica del procedimiento

En el presente capítulo se pone en práctica la propuesta de procedimiento que se realizó en el capítulo anterior. Para la aplicación se tomó como ejemplo al portal que se le está realizando a la emisora radial Radio Habana Cuba (RHC), el cual va a ser desarrollado utilizando la tecnología Zope/Plone por estudiantes del proyecto.

Para el nuevo portal de la emisora sólo se van a realizar los flujos de trabajo de especificación de requisitos, análisis, diseño y una parte de la implementación debido a que el portal aún está en desarrollo y por tanto no se han completado los últimos flujos de trabajo.

3.1 Catálogo de requisitos

Para la definición de los requisitos del nuevo portal se llevaron a cabo dos reuniones en las que participaron el Jefe de proyecto y el Analista. En la primera reunión los clientes sólo hablaron del entorno de trabajo para que el Jefe de proyecto y el Analista comprendieran cómo funcionaba. En la segunda reunión los clientes plantearon todos los requisitos que necesitaban que tuviera el portal y además explicaron el flujo de trabajo que siguen los contenidos y el portal. El resultado de estas reuniones se muestra continuación.

3.1.1 Requisitos funcionales

RF1- Insertar contenido.

RF1.1- Insertar noticias (título, descripción, nombre y foto del periodista, cuerpo de la noticia, género periodístico al que pertenece, sección a la que pertenece, imagen identificadora del contenido y nombre del editor).

RF1.2- Insertar imagen (título, pie de foto, créditos (autor, tomada de)).

RF1.3- Insertar encuesta.

RF1.4- Insertar sección.

RF2- Modificar contenido.

RF2.1- Modificar noticias (título, descripción, nombre y foto del periodista, cuerpo de la noticia, género periodístico al que pertenece, sección a la que pertenece, imagen identificadora del contenido y nombre del editor).

RF2.2 Modificar imagen (título, pie de foto, créditos (autor, tomada de)).

RF2.3- Modificar encuesta.

RF2.4- Modificar sección.

RF3- Eliminar contenido.

RF3.1- Eliminar noticias.

RF3.2- Eliminar imagen.

RF3.3- Eliminar encuesta.

RF3.4- Eliminar sección.

RF4- Publicar contenidos.

RF5- Revisar contenidos.

RF6- Autenticar usuario.

3.1.2 Requisitos no funcionales

Apariencia o interfaz externa

Va a ser sencilla y fácil de manejar. El diseño debe estar orientado a imprimirle una identidad a la emisora basada en su nueva imagen.

Usabilidad:

El sistema podrá ser usado por cualquier usuario que tenga conocimientos básicos de computación, pero está pensado y orientado a las personas que trabajan directamente con el sitio web.

Rendimiento:

Los tiempos de respuestas del sistema y los de procesamiento de la información deben ser rápidos. También debe ser rápido el tiempo de respuesta en accesos concurrentes debido a que existirán varios usuarios conectados al mismo tiempo al sitio.

Soporte:

Plone brinda la posibilidad de mejorar el sistema continuamente por lo que se puede garantizar una extensión progresiva de sus funcionalidades, esto se logra mediante la incorporación de productos.

Portabilidad:

Es capaz de ejecutarse en distintos sistemas operativos sin hacer modificaciones en el código fuente, esto se debe a que Plone resulta compatible con distintos sistemas operativos existentes en la actualidad entiéndase GNU/Linux, Windows, Mac OS X, Solaris y BSD.

Seguridad:

Para la entrada al sistema resulta obligatorio autenticarse con el objetivo de mostrar a cada usuario los servicios que le corresponde ver de acuerdo al rol que desempeñe dentro del sitio. Las contraseñas de los usuarios de la aplicación se almacenen y viajen por la red encriptadas.

Disponibilidad:

El sistema estará disponible las 24 horas del día durante todo el año ofreciendo sus servicios. Está creado para que los cambios y nuevas versiones de los productos correspondientes a cada origen de información no requieran que el sistema se detenga ni ocurra ningún daño en los servicios.

Hardware:

La máquina que se utilice como servidor para el buen funcionamiento del sistema requiere como mínimo una Pentium III, 256 de RAM, además de disponer de conectividad y requiere de gran capacidad de almacenamiento.

Navegación:

Según la página del sitio en que se encuentre el usuario siempre habrá un identificador que le indique donde se encuentra.

Legalidad:

Todas las herramientas que se utilicen en el desarrollo del sistema deben estar respaldadas por licencias bajo las condiciones de software libre.

3.1.3 Modelo de casos de uso

De acuerdo con los requisitos funcionales planteados, el modelo de casos de uso quedaría de la siguiente manera:

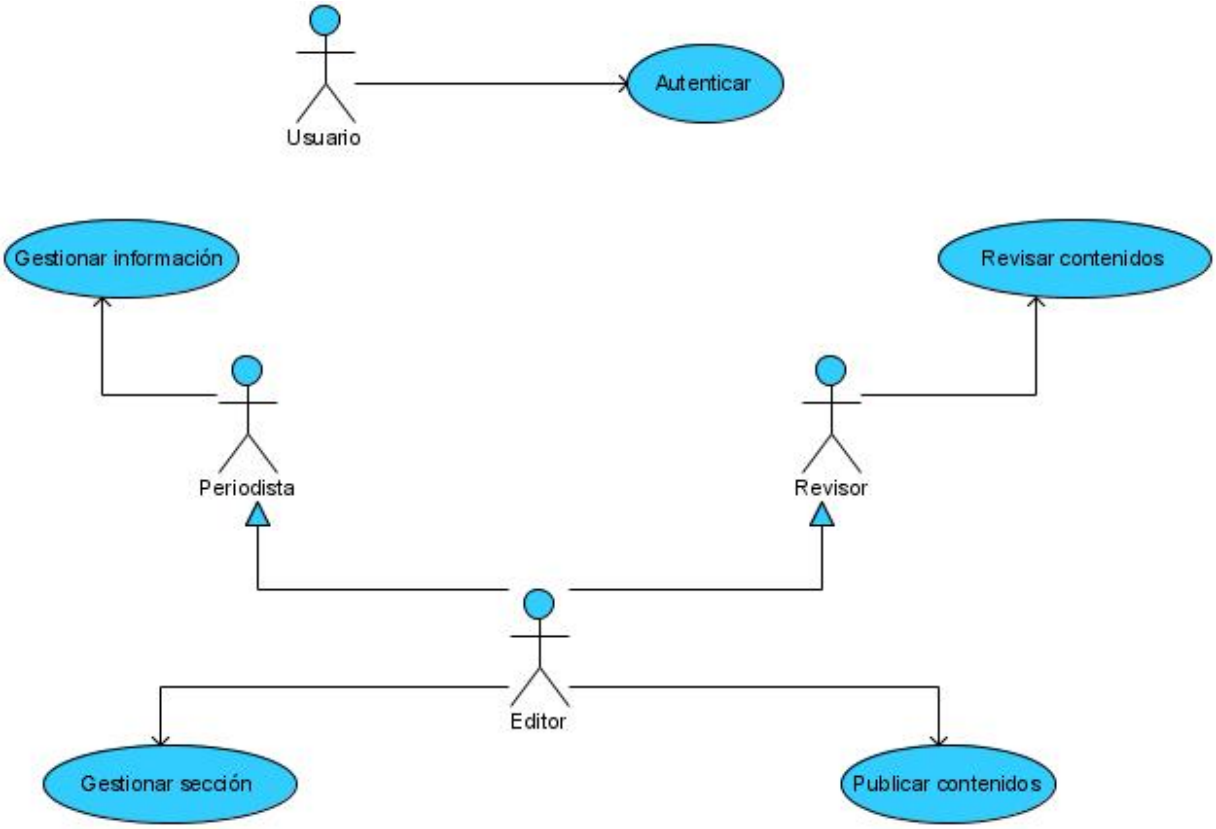

Figura 13: Modelo de casos de uso

3.1.4 Descripción de los actores del sistema

A continuación se dan a conocer los actores que van a interactuar con el sistema y se realiza una justificación de sus funciones dentro del portal.

Actores del sistema	Justificación
Editor	Es quien se ocupa de administrar el sitio. También publica los contenidos que han sido previamente revisados por el Revisor y gestiona las secciones. Los editores pueden desempeñarse como periodistas y revisores. Como Periodistas gestionan información, siendo esta: noticias, encuestas e imágenes. Como Revisores se encargan de revisar los contenidos que pueden tanto información como secciones.
Periodista	Como miembro del sitio puede, únicamente, gestionar informaciones (noticias, encuestas e imágenes).
Revisor	La función de los revisores es revisar los contenidos para luego mandarlos a publicación en caso de que no tengan problemas o errores.
Usuario	Como usuario anónimo que es, su objetivo es exclusivamente, autenticarse en el sistema. Luego que se autentique cambia de rol, pudiendo ser: Periodista, Editor o Revisor.

Tabla 3: Descripción de los actores del sistema

3.1.5 Descripción de los casos de usos

En las tablas que se muestran a continuación se describen los casos de usos que fueron definidos y mostrados en el modelo presentado anteriormente.

Tabla 4: Descripción del caso de uso: Autenticar

Caso de uso	
CU 1	Autenticar
Propósito	Permitir a los usuarios del sistema la realización de las acciones que tenga habilitadas de acuerdo a su rol.
Actor: Usuario	
Resumen: Este caso de uso se inicia cuando el Usuario, que puede tener cualquier rol, desea loguearse; para lo cual tiene que poner un nombre y una contraseña. El sistema comprueba que los datos introducidos son válidos y en caso de estar las credenciales correctas, el sistema cambia el rol del usuario y le da los permisos que tiene asignado.	
Referencias	RF6
Precondiciones: El usuario no está logueado en el sistema.	
Poscondiciones: El usuario ya está logueado en el sistema y tiene otro rol.	
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Introduce sus credenciales (usuario y contraseña) y selecciona “Entrar”.	2. Comprueba las credenciales, autentica el actor y automáticamente cambia el rol del usuario convirtiéndolo en: Periodista, Editor o Revisor, de acuerdo al rol que tome así serán los permisos que le asigne en el sistema.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
	2. En caso de no ser válidas las credenciales el sistema devuelve un error: “Login failed”(autenticación fallida) y brinda la posibilidad de autenticarse de nuevo.

Tabla 5: Descripción del caso de uso: Gestionar información

Caso de uso		
CU 2	Gestionar información	
Propósito	Insertar, modificar y eliminar la información, siendo esta: noticias, imágenes y encuestas.	
Actor: Periodista y Editor		
Resumen: El caso de uso se inicia cuando el Periodista y el Editor desean hacer alguna de las operaciones que les están permitidas, las cuales son: insertar, modificar o eliminar un tipo de información (noticias, imágenes y/o encuestas).		
Referencias	RF1, RF2, RF3	
Precondiciones	El Periodista y el Editor tienen que haber solicitado insertar, modificar o eliminar un tipo de información.	
Poscondiciones	Después de concluido el caso de uso el tipo de información solicitado queda insertado, modificado o eliminado.	
Sección "Insertar información"		
Curso normal de los eventos		
Acción del actor		**Respuesta del sistema**
1. Selecciona insertar un tipo de información 3. Llena los campos de la planilla, salva la información y lo manda al Revisor.		2. Muestra la planilla que el actor debe llenar para insertar la información.
Sección "Modificar información"		
Curso normal de los eventos		
Acción del actor		**Respuesta del sistema**
1. Selecciona el tipo de información a modificar y da clic en la ventana editar. 3. Modifica los datos de los campos de la planilla del tipo de información elegida, lo salva y lo manda al Revisor.		2. Muestra la planilla con el tipo de información seleccionada.

Sección "Eliminar información"	
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona el tipo de información a eliminar y da clic en eliminar.	2. El sistema elimina el tipo de información seleccionado por el actor.

Tabla 6: Descripción del caso de uso: Gestionar sección

Caso de uso	
CU 3	Gestionar sección
Propósito	Insertar, modificar y eliminar secciones.
Actor: Editor	
Resumen: El caso de uso se inicia cuando el Editor desea hacer alguna de las operaciones que le están permitidas, las cuales son: insertar, modificar o eliminar alguna sección.	
Referencias	RF1, RF2, RF3
Precondiciones	El editor tiene que haber solicitado insertar, modificar o eliminar una sección.
Poscondiciones	Después de concluido el caso de uso la sección queda insertada, modificada o eliminada.
Sección "Insertar información"	
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona insertar una sección. 3. Llena los campos de la planilla y salva la sección.	2. Muestra la planilla que el actor debe llenar para insertar la sección.

Sección “Modificar información”	
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la sección a modificar y da clic en la ventana editar. 3. Modifica los datos de los campos de la planilla de la sección y la salva.	2. Muestra la planilla de la sección.
Sección “Eliminar información”	
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. Selecciona la sección a eliminar y da clic en eliminar.	2. El sistema elimina la sección seleccionada por el actor

Tabla 7: Descripción del caso de uso: Publicar contenidos

Caso de uso		
CU 4	Publicar contenidos	
Propósito	Publicar los diferentes contenidos, estos son: noticias, imágenes, encuestas y secciones.	
Actor: Editor		
Resumen: El caso de uso se inicia cuando el Editor publica un contenido el cual es mandado a él por el Revisor.		
Referencias	RF4	
Precondiciones	El contenido a publicar tiene que estar en la lista de pendientes a publicación.	
Poscondiciones	Después de haber sido concluido este caso de uso el contenido queda publicado en el sitio web.	
Curso normal de los eventos		
Acción del actor		**Respuesta del sistema**
1. El Editor solicita todos los contenidos que están pendientes a publicación. 3. Selecciona un contenido, lo revisa y lo publica.		*2.* Muestra los contenidos que están en espera para ser publicados.
Flujo alternativo		
Acción del actor		**Respuesta del sistema**
3. El contenido no está bien, entonces se lo envía al Periodista para que lo arregle.		

Tabla 8: Descripción del caso de uso: Revisar contenidos

Caso de uso	
CU 5	Revisar Contenidos
Propósito	Revisar los diferentes contenidos.
Actor: Revisor	
Resumen: El caso de uso se inicia cuando el Revisor selecciona un contenido para revisarlo que ha sido enviado a él por el Periodista.	
Referencias	RF4.
Precondiciones	El contenido a publicar tuvo que haber sido insertado y agregado en la lista de pendientes a revisión.
Poscondiciones	Después de haber sido concluido este caso de uso el contenido es enviado a la lista de contenidos pendientes a publicación.
Curso normal de los eventos	
Acción del actor	**Respuesta del sistema**
1. El Revisor solicita todos los contenidos que están pendientes a revisión. 3. Selecciona un contenido, lo revisa y lo manda al Editor.	2. Muestra los contenidos que están en espera para ser revisados.
Flujo alternativo	
Acción del actor	**Respuesta del sistema**
3. El contenido no está bien, entonces se lo envía al Periodista para que este lo arregle.	

3.2 Documento del análisis

Modelo conceptual del análisis

Para realizar el modelo conceptual primeramente se definen las clases entidades con las que se cuenta, las cuales son: carpeta inteligente, encuesta, imagen y noticia. Excepto la entidad encuesta todas las demás vienen con Plone, por lo que ya están definidas sus funciones y campos, por ende, a continuación se explica lo referente a la entidad que falta.

Encuesta: grupo de preguntas con posibles respuestas, las cuales el usuario marca de acuerdo a su criterio. Los campos son: *título*, *preguntas* y *posibles respuestas*.

El flujo de trabajo que siguen los contenidos que se van a utilizar en este portal siguen el flujo de trabajo que trae Plone por defecto pero en este caso en particular quien permite pasar al estado publicado y luego retractar para llevar el contenido a visible es el Editor, que puede además retractar el contenido publicado. En la siguiente figura se muestran los estados.

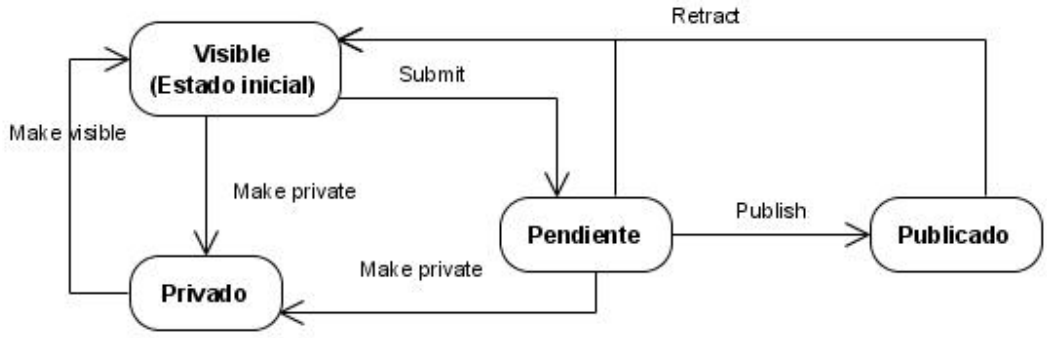

Figura 14: Flujo de trabajo para los contenidos

La noticia, en este caso consta de los siguientes campos: *título, descripción, nombre y foto del periodista, cuerpo de la noticia, género periodístico al que pertenece, sección a la que pertenece* (puede ser deporte, cultura, etc.)*, imagen identificadora del contenido y nombre del edito*r.

La imagen cuenta con los campos: *título, pie de foto* (nombre, descripción y año en que fue tomada)*, créditos* (autor, tomada de (si fue sacada de otro medio, de un colaborador o de otra persona)).

Ya definidas las clases entidades, sus campos y flujo de trabajo, se mostrará el modelo conceptual que relaciona las plantillas y las entidades como se muestra en la figura 15.

Página principal: plantilla compuesta por dos: de Noticias y de Carpetas inteligentes. De todas las plantillas antes mencionadas se puede llegar a la principal, al igual que desde la plantilla Noticias.

Noticias de portadas: plantilla que contiene a las clases entidades Noticias e Imagen. Pueden existir en la plantilla de una a diez noticias y puede no contar con imágenes, pero en caso de que haya, el máximo permitido es diez, es decir, una imagen por noticias.

Carpetas inteligentes: plantillas que contienen a la plantilla Noticias, y a la clase entidad Carpeta inteligente. Pueden existir tantas carpetas inteligentes como desee el cliente pero el mínimo permitido es una.

Noticias: plantilla que contiene a la plantilla Carpetas inteligentes e incluye las clases Noticias e Imagen. En esta plantilla pueden haber tantas noticias como el cliente desee, pero el mínimo es una y puede tener tantas imágenes quiera, no tiene límites.

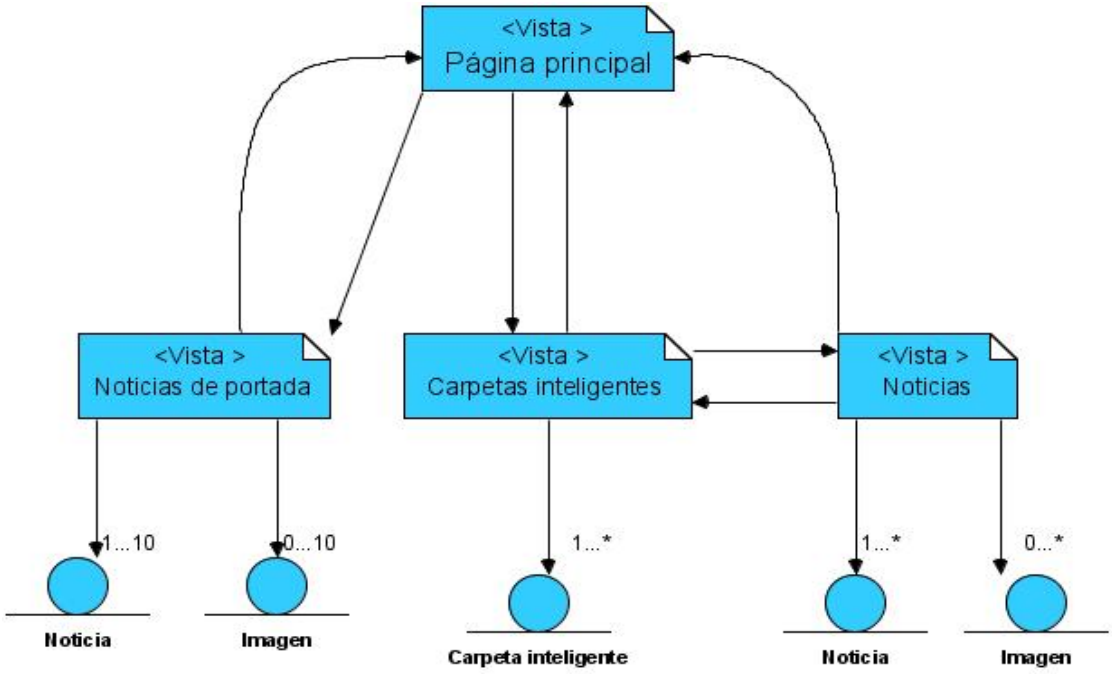

Figura 15: Modelo conceptual

3.3 Diseño

Habiendo realizado el flujo de trabajo de análisis se da paso al flujo de trabajo del diseño, a continuación se realiza y explica la primera actividad a realizar la cual es el prototipar la interfaz de usuario**,** a continuación se muestran los diferentes prototipos de interfaz de usuario que hasta el momento posee el sitio de RHC (Radio Habana Cuba).

En la figura 16 se muestra el prototipo de interfaz de la página principal, esta se divide en cuatro partes. La primera es la superior, en la cual se va a encontrar el logo de RHC, así como algunos servicios, estos elementos se mantendrán constantes en todos los prototipos. La segunda es la izquierda, en la cual se encontrará un menú con las temáticas requeridas, al igual que la primera parte, este menú será constante. La tercera es la central, que va a ser la zona editable, en la cual se mostrarán los elementos noticiosos con los que consta el sitio, además de tener el servicio de búsqueda. La cuarta es la derecha, la cual va a constar de otros servicios y de banners promocionando las diferentes efemérides del día y del mes, esta zona varía de una interfaz a otra y puede o no aparecer.

Figura 16: Prototipo de interfaz de la página principal

En la figura 17 se muestra el prototipo de interfaz de la temática *Actualidad* del menú principal, la misma se divide en cuatro partes, las dos primeras y la cuarta parte van a contener los mismos elementos que contiene la anterior y la tercera parte (la central) va a tener el listado de las noticias contenidas en esa temática, al igual que el servicio de búsqueda.

Figura 17: Prototipo de interfaz de la temática "Actualidad" del menú principal

En la figura 18 se muestra el prototipo de interfaz de las noticias, al igual que los anteriores prototipos esta se divide en cuatro partes, las dos primeras y la cuarta parte van a contener los mismos elementos que contiene la figura 16 y la 17, la tercera parte, la central, va a mostrar un material noticioso, el cual va a mostrar el título, la foto que identifica y el texto del material noticioso, así como otros servicios, al igual que el servicio de búsqueda.

Figura 18: Prototipo de interfaz de las noticias

La siguiente actividad a realizar es definir la estructura del sitio, en las figura 19, 20 y 21 se muestran la estructura del sitio a nivel de carpetas y carpetas inteligentes, figura 19. En la carpeta *Secciones* se agruparán un conjunto de carpetas inteligentes que representan las diferentes temáticas que debe tener el menú principal, figura 20. Lo que realiza cada una de las carpetas inteligentes es buscar, mediante criterios de búsqueda, los diferentes materiales noticiosos que pertenecen a esa temática. La carpeta *Imágenes*, va a contener todas las imágenes con las que conste el sitio. Las noticias van a estar localizadas en las carpetas personales de cada periodista que las crea, figura 21.

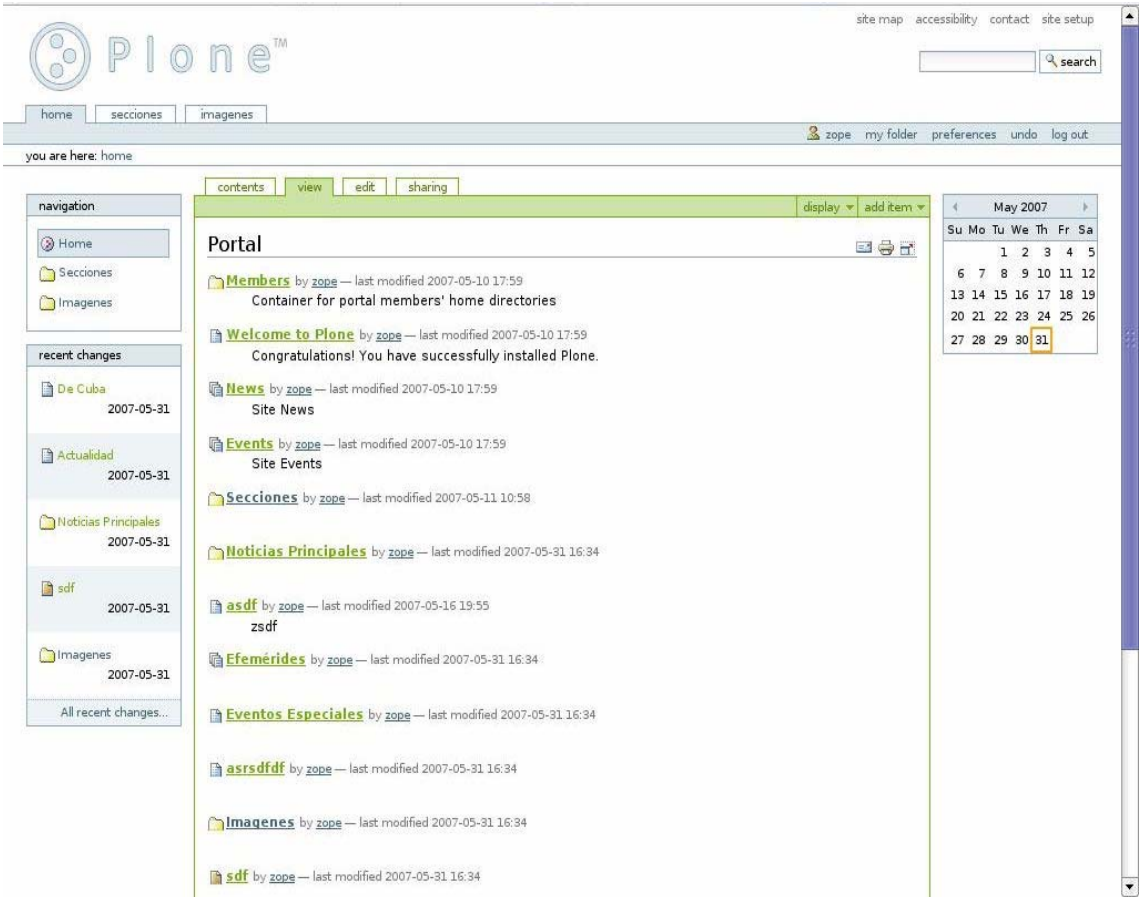

Figura 19: Estructura general del sitio

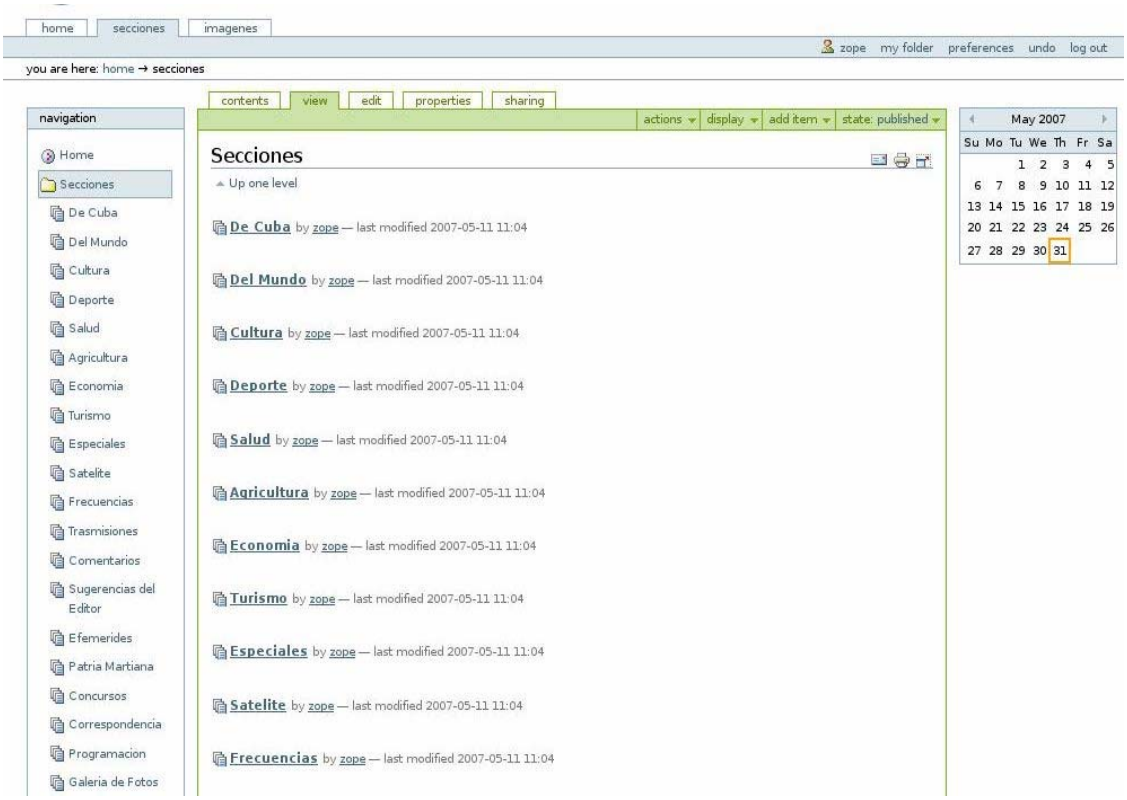

Figura 20: Estructura de las secciones

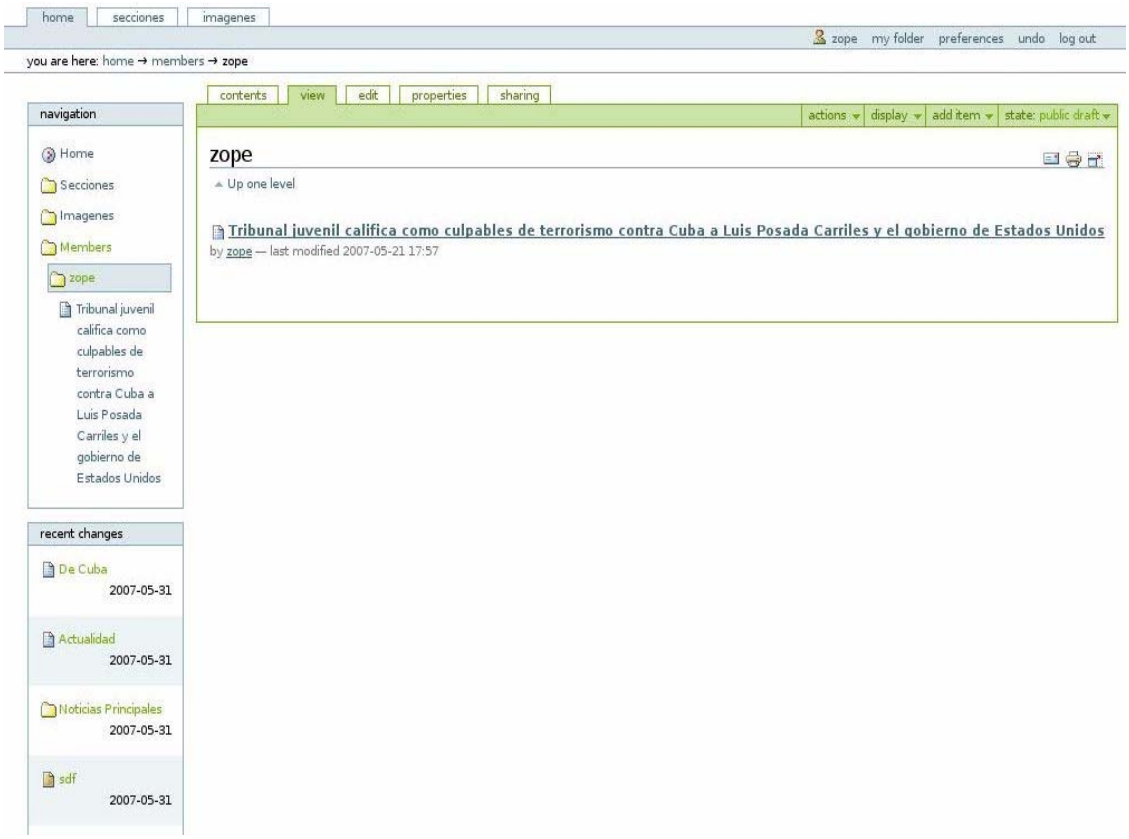

Figura 21: Estructura de las noticias a través de las carpetas personales

Diagrama de diseño

Teniendo en cuenta las actividades previamente descritas se da paso a la realización del diagrama de diseño mostrado en la figura 22. El cual describe la relación existente entre todos los componentes relacionados con los contenidos y que conforman el sitio web.

La tecnología Zope/Plone implementa por sí mismo funcionalidades como: autenticación de usuarios, publicación, y gestión de los contenidos, las cuales son necesarios para describir el procedimiento que propone este trabajo de diploma, por tal motivo en los diagramas de clases del diseño no es posible hacer un modelo de los casos de uso que contengan estas funcionalidades ya que no se conoce cómo están implementadas, no hay forma de saber qué clases utilizaron y mucho menos las relaciones que existen entre ellas. En este proyecto sólo se podrán realizar los diagramas de clases de diseño en los cuales no esté implícita ninguna de las funcionalidades antes mencionadas. Esto no trae consigo ningún tipo de

problema ya que todo lo que Plone y Zope trae implementado ha sido debidamente probado y lo hace de manera que cumple con eficiencia las necesidades que se tienen en la creación de portales web para los medios de comunicación y específicamente para la emisora radial Radio Habana Cuba.

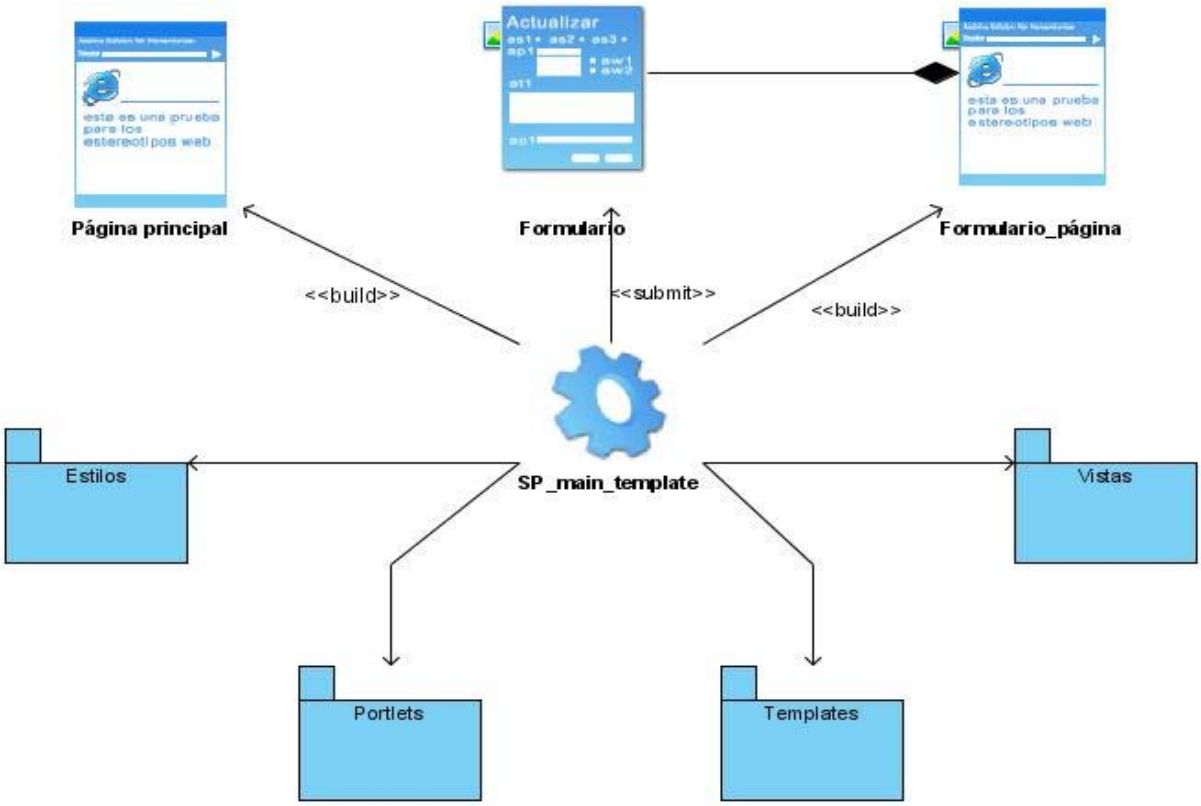

Figura 22: Diagrama de diseño

3.4 Implementación

Diagrama de implementación

En la primera iteración de este flujo de trabajo solo se realiza la actividad de elaborar el diagrama de implementación que es mostrado en la figura 23, el cual está sujeto a cambios ya que a medida que vaya avanzando el proyecto van surgiendo nuevos componentes a agregar.

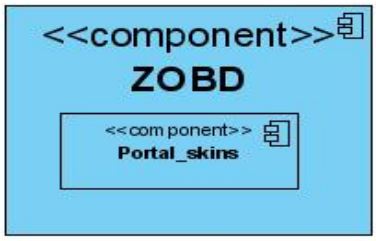

Figura 23: Diagrama de implementación

La base de datos de Zope contiene al componente Portal_skins, y este a su vez engloba los componentes que se muestran en la figura que aparece a continuación.

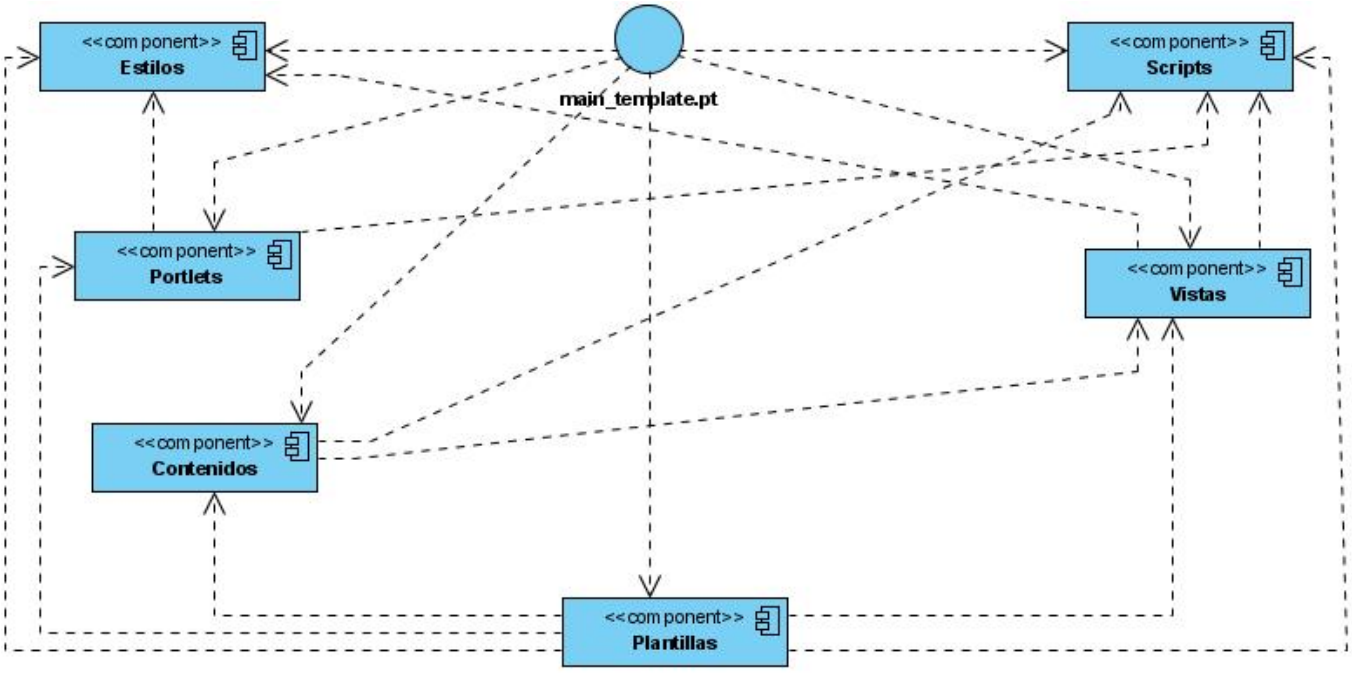

Figura 24: Diagrama de implementación del componente Portal_skins

Este diagrama está representando la relación que existe entre los diversos componentes de un sitio web. La página principal (main_template) se relaciona con los componentes: vistas, plantillas, portlets, contenidos, estilos y scripts.

El componente *contenidos* incluye todos los contenidos a utilizar siendo estos: imágenes, noticias, encuestas y carpetas inteligentes. Los contenidos contienen vistas y scripts.

El componente *estilos* contiene el conjunto de CSS con los que va a contar el sitio.
El componente *plantillas* engloba todas las plantillas que va a contener el sitio, entre las que están: main_template, header, footer, entre otras. Las plantillas engloban contenidos, estilos, portlets, vistas y scripts.
El componente *portlet* agrupa todos los portlets que va a ser utilizados como son: portlet_login, portlet_news, portlet_navigation, etc. Los portlets contienen scripts.
El componente *vistas* reúne al conjunto de vistas que van a componer al sitio. Las vistas abarcan los estilos y scripts.
El componente *scripts* incluye todos los scripts de Python y JavaScript que se realicen en la creación del sitio.

Actualmente del portal de RHC, que es caso de estudio que se presenta en este capítulo, sólo se ha trabajado en los flujos de trabajo de especificación de requisitos, análisis y diseño y en menos medida en el de implementación, puesto que existen demoras con los clientes para la entrega de prototipos del diseño, frenando de esta manera, el desarrollo del portal. Por tanto los flujos de trabajo de implementación y el de prueba no se podrán presentar en toda su magnitud porque todavía no están definidos y además están sujetos a muchos cambios.

En el capítulo se puso en práctica el procedimiento propuesto en el capítulo anterior. Se abarcaron los flujos de trabajo de: especificación de requisitos, análisis, diseño y en menor medida el de implementación. De todos ellos y siguiendo un orden para el desarrollo de las actividades, se obtuvieron de cada fase tratada los artefactos imprescindibles para la buena documentación del portal.

CONCLUSIONES

En el presente trabajo se diseñó un procedimiento para la creación de portales web utilizando la tecnología Zope/Plone. Con esta investigación se resuelve uno de los problemas que está incidiendo en el proyecto “Informatización de la prensa”, y que es precisamente la inexistencia de un lineamiento para crear aplicaciones web utilizando las tecnologías antes mencionadas.

Luego de realizar un análisis de diversas metodologías que se utilizan para la creación de portales en el mundo como son: WSDM, WebML, OOWS y UML Based Web Engineering y otras como XP y RUP que son aplicables a varios procesos de desarrollo, se arribó a la conclusión de que ninguna se adaptaba a las necesidades del proyecto. Una de las causa principales por las cuales no se eligió ninguna de las metodologías mencionadas para adecuarla a las necesidades existentes en el proyecto, es que existe muy poca información y la documentación que se puede obtener no las aborda en profundidad, ni explícitamente. XP no se utilizó debido a que casi no se documenta ninguna de las actividades que se van realizando ni lo que se va obtenido durante el desarrollo del sistema; otra dificultad que posee es que cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios que se proponen puede llevar el proceso al fracaso, puesto que estas son de estricto cumplimiento. RUP no fue empleada del todo, debido principalmente a que está elaborada para proyectos muy grandes en los cuales los requisitos están bien definidos desde el inicio y además la gran cantidad de artefactos y trabajadores con los que cuentan, ocasionan demora en el tiempo de entrega de los portales.

Para el desarrollo del procedimiento se escogieron partes de cada metodología que el equipo de investigación consideró que eran las que más se ajustaban a las necesidades del proyecto. Dichas partes fueron adaptándose a las necesidades y requerimientos de la guía, y contribuyeron en gran medida a su conformación.

Una vez conocidos los pasos a seguir para saber qué hacer, qué obtener en cada fase y qué presentar al cliente, así como todos los conceptos fundamentales de Zope y Plone, el personal del proyecto, fundamentalmente los nuevos integrantes, van a tener un cierto grado de conocimiento de la tecnología. Esto contribuye al logro de una mayor organización del trabajo, propicia la uniformidad en el mismo y de un producto final de forma eficiente, en tiempo y con calidad.

RECOMENDACIONES

- Continuar aplicando el procedimiento al sitio oficial de Radio Habana Cuba.
- Aplicar el procedimiento para el desarrollo de otros portales web en cualquier politécnico, universidad o centro de Cuba que trabaje con Zope y Plone.
- Publicar la propuesta de procedimiento en los servidores de documentación del proyecto, de la facultad y en el sitio web de la biblioteca, para que todas las personas interesadas puedan tener una mayor accesibilidad.

REFERENCIAS BIBLIOGRÁFICAS

Appelmans, Thomas. 2004. Web Globalization and WSDM Methodology of Web Design. Vrije Universiteit Bruselas. Bélgica. 62 p. [En línea]. [Citado el: 10 de abril del 2007.] http://wise.vub.ac.be/Downloads/Theses/AppelmansT-thesis-2004.pdf.

Baresi, Luciano, Fraternali, Piero y Tisi, Massimo y Morasca, Sandro. Towards Model-Driven Testing of a Web Application Generator. [En línea] [Citado el: 19 de mayo del 2007.] http://www.elet.polimi.it/upload/baresi/papers/ICWE2005.pdf.

Canós, José H. y Letelier, Patricio y Penadés, Mª Carmen. Métodologías Ágiles en el Desarrollo de Software. [En línea] [Citado el: 20 de abril del 2007.] http://www.willydev.net/descargas/prev/TodoAgil.Pdf.

Cuerda Garcia, Xavier y Minguillón Alfonso, Julià. 2004. Introducción a los Sistemas de Gestión de Contenidos (CMS) de código abierto. [En línea]. [Citado el: 16 de enero del 2007.] http://mosaic.uoc.edu/articulos/cms1204.html.

Del Castillo, Alvaro. Zope, Python y la programación web. [En línea] [Citado el: 25 de enero del 2007.] http://www.programatium.com/manuales/zope/2.htm.

Díaz, Moisés Daniel. Gestión de Contenidos en Portales Web. [En línea] [Citado el: 16 de enero del 2007.] http://www.programacion.com/html/articulo/moises_gcpw/.

Escalona Cuaresma, María José. 2001. Metodologías para el desarrollo de sistemas de información global: análisis comparativo y propuesta. [En línea]. [Citado el: 15 del febrero de 2007.] http://www.lsi.us.es/docs/informes/EstadoActual.pdf.

Escalona, M.J. & Koch, N. Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo. [En línea] [Citado el: 10 de abril del 2007.] http://www.pst.informatik.uni-muenchen.de/personen/kochn/ideas03-escalona-koch.pdf.

Fons, Joan, Pastor, Oscar y Valderas, Pedro y Ruiz, Marta. OOWS: Un Método de Producción de Software en Ambientes Web. [En línea] [Citado el: 17 de abril del 2007.] http://oomethod.dsic.upv.es/anonimo/..%5Cfiles%5CBookChapter%5Cfons02b.pdf.

Fowler, Martin. Refactoring. [En línea] [Citado el: 20 de abril del 2007.] http://www.refactoring.com/.

Fowler, Martin. 2000. The New Methodology. [En línea]. [Citado el: 17 de abril del 2007.] http://martinfowler.com/articles/newMethodology.html.

García Gómez, Juan C. y Saorín Pérez, Tomás. Los portales en Internet. [En línea] [Citado el: 18 de enero del 2007.] http://www.um.es/gtiweb/curso/seis.htm.

Informática Milenium, S.A.de C.V. 2006. Principales definiciones de los términos más usados en Internet. [En línea]. [Citado el: 17 de enero del 2007.] http://www.informaticamilenium.com.mx/paginas/espanol/sitioweb.htm.

Jacobson, Ivar y Booch, Grady & Rumbaugh, James. 2004. *El Proceso Unificado de Desarrollo de Software.* La Habana: Félix Varela,. pág. 438. [Citado el: 3 de mayo del 2007.]

McKay, Andy. 2005. The Definitive Guide to Plone. [En línea]. [Citado el: 27 de enero del 2007.] http://plone.org/documentation/manual/definitive-guide/definitive_guide_to_plone.pdf.

Tramullas, Jesús. 2005. Herramientas de software libre para la gestión de contenidos. [En línea]. [Citado el: 18 de enero del 2007.] http://www.hipertext.net/web/pag258.htm.

Wells, Donovan. 2001. The Rules and Practices of Extreme Programming. [En línea]. [Citado el: 17 de abril del 2007.] http://www.extremeprogramming.org/rules.html.

BIBLIOGRAFÍA

1. **Alvarez, Miguel Angel.** Características y ventajas de las CSS. [En línea] [Citado el: 5 de febrero del 2007.] http://www.desarrolloweb.com/articulos/182.php.

2. **Alvarez, Miguel Angel.** Qué es Python. [En línea] [Citado el: 5 de febrero del 2007.] http://www.desarrolloweb.com/articulos/1325.php.

3. **Escalona Cuaresma, María José. 2004.** Modelos y técnicas para la especificación y el análisis de la navegación en sistemas software. Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla. Sevilla. 266 p. [En línea] [Citado el: 9 de abril del 2007.]
4. http://www.lsi.us.es/docs/doctorado/tesis/tesis.pdf

5. **Funes, Hernán. 2005.** Primeros pasos con CSS. [En línea]. [Citado el: 5 de febrero del 2007.] http://www.elguruprogramador.com.ar/zonas/ver.asp?id=117.

6. **Galdames, Patricio.** XHTML. [En línea] [Citado el: 6 de febrero del 2007.] http://www.tejedoresdelweb.com/307/article-10152.html#h2_2.

7. **Gayo Avello, Daniel, y otros. 2001.** Utilización de software libre como única tecnología para el desarrollo de portales web. [En línea]. [Citado el: 31 de enero del 2007.] http://www.di.uniovi.es/~dani/publications/sisoft2001.PDF.

8. **Islas, Octavio y Gutiérrez, Fernando.** Internet como recurso de apoyo para el trabajo periodístico. [En línea] [Citado el: 25 de mayo del 2007.] http://www.mexicanadecomunicacion.com.mx/Tables/RMC/rmc87/investigar.html.

9. **Lattelier, Amos, Pelletier, Michel y McDonough, Chris y Sabaini, Peter.** The Zope Book (2.6 Edition). [En línea] [Citado el: 3 de febrero del 2007.] http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/ZopeBook-2_6.pdf.

10. **Merelo, J. J.** Introducción a los sistemas de gestión de contenidos. [En línea] [Citado el: 18 de enero del 2007.] http://geneura.ugr.es/~jmerelo/tutoriales/cms/.

11. **Piñero Pérez, Yadenis. 2007.** Metodología para la gestión de contratación en proyectos de desarrollo de Software Educativo. Universidad d la Ciencias Informáticas (UCI). Ciudad Habana. 130 p. [Citado el: 10 de abril del 2007.]

12. **Robles Prado, Tomás Javier. 2002.** Introducción a Python v1.01. [En línea]. [Citado el: 5 de abril del 2007.] http://users.servicios.retecal.es/tjavier/archivos/IntroduccionPython.pdf.

13. **Ronda León, Rodrigo. 2005.** La Arquitectura de la Información y las Ciencias de la Información. [En línea]. [Citado el: 13 de febrero del 2007.] http://www.nosolousabilidad.com/articulos/ai_cc_informacion.htm.

14. **Sánchez, Javier N. y Ortega Santamaría, Sergio. 2001.** Metodología para el desarrollo de contenidos en hipermedia. [En línea]. [Citado el: 21 de mayo del 2007.] http://www.esev.ipv.pt/3siie/actas/actas/doc04.pdf.

15. **Solís Fernández, Jonathan.** Zope un servidor de aplicaciones libre. [En línea] [Citado el: 21 de mayo del 2007.] http://www.socios.asturlinux.org/archivos/jornadas2004/ponencias/zope.pdf.

16. **W3C HTML Working Group. 2000.** XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). A Reformulation of HTML 4 in XML 1.0. [En línea]. [Citado el: 6 de febrero del 2007.] http://www.w3.org/TR/2002/REC-xhtml1-20020801/.

17. **WebML Group.** The Web Modeling Language. *2003.* [En línea] [Citado el: 17 de abril del 2007.] http://www.webml.org/webml/page3.do?ctx1=EN.

18. **—.**¿Buscas Información sobre Zope y Plone? [En línea] [Citado el: 3 de febrero del 2007.] http://www.zopeteca.com/.

19. **—.**Frequently Asked Questions about Zope 3. [En línea] [Citado el: 16 de febrero del 2007.] http://wiki.zope.org/zope3/FAQ#what-is-zope-x3.

20. —.Metodologías Ágiles. [En línea] [Citado el: 19 de enero del 2007.] http://www.chuidiang.com/chuwiki/index.php?title=Metodolog%C3%ADas_%C3%A1giles.

21. **—. 2005.** Primeros pasos con CSS. [En línea]. [Citado el: 5 de febrero del 2007.] http://www.elguruprogramador.com.ar/zonas/ver.asp?id=117.

22. **—.** Sistemas CMS para la creación de Blogs. [En línea] [Citado el: 5 de abril del 2007.] http://www.desarrolloweb.com/articulos/2129.php.

GLOSARIO DE TÉRMINOS Y SIGLAS

Applets: pequeñas aplicaciones escritas en el lenguaje de programación Java, que se difunden a través de Internet para ejecutarse en el navegador del usuario. Carece de un método main, por eso se utiliza principalmente para el trabajo de páginas web.

ArchGenXML: herramienta que permite generar productos y crear un contenidos en Plone basados en arquetipos (archetypes) a apartir de modelos UML.

ArgoUML: aplicación de código abierto utilizada en el modelado de sistemas, mediante la cual se realizan diseños en UML llevados acabo en el análisis y pre-diseño de sistemas de software.

BSD: Berkeley Software Distribution (Distribución de Software Berkeley). Se utiliza para identificar un sistema operativo derivado del sistema Unix nacido a partir de las aportaciones realizadas a ese sistema por la Universidad de California en Berkeley.

Elicitación de requisitos: es la etapa de elicitación de requisitos abarca la primera y quizás más importante fase dentro del desarrollo de un sistema informático. Uno de los retos más importantes de la elicitación de requisitos es garantizar que los requisitos del sistema sean consistentes con las necesidades de la organización donde se utilizará el mismo y con las futuras necesidades de los usuarios.

FastCGI: alternativa al CGI estándar, cuya diferencia radica principalmente en el hecho de que el servidor crea un único proceso persistente por cada programa FastCGI en lugar de uno por cada solicitud del cliente.

FTP: File Transfer Protocol (Protocolo de transferencia de ficheros). También es el nombre del programa que, tal como su nombre indica, permite copiar ficheros desde un sistema a otro a través de la red.

GNU: Gnu is Not Unix. El proyecto GNU fue iniciado por Richard Stallman con el objetivo de crear un sistema completamente libre: el sistema GNU.

GNU/Linux: (GNU con Linux o GNU+Linux) es la denominación defendida por Richard Stallman y otros para el sistema operativo que utiliza el kernel Linux en conjunto con las aplicaciones de sistema creadas por el proyecto GNU y de varios otros proyectos/grupos de software.

GPL: General Public License. Licencia creada por la Free Software Foundation. Está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

Herramienta CASE (Computer Aided Software Engineering): diversas aplicaciones informáticas destinadas aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Estas herramientas ayudan en todos los aspectos del ciclo de vida de desarrollo del software.

HTML: Hyper Text Mark-up Language. Lenguaje de programación diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web.

IEEE: corresponde a las siglas de The Institute of Electrical and Electronics Engineers, el Instituto de Ingenieros Eléctricos y Electrónicos. Asociación técnico-profesional mundial dedicada a la estandarización, entre otras cosas. Es la mayor asociación internacional sin fines de lucro formada por profesionales de las nuevas tecnologías, como ingenieros eléctricos, ingenieros en electrónica, científicos de la computación e ingenieros en telecomunicación.

IMS: Instructional Management System. Consorcio de empresas y agentes educativos que están desarrollando el marco (especificaciones y prototipos) para facilitar el crecimiento y la viabilidad de los productos de aprendizaje distribuido basados en el uso de las redes de comunicaciones (Internet/intranets). IMS enfoca su interés en el desarrollo de especificaciones para objetos didácticos.

JSP: JavaServer Pages. Es una tecnología Java que permite a los programadores generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. Permiten al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento web.

Mac OS: abreviatura de Macintosh Operating System. Sistema operativo creado por Apple.

Metadatos: pueden considerarse "datos sobre datos". Información que describe el contenido de los datos.

PCGI (Persistent CGI):. extensión de la interface CGI que permite que un programa CGI siga siendo activo a través de peticiones múltiples del navegador y mantenga una sesión con ese cliente del navegador.

PHP: acrónimo recursivo que significa “**P**HP **H**ypertext **P**re-processor”. Lenguaje de programación usado frecuentemente para la creación de contenido para sitios web con los cuales se puede programar las paginas html y los codigos de fuente.

Protocolo CGI (Common Gateway Interface): es un protocolo para comunicar un servidor web con

programas externos y funciona incorporando, dentro de la página web, una llamada a un fichero ejecutable (el programa CGI).

SCORM: Shareable Content Object Reference Model. Conjunto de especificaciones que, aplicadas a contenidos formativos, producen objetos didácticos pequeños y reusables.

Scripts: Conjunto de comandos escritos en un lenguaje interpretado para automatizar ciertas tareas de aplicación.

Sentencias SQL: es como una frase (escrita en inglés) con la que se dice lo que se quiere obtener y de donde obtenerlo.

SQL: Structured Query Language. Es un lenguaje de consulta de bases de datos relacionales. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan.

Solaris: sistema operativo desarrollado por Sun Microsystems.

SSI: Server Side Incluyes. Directivas insertadas en páginas HTML que permiten la inserción de contenido generado dinámicamente en las páginas web. Permiten al servidor modificar automáticamente el documento solicitado antes de ser enviado al navegador del cliente.

TAL: Template Attribute Language. Lenguaje de atributo creado para Zope y usado para crear templates (plantillas) dinámicas.

TIC: Tecnología de la Información y las Comunicaciones.

UML: Unified Modeling Language. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

URL: Uniform Resource Locator. Sistema unificado de identificación de recursos en la red. Es el modo

estándar de proporcionar la dirección de cualquier recurso en Internet.

WebDAV: Web-based Distributed Authoring and Versioning. Es un conjunto de extensiones HTTP que permiten a los usuarios editar y administrar archivos colaborativamente en servidores web remotos. Suele referirse al WebDav como sólo DAV.

WebGUI: es un sistema de gestión de contenido (CMS) gratuito basado en el lenguaje Perl. Dispone de versiones para diferentes sistemas operativos.

Webmaster: o **Webmistress** (en su denominación para el género femenino): es un término comúnmente usado para referirse a él/la, o los/las personas responsables de un sitio web específico.

W3C: **World Wide Web Consortium**, abreviado W3C, es un consorcio internacional que produce estándares para la World Wide Web.

XML: acrónimo de eXtensible Markup Language. Es un metalenguaje extensible de etiquetas desarrollado por el W3C. Es una manera de definir lenguajes para diferentes necesidades.

XML-RPC: protocolo extenso de servicios web que utiliza tecnología de XML para codificar llamadas y mensajes usando el http. Proporciona medios bastante sencillos por los cuales una computadora puede ejecutar un programa sobre una máquina que coopera a través de una red como el Internet.

XSL: acrónimo de eXtensible Scripting Language. Lenguaje descriptivo para formación de documentos en XML. Permite describir cómo la información contenida en un documento XML cualquiera debe ser transformada o formateada para su presentación en un medio específico.

WYSIWYG: acrónimo de What You See Is What You Get. Se aplica a los procesadores de texto y otros editores de texto con formato (como los editores de HTML) que permiten escribir un documento viendo directamente el resultado final, frecuentemente el resultado impreso.