ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE UN SITIO WEB PARA GESTIÓN INFORMATIVA Y TURISTICA DE LA "PARROQUIA NANEGAL"

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ANÁLISIS DE SISTEMAS INFORMÁTICOS

ANA ISABEL CELA HIGUERA

DIRECTOR: ING. MYRIAM PEÑAFIEL, MSc.

Quito, Febrero 2010

DECLARACIÓN

Yo, Ana Isabel Cela Higuera, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

ANA ISABEL CELA HIGUERA

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Ana Isabel Cela Higuera, bajo mi supervisión.

Ing. Myriam Peñafiel, MSc

DIRECTORA DEL PROYECTO

DEDICATORIA

La presente tesis y toda mi carrera universitaria la dedico con todo mi amor y cariño a Dios, por la oportunidad de vivir, regalarme una familia maravillosa y permitirme conocer personas muy valiosas.

A mis padres Juan Cela y Luisa Higuera por todo el apoyo y confianza que me han brindado siempre, por estar a mi lado en todo momento ya que gracias a ellos soy quien soy hoy en día. Gracias por ayudarme a cumplir mis objetivos como persona y estudiante. Por enseñarme a ser una persona de bien y creer en mí.

A mis hermanos Juan, Luis, Victor, Diana por estar conmigo y apoyarme siempre, los quiero mucho.

A mis sobrinos Miki, Alex y Emily por la alegría y desvelarse por hacerme compañía.

A mis amigas y amigos y todas esas personas que han sido parte de este esfuerzo para lograr culminar este proyecto.

AGRADECIMIENTOS

En primer lugar quiero agradecer a Dios porque sin el nada es posible.

Quiero agradecer a mis padres, que supieron inculcarme valores y formas correctas de comportamiento que me han servido a lo largo de mi vida.

Quiero agradecer a mis hermanos y hermana por estar junto a mi tanto en los buenos como en los malos momentos.

Quiero agradecer a mis amigas y amigos por sus consejos y también por sus regaños.

Quiero agradecer a todas las personas que de una u otra manera fueron y son parte importante de mi vida y siempre han estado presentes directa e indirectamente, mil gracias por todo.

Quiero agradecer a mis profesores de la Politécnica Nacional, quienes aportaron mucho para mí crecimiento personal e intelectual y en especial a mi Directora de Tesis, Ing. Myriam Peñafiel por su paciencia, amistad y sobre todo por el apoyo brindado en mi vida estudiantil y en la culminación de mi tesis.

INDICE DEL CONTENIDO

CAPITULO I.- INTRODUCCION

1.1 Planteamiento del problema

En la actualidad uno de los principales rubros de ingreso de divisas a la economía nacional es el turismo. El internet constituye uno de los medios más utilizados para difundir los atractivos turísticos de cualquier parte del mundo.

La parroquia de Nanegal, entre sus principales actividades, tiene al turismo como una importante fuente de ingresos. No obstante de ello, la Parroquia no cuenta con un medio de difusión para dar a conocer su cultura, belleza natural y demás atractivos, esto ocasionan perdidas en el aspecto turístico y económico de sus habitantes.

Además la parroquia de Nanegal no cuenta con un registro de las actividades para gestionar y controlar las obras internas de la Parroquia.

1.2 Formulación y sistematización del problema

a) **Formulación**

¿Cómo apoyar a la difusión del turismo en la Parroquia del Nanegal?

b) **Sistematización**

¿Cómo dar a conocer la información General de la Parroquia?
¿Cómo difundir la Información Turística de la Parroquia?
¿Cómo manejar la Información de los Proyectos en forma transparente?
¿Cómo dar a conocer las funciones de los miembros de H. Junta Parroquial?
¿Cómo limitar el acceso a la información por los usuarios?

1.3 Objetivos

1.3.1 Objetivo general

- Crear un Sitio Web Dinámico para gestión Informática y Turística de la Parroquia de Nanegal

1.3.2 Objetivos específicos

- Presentar Información General de la Parroquia
- Presentar Información Turística de la Parroquia.
- Permitir la consulta de actividades de los miembros de la H. Junta Parroquial.
- Manejar Información de Proyectos.
- Crear cuentas para validar usuarios.
- Permitir el acceso a otras Páginas.

1.4 Justificación

Actualmente la Parroquia de Nanegal dado los avances de la tecnología del Internet tiene como propósito promocionar el Turismo y contar con registro de las actividades que se realizan.

Por tal motivo es necesaria la creación de un Sitio Web, con el cual se dará a conocer el Turismo y las actividades de la Parroquia.

El Sitio contará con un acceso para actualización del Turismo y las actividades, permitiendo tener un control permanente de la información de la Parroquia.

1.5 Alcance y limitaciones

El Sitio Web, contará con información actualizada del Turismo y un registro de los proyectos que se realizan en la Parroquia, a
la vez que contará con un acceso, para realizar las actualizaciones necesarias de la información por parte del administrador del sitio.

1.6 Presupuesto

	Costo por Hora($)	**Horas**	**Total ($)**
Recursos Humanos			
Diseñador	8	192	1536
Programador	8	260	2080
Software			
Apache	-	-	0
PHP	-	-	0
Javascript	-	-	0
Macromedia	-	-	0
Hardware			
PC de escritorio	-	-	0
Otros			
Impresiones y copias	-	-	120
Internet	0.8	130	104
		Total:	3840

CAPITULO II.- MARCO TEORICO

2.1 Ingeniería Web

Es el proceso utilizado para crear, implantar y mantener aplicaciones y sistemas Web de alta calidad.

2.1.1 Proceso de la Ingeniería Web

Su inmediatez y evolución hace que tenga un proceso incremental y evolutivo permitiendo de este modo que el usuario se involucre activamente, facilitando el desarrollo de productos que se ajustan mucho a lo que se busca y necesita.
Las actividades que forman parte de este proceso son:

- Formulación
- Planificación
- Análisis
- Modelización
- Generación de Páginas
- Test

Formulación: Identifica objetivos y establece el alcance de la primera entrega.
Planificación: Genera la estimación del coste general del proyecto, la evaluación de riesgos y el calendario de desarrollo y fechas de entrega.
Análisis: Especifica los requerimientos e identifica el contenido.
Modelización: Se compone de dos secuencias paralelas de tareas:

- Diseño y Producción del Contenido, que forma parte del contenido.
- Diseño de la arquitectura navegación e interfaz del usuario.

Generación de Páginas: Se integra contenido, arquitectura, navegación e interfaz para crear estática o dinámicamente el aspecto más visible de la aplicación, las páginas.
Test: El test busca errores en todos los niveles: contenido, funcional, navegacional, rendimiento, etc.
El echo de que las aplicaciones residan en la red, y que inter-operen en plataformas muy distintas, hace que el proceso de test sea especialmente difícil.

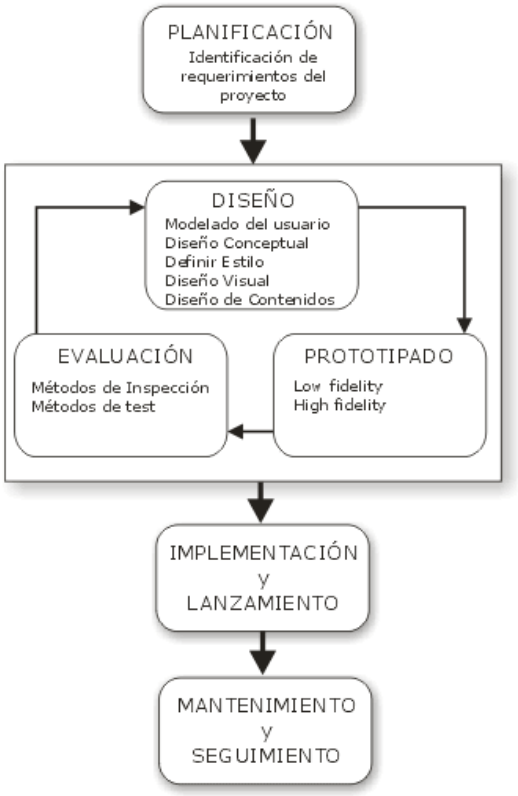
2.1.2 Diseño Web centrado en el usuario

Para asegurar empíricamente que un sitio cumple con los niveles de usabilidad requeridos, el diseñador necesita de una metodología, de técnicas y procedimientos ideados para tal fin.

En este trabajo proponemos la aplicación del marco metodológico conocido como Diseño Centrado en el Usuario o User-Centered Design adaptándolo a las características propias del desarrollo de aplicaciones web.

El Diseño Web Centrado en el Usuario se caracteriza, implica involucrar desde el comienzo a los usuarios en el proceso de desarrollo del sitio; conocer cómo son, qué necesitan, para qué usan el sitio; testar el sitio con los propios usuarios; investigar cómo reaccionan ante el diseño, cómo es su experiencia de uso; e innovar siempre con el objetivo claro de mejorar la experiencia del usuario.

El proceso de Diseño Web Centrado en el Usuario propuesto en este trabajo se divide en varias fases o etapas, algunas de las cuales tienen carácter iterativo. Sirva como aproximación el siguiente esquema:



2.1.2.1 Planificación

Todo proyecto debe comenzar por una correcta planificación. En esta etapa se identifican los objetivos del sitio, así como las necesidades, requerimientos y objetivos de la audiencia potencial.

Confrontando esta información se definen los requerimientos del sitio web, entre los que podemos contar requerimientos técnicos (back-end y front-end), recursos humanos y perfiles profesionales necesarios, y adecuación del presupuesto disponible.

Se trata, pues, de establecer un equilibrio entre lo que puede ofertar el proveedor y lo que necesita el usuario. El sitio web - sus contenidos y diseño - debe cumplir precisamente este cometido: servir de medio para la consecución de objetivos por parte de proveedor y usuario.

El diseñador debe obtener información precisa tanto de las necesidades y objetivos del proveedor como del usuario. En el primer caso, mediante entrevistas y reuniones con los responsables del sitio, será relativamente fácil obtener dicha información. Más dificultoso, pero al mismo tiempo más importante, es obtener esta información del usuario: Qué necesita, cuáles son sus objetivos, cómo se comporta y actúa, cuál será el contexto de uso y cómo afectará a la interacción, experiencia y conocimientos previos, se resuelve estudiando a la audiencia por medio de encuestas, cuestionarios y entrevistas.

2.1.2.2 Diseño

La etapa de Diseño es el momento del proceso de desarrollo para la toma de decisiones acerca de cómo diseñar o rediseñar, en base siempre al conocimiento obtenido en la etapa de planificación.

2.1.2.2.1 Modelado del usuario

Toda la información obtenida de los estudios de usuarios realizados en la anterior fase de planificación debe servir como base para comenzar el diseño, pero para ello se debe resumir y sintetizar dicha información.

Este paso se denomina modelado del usuario y consiste en la definición de clases o perfiles de usuarios en base a atributos comunes. Los atributos sobre los que se hará la clasificación dependen de la información que se tenga de la audiencia, pero normalmente se tratarán de atributos tales como necesidades de información, condiciones de acceso, experiencia y conocimientos.

Mediante esta técnica, el diseñador tendrá en mente para quién diseña, qué espera encontrar el usuario y en qué forma. El problema de esta técnica de modelado de usuario es que cuando la audiencia es demasiado extensa y heterogénea, la categorización total de la audiencia puede no ser viable, con estos arquetipos se conseguirá que el diseñador tenga en mente a un usuario ‘real’.

2.1.2.2.2 Diseño conceptual

El objetivo de la fase de Diseño Conceptual es definir el esquema de organización, funcionamiento y navegación del sitio. No se especifica qué apariencia va a tener el sitio, sino que se centra en el concepto mismo del sitio: su arquitectura de información.

La "estructura" del sitio web se refiere precisamente a las conexiones y relaciones entre páginas, a la topología de la red de páginas, así como a la granularidad de los elementos de información contenidos en las páginas; y la "navegación" a las posibilidades y forma en que cada página presenta las opciones de desplazamiento hacia otras páginas.

Si la arquitectura es ascendente normalmente se documentará a través de diagramas entidad-relación. Si es la descendente, para sitios web proponemos el uso del vocabulario gráfico de Garret. A través de unas sencillas convenciones gráficas para la diagramación de la arquitectura, podemos definir la estructura de la información así como la navegación del sitio.

2.1.2.2.3 Diseño visual y definición del estilo

En esta fase se especifica el aspecto visual del sitio web: composición de cada tipo de página, aspecto y comportamiento de los elementos de interacción y presentación de elementos multimedia.

Con el objetivo de evitar la sobrecara informativa, en el diseño de cada interfaz se debe tener en cuenta el comportamiento del usuario en el barrido visual de la página, distribuyendo los elementos de información y navegación según su importancia en zonas de mayor o menor jerarquía visual - por ejemplo, las zonas superiores del interfaz poseen más jerarquía visual que las inferiores.

Otro aspecto importante en el diseño visual del sitio es la accesibilidad. En el uso de colores, por ejemplo, se debe ofrecer suficiente contraste entre texto y fondo para no dificultar la lectura, se debe cuidar su resolución y tamaño.

2.1.2.2.4 Diseño de contenidos

En el diseño de contenidos hipermedia se debe mantener un equilibrio entre lo que serían contenidos que no aprovechasen las nuevas posibilidades hipertexto y multimedia, y lo que serían contenidos caóticos o desorientativos debido a un uso excesivo y no sosegado de las posibilidades hipermedia.

Sin prescindir de las capacidades que ofrece el nuevo medio, de lo que se trata es de diseñar contenidos interrelacionados y vinculados, manteniendo cierta coherencia informativa, comunicacional y organizativa.

La escritura hipertextual se debe realizar de forma diferente a la tradicional. El nuevo medio y sus características obligan a ser concisos, precisos, creativos y estructurados a la hora de redactar. Debemos conocer a quién nos dirigimos y adaptar el lenguaje, tono y vocabulario utilizado al usuario objetivo.

Algunos consejos a seguir en el diseño y redacción de contenidos son:

- Seguir una estructura piramidal: La parte más importante del mensaje, el núcleo, debe ir al principio.

- Permitir una fácil exploración del contenido: El lector en entornos Web, antes de empezar a leer, suele explorar visualmente el contenido para comprobar si le interesa.
- Ser conciso y preciso: Al lector no le gusta leer en pantalla.
- Vocabulario y lenguaje: Se debe utilizar el mismo lenguaje del usuario, no el de la empresa o institución. El vocabulario debe ser sencillo y fácilmente comprensible.
- Tono: Cuanto más familiar y cercano (sin llegar a ser irrespetuoso) sea el tono empleado, más fácil será que el lector preste atención.
- Confianza: La mejor forma de ganarse la confianza del lector es permitiéndole el diálogo, así como conocer cuanta más información posible acerca del autor.

2.1.2.3 Prototípico

La etapa de prototipado se basa en la elaboración de modelos o prototipos de la interfaz del sitio, la evaluación de la usabilidad se debe realizar desde las primeras etapas de diseño. Su aspecto no se corresponde exactamente con el que tendrá el sitio una vez finalizado, pero pueden servir para evaluar la usabilidad del sitio sin necesidad de esperar a su implementación. Podemos clasificar los tipos de prototipado según el nivel de funcionalidad reproducida:

- Prototipado horizontal: Se reproduce gran parte del aspecto visual del sitio, pero sin que esos modelos de interfaz estén respaldados por la funcionalidad real que tendrá finalmente el sitio.
- Prototipado vertical: Se reproduce únicamente el aspecto visual de una parte del sitio, pero la parte reproducida poseerá la misma funcionalidad que el sitio web una vez implementado.

Según el grado de fidelidad o calidad del prototipo se distingue entre:

- Prototipado de alta fidelidad: El prototipo será muy parecido al sitio web una vez terminado.

- Prototipado de baja fidelidad: El aspecto del prototipo distará bastante del que tenga el sitio web final.

El Prototipado se puede hacer en papel en las primeras etapas u otra forma de realizar prototipos es mediante la reproducción del aspecto del sitio a través de herramientas software. Mediante el procesador de textos o un simple editor HTML podemos esbozar cómo será la interfaz del sitio.

Hay que recordar que una vez que el prototipo se ha utilizado se tira, no es parte del sitio web.

2.1.2.4 Evaluación

La evaluación de la usabilidad - la etapa más importante en el proceso de Diseño Centrado en el Usuario - se puede realizar a través de varios métodos o técnicas y sobre diferentes representaciones del sitio (prototipos en papel, prototipos software, sitio web implementado...).

Existe una gran diversidad de métodos para evaluación de usabilidad, aunque en el presente trabajo únicamente se describirán aquellos que creemos de más utilidad y aplicabilidad real en el contexto del desarrollo de aplicaciones web.

2.1.2.4.1 Método por inspección

Los métodos de inspección de la usabilidad de un sitio web son aquellos realizados por el experto en usabilidad, y que se basan en el recorrido y análisis del sitio identificando errores y problemas de diseño.

La Evaluación Heurística es un tipo de método de inspección, que tiene como ventaja la facilidad y rapidez con la que se puede llevar a cabo.

Este tipo de evaluación normalmente la lleva a cabo un grupo reducido de evaluadores que, en base a su propia experiencia, fundamentándose en reconocidos principios de usabilidad (heurísticos), y apoyándose en guías elaboradas para tal fin, evalúan de forma independiente el sitio web, contrastando finalmente los resultados con el resto de evaluadores.

Algunos principios de usabilidad a través de los cuales evaluar la usabilidad.

- Visibilidad del estado del sistema: El sistema (o sitio web) siempre debe informar al usuario acerca de lo que está sucediendo. Por ejemplo, cuando en una interfaz tipo webmail se adjuntan ficheros a un mensaje, el sistema debe informar del hecho mostrando un mensaje de espera.
- Lenguaje común entre sistema y usuario: El sistema debe hablar el lenguaje del usuario, huyendo de tecnicismos incomprensibles o mensajes crípticos.
- Consistencia y estándares: La consistencia se refiere a, por ejemplo, no utilizar dos rótulos distintos para referirse a un mismo contenido, o no usar estilos diferentes dentro de un mismo sitio. Además el sitio web debe seguir estándares o convenciones de diseño ampliamente aceptados. Cuanto más se parezca un diseño y su funcionamiento al resto de sitios web, más familiar y fácil de usar resultará para el usuario.

2.1.2.4.2 Método de test con usuarios

El test con usuarios es una prueba de usabilidad que se basa en la observación y análisis de cómo un grupo de usuarios reales utiliza el sitio web, anotando los problemas de uso con los que se encuentran para poder solucionarlos posteriormente.

Como toda evaluación de usabilidad, cuanto más esperamos para su realización, más costoso resultará la reparación de los errores de diseño descubiertos. Esto quiere decir que no sólo debemos realizar este tipo de pruebas sobre el sitio web una vez implementado, sino también, sobre los prototipos del sitio.

Es una prueba complementaria a la evaluación heurística, pero un test con usuarios es más costoso, por lo que es recomendable realizarlo siempre después de una evaluación heurística, ya que sería desperdiciar tiempo y dinero utilizarlo para descubrir errores de diseño motivados por el no

cumplimiento en el desarrollo de principios generales de usabilidad (heurísticos).

La ventaja que ofrecen los test de usuarios frente a otro tipo de evaluaciones es que por un lado es una demostración con hechos, por lo que sus resultados son más fiables, y por otro porque posibilitan el descubrimiento de errores de diseño imposibles o difíciles de descubrir mediante la evaluación heurística.

2.1.2.5 Implementación y lanzamiento

En la implementación del sitio es recomendable utilizar estándares (HTML, XHTML...) para asegurar la futura compatibilidad y escalabilidad del sitio. Esto se debe a que, aunque puede ser tentador utilizar tecnologías propietarias, el panorama tecnológico puede hacerlas desaparecer o cambiar en poco tiempo.

Igualmente es recomendable separar en la implementación contenido de estilo, mediante el uso de hojas de estilo (CSS) del lado del cliente y uso de bases de datos del lado del servidor. De esta forma se facilitará tanto el rediseño del sitio como la posibilidad de adaptación dinámica del diseño a las necesidades de acceso de cada tipo de usuario.

En esta etapa del desarrollo se debe llevar, así mismo, un control de calidad de la implementación, supervisando que todo funcione y responda a cómo había sido planificado, ya que la usabilidad del sitio depende directamente de la funcionalidad. Si algo no funciona, sencillamente no se puede usar.

Una vez implementado el sitio y testada su funcionalidad se procede al lanzamiento del sitio, que consiste en su puesta a disposición para los usuarios. Se trata de un evento importante, muchas veces erróneamente apresurado debido a la necesidad de cumplir plazos de entrega.

El primer encuentro entre usuario y el sitio web modelará en gran medida la percepción que el usuario tendrá del sitio en posteriores visitas. Por ello es necesario que durante los primeros meses a partir del lanzamiento, el sitio

tenga un diseño y contenidos adaptados a este importante momento de su ciclo de vida.

Una vez realizado el lanzamiento se deben utilizar técnicas de promoción para atraer a los usuarios hacia el sitio:

- Banners publicitarios ya sea desde sitios web externos pero relacionados temáticamente con el sitio a promocionar.
- Inclusión en buscadores y directorios: La inclusión del sitio web en índices y motores de búsqueda es la técnica más eficiente para atraer usuarios. Si el sitio web es público (de acceso no limitado o controlado) se debe haber diseñado de tal forma que facilite su indización automática. Si el sitio web no es público (por ejemplo un master virtual), y los contenidos no son accesibles, se debe crear un mini-sitio público que explique toda la información posible acerca del sitio, para que este sea indizado por los buscadores.
- Campañas de correo electrónico: Si se posee una base de datos con correos electrónicos de usuarios potenciales (y es legal la posesión y uso de esta información), se puede informar directamente a estos usuarios del lanzamiento del sitio.

2.1.2.5 Mantenimiento y seguimiento

Un sitio web no es una entidad estática, es un objeto vivo cuyos contenidos cambian; cuya audiencia, necesidades y perfiles cambian, y que por lo tanto requiere de continuos rediseños y mejoras.

Estos rediseños deben ser muy sutiles, no se puede cambiar el aspecto y diseño de forma drástica de un día para otro, pues aunque estos cambios estén fundamentados en problemas de usabilidad descubiertos post-lanzamiento, los cambios pueden resultar dramáticos para los actuales usuarios que ya estaban acostumbrados y familiarizados con el actual diseño.

Los problemas de uso no detectados durante el proceso de desarrollo pueden descubrirse a través de varios métodos, principalmente a través de los mensajes y opiniones de los usuarios, y su comportamiento y uso del sitio.

2.1.2.5.1 Opiniones de los usuarios

Esta información puede ser obtenida de forma pasiva - a través de los mensajes enviados por los usuarios acerca de problemas que han tenido con el uso del sitio - o de forma activa - por medio de cuestionarios y encuestas realizadas sobre la audiencia -.

Las opiniones expresadas por los usuarios indican posibles problemas de usabilidad, pero no son en sí mismas la respuesta a estos problemas. Por ejemplo, si un usuario envía un email preguntando por qué desde la home page no encuentra un enlace al recurso X, no significa que debamos implementar este enlace, sino que posiblemente el recurso X sea poco visible o de difícil localización.

2.1.2.5.2 Comportamiento del usuario y uso del sitio

Una vez que el sitio web ha sido lanzado y es usado diariamente, tenemos a nuestra disposición una nueva fuente de información sobre el comportamiento del usuario: Los ficheros "log".

Estos, son extensos ficheros de texto plano que genera el servidor web, y en los que se registra cada una de las peticiones de páginas realizadas por los clientes al servidor.

Por cada petición del cliente al servidor se suele registrar la siguiente información:

- Dirección IP del cliente
- Identidad del usuario (para sitios con identificación)
- Password de acceso (para sitios con identificación)
- Fecha y hora de la petición entre otros más.

A través del análisis de los ficheros logs se pueden responder preguntas como: ¿quién usa el sitio? ¿Cuándo lo usa? ¿Qué páginas suelen ser las más visitadas? ¿Desde qué páginas se llega? ¿Qué términos utiliza el usuario para interrogar al buscador interno?...

Se trata realmente de una información muy valiosa que correctamente analizada, puede servirnos para la toma de decisiones sobre el rediseño en sitios web implementados.

2.2 Aplicaciones Web

En la ingeniería software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, asp.net, etc.) en la que se confía la ejecución al navegador.

Es importante mencionar que una página Web puede contener elementos que permiten una comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como por ejemplo rellenar y enviar formularios, participar en juegos diversos y acceder a gestores de base de datos de todo tipo.

2.3 Herramientas de Desarrollo

2.3.1 Javascript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

Al igual que Java, JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de Herencia, si bien ésta se realiza siguiendo el

paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM.

El lenguaje fue inventado por Brendan Eich en la empresa Netscape Communications, que es la que desarrolló los primeros navegadores web comerciales. Apareció por primera vez en el producto de Netscape llamado Netscape Navigator 2.0.

Tradicionalmente, se venía utilizando en páginas web HTML, para realizar tareas y operaciones en el marco de la aplicación únicamente cliente, sin acceso a funciones del servidor. JavaScript se ejecuta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Los autores inicialmente lo llamaron Mocha y más tarde LiveScript pero fue rebautizado como JavaScript en un anuncio conjunto entre Sun Microsystems y Netscape, el 4 de diciembre de 1995.

En 1997 los autores propusieron JavaScript para que fuera adoptado como estándar de la European Computer Manufacturers' Association ECMA, que a pesar de su nombre no es europeo sino internacional, con sede en Ginebra. En junio de 1997 fue adoptado como un estándar ECMA, con el nombre de ECMAScript. Poco después también lo fue como un estándar ISO.

JScript es la implementación de ECMAScript de Microsoft, muy similar al JavaScript de Netscape, pero con ciertas diferencias en el modelo de objetos del navegador que hacen a ambas versiones con frecuencia incompatibles.

Para evitar estas incompatibilidades, el World Wide Web Consortium diseñó el estándar Document Object Model (DOM, ó Modelo de Objetos del Documento en castellano), que incorporan Konqueror, las versiones 6 de Internet Explorer y Netscape Navigator, Opera versión 7, y Mozilla desde su primera versión.

2.3.2 Php

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor (server-side scripting) pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando las bibliotecas Qt o GTK+.

PHP es un acrónimo recursivo que significa PHP Hypertext Pre-processor (inicialmente PHP Tools, o, Personal Home Page Tools). Fue creado originalmente por Rasmus Lerdorf en 1994; sin embargo la implementación principal de PHP es producida ahora por The PHP Group y sirve como el estándar de facto para PHP al no haber una especificación formal. Publicado bajo la PHP License, la Free Software Foundation considera esta licencia como software libre.

2.3.2.1 Ventajas

- Es un lenguaje multiplataforma.
- Completamente orientado a la web.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Posee una amplia documentación en su página oficial ([2]), entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.

- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun estando dirigido a alguna en particular, el programador puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (o MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes (ver más abajo Frameworks en PHP).

2.3.2.2 Inconvenientes

- La ofuscación de código es la única forma de ocultar los fuentes.

2.3.2.3 Aplicaciones desarrolladas con PHP

- Redes Sociales
  - Facebook
- Blogs
  - WordPress
- Mambo Open Source
- Joomla
- MODx
- SMF

2.3.3 Html

HTML, siglas de **HyperText Markup Language** (Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME text/html o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (como XHTML 1.0 y posteriores) o en forma descendida directamente de SGML (como HTML 4.01 y anteriores).

Por convención, los archivos de formato HTML usan la extensión .htm o .html.

2.3.4 Apache

El **servidor HTTP Apache** es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1[1] y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. Su nombre se debe a que Behelendorf eligió ese nombre porque quería que tuviese la connotación de algo que es firme y enérgico pero no agresivo, y la tribu Apache fue la última en rendirse al que pronto se convertiría en gobierno de EEUU, y en esos momentos la preocupación de su grupo era que llegasen las empresas y "civilizasen" el paisaje que habían creado los primeros ingenieros de internet. Además Apache consistía solamente en un conjunto de parches a aplicar al servidor de NCSA. Era, en inglés, a patchy server (un servidor "parcheado").

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation.

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.

Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el

servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft[2]).

La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

2.3.4.1 Ventajas

- Modular
- Open source
- Multi-plataforma
- Extensible
- Popular (fácil conseguir ayuda/soporte)

2.3.4.2 Uso

Apache es usado primariamente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache, o que utilizarán características propias de este servidor web.

Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP/Perl/Python (y ahora también Ruby).

Este servidor web es redistribuido como parte de varios paquetes propietarios de software, incluyendo la base de datos Oracle y el IBM WebSphere application server. Mac OS X integra apache como parte de su propio servidor web y como soporte de su servidor de aplicaciones WebObjects. Es soportado de alguna manera por Borland en las herramientas de desarrollo Kylix y Delphi. Apache es incluido con Novell NetWare 6.5, donde es el servidor web por defecto, y en muchas distribuciones Linux.

Apache es usado para muchas otras tareas donde el contenido necesita ser puesto a disposición en una forma segura y confiable. Un ejemplo es al momento de compartir archivos desde una computadora personal hacia Internet. Un usuario que tiene Apache instalado en su escritorio puede colocar arbitrariamente archivos en la raíz de documentos de Apache, desde donde pueden ser compartidos.

Los programadores de aplicaciones web a veces utilizan una versión local de Apache en orden de previsualizar y probar código mientras éste es desarrollado.

Microsoft Internet Information Services (IIS) es el principal competidor de Apache, así como Sun Java System Web Server de Sun Microsystems y un anfitrión de otras aplicaciones como Zeus Web Server. Algunos de los más grandes sitios web del mundo están ejecutándose sobre Apache. La capa frontal (front end) del motor de búsqueda Google está basado en una versión modificada de Apache, denominada Google Web Server (GWS). Muchos proyectos de Wikimedia también se ejecutan sobre servidores web Apache.

2.4 Herramientas de Apoyo

2.4.1 DreamWeaver

Adobe Dreamweaver® (Dw)es una aplicación en forma de estudio (Basada por supuesto en la forma de estudio de Adobe Flash®) enfocada en la construcción y edición de sitios y aplicaciones Web basados en estándares. Creado inicialmente por Macromedia (actualmente producido por Adobe Systems). Es el programa de este tipo más utilizado en el sector del diseño y la programación web, por sus funcionalidades, su integración con otras herramientas como Adobe Flash y, recientemente, por su soporte de los estándares del World Wide Web Consortium. Su principal competidor es Microsoft Expression Web y tiene soporte tanto para edición de imágenes como para animación a través de su integración con otras herramientas

Hasta la versión MX, fue duramente criticado por su escaso soporte de los estándares de la web, ya que el código que generaba era con frecuencia sólo válido para Internet Explorer, y no validaba como HTML estándar. Esto se ha ido corrigiendo en las versiones recientes.

Se vende como parte de la suite Adobe Creative Suite 3 y 4

La gran base de este editor sobre otros es su gran poder de ampliación y personalización del mismo, puesto que en este programa, sus rutinas (como la de insertar un hipervinculo, una imagen o añadir un comportamiento) están hechas en Javascript-C, lo que le ofrece una gran flexibilidad en estas materias. Esto hace que los archivos del programa no sean instrucciones de C++ sino, rutinas de Javascript que hace que sea un programa muy fluido, que todo ello hace, que programadores y editores web hagan extensiones para su programa y lo ponga a su gusto.

Las versiones originales de la aplicación se utilizaban como simples editores WYSIWYG. Sin embargo, versiones más recientes soportan otras tecnologías web como CSS, JavaScript y algunos Framework del lado servidor.

Dreamweaver ha tenido un gran éxito desde finales de los 90 y actualmente mantiene el 90% del mercado de editores HTML. Esta aplicación está disponible tanto para la platafoma MAC como para Windows, aunque también se puede ejecutar en plataformas basadas en UNIX utilizando programas que implementan las API's de Windows, tipo Wine.

Como editor WYSIWYG que es, Dreamweaver oculta el código HTML de cara al usuario, haciendo posible que alguien no entendido pueda crear páginas y sitios web fácilmente.

Algunos desarrolladores web critican esta propuesta ya que crean páginas HTML más largas de lo que solían ser al incluir mucho código inútil, lo cual va en detrimento de la ejecución de las páginas en el navegador web. Esto puede ser especialmente cierto ya que la aplicación facilita en exceso el diseño de las páginas mediante tablas. Además, algunos desarrolladores web han criticado

Dreamweaver en el pasado porque creaba código que no cumplía con los estándares del consorcio Web (W3C).

No obstante, Adobe ha aumentado el soporte CSS y otras maneras de diseñar páginas sin tablas en versiones posteriores de la aplicación.

Dreamweaver permite al usuario utilizar la mayoría de los navegadores Web instalados en su ordenador para previsualizar las páginas web. También dispone de herramientas de administración de sitios dirigidas a principiantes como, por ejemplo, la habilidad de encontrar y reemplazar líneas de texto y código por cualquier tipo de parámetro especificado, hasta el sitio web completo. El panel de comportamientos también permite crear JavaScript básico sin conocimientos de código.

Con la llegada de la versión MX, Macromedia incorporó herramientas de creación de contenido dinámico en Dreamweaver. En lo fundamental de las herramientas HTML WYSIWYG, también permite la conexión a Bases de Datos como MySQL y Microsoft Access, para filtrar y mostrar el contenido utilizando tecnología de script como, por ejemplo, ASP (Active Server Pages), ASP.NET, ColdFusion, JSP (JavaServer Pages) y PHP sin necesidad de tener experiencia previa en programación.

Un aspecto de alta consideración de Dreamweaver es su arquitectura extensible. Es decir, permite el uso de "Extensiones". Las extensiones, tal y como se conocen, son pequeños programas, que cualquier desarrollador web puede escribir pupo (normalmente en HTML y Javascript) y que cualquiera puede descargar e instalar, ofreciendo así funcionalidades añadidas a la aplicación. Dreamweaver goza del apoyo de una gran comunidad de desarrolladores de extensiones que hacen posible la disponibilidad de extensiones gratuitas y de pago para la mayoría de las tareas de desarrollo web, que van desde simple efectos rollover hasta completas cartas de compra.

También podría decirse, que para un diseño mas rápido y a la vez fácil podría complementarse con fireworks en donde podría uno diseñar un menú o para otras creaciones de imágenes (gif web, gif websnap, gif adaptable,jpeg calidad

superior, jpeg archivo mas pequeño, gif animado websnap) para un sitio web y después exportar la imagen

2.4.2 Editor Case Power Designer

La Herramienta Líder en Modelamiento

PowerDesigner, la herramienta de modelamiento número uno de la industria, permite de manera más fácil, visualizar, analizar y manipular metadatos, logrando un efectiva arquitectura en información.

PowerDesigner brinda un enfoque basado en modelos, facilitando la implementación de arquitecturas efectivas de información. Brinda potentes técnicas de análisis, diseño y gestión de metadatos.

PowerDesigner combina varias técnicas estándar de modelamiento con herramientas líder de desarrollo, como .NET, Sybase WorkSpace, Sybase Powerbuilder, Java y Eclipse, para brindar soluciones de análisis y de diseño formal de base de datos. Además trabaja con más de 60 bases de datos relacionales.

2.4.2.1 Beneficios

- Brinda soporta abierto a ambientes heterogéneos de todas clases.
- Es altamente personalizable, permitiendo acogerse a los estándares y regulaciones.
- Facilita la arquitectura empresarial, documentando los sistemas existentes.

2.4.3 Editor Multimedia (Adobe Flash)

Adobe Flash (Macromedia Flash) es una aplicación en forma de estudio de animación que trabaja sobre *"Fotogramas"* destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma. Es actualmente escrito y distribuido por Adobe Systems, y utiliza gráficos vectoriales e imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional (el flujo de subida sólo está disponible si se

usa conjuntamente con Macromedia Flash Communication Server). En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash.

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente Aplicaciones de Internet Ricas. Son también ampliamente utilizados en anuncios de la web.

En versiones anteriores, Macromedia amplió a Flash más allá de las animaciones simples, convirtiéndolo en una herramienta de desarrollo completa, para crear principalmente elementos multimedia e interactivos para Internet.

Fue hasta 2005 perteneciente a la empresa Macromedia conocido hasta entonces como Macromedia Flash® y adquirido por Adobe Systems (desde entonces conocido como Adobe Flash) ampliando con ello su portafolio de productos dentro del mercado.

2.4.4 Editor de Gráficos Adobe FireWorks

Adobe Flash (Macromedia Flash) es una aplicación en forma de estudio de animación que trabaja sobre *"Fotogramas"* destinado a la producción y entrega de contenido interactivo para diferentes audiencias alrededor del mundo sin importar la plataforma. Es actualmente escrito y distribuido por Adobe Systems, y utiliza gráficos vectoriales e imágenes ráster, sonido, código de programa, flujo de vídeo y audio bidireccional (el flujo de subida sólo está disponible si se usa conjuntamente con Macromedia Flash Communication Server). En sentido estricto, Flash es el entorno y Flash Player es el programa de máquina virtual utilizado para ejecutar los archivos generados con Flash.

Los archivos de Flash, que tienen generalmente la extensión de archivo SWF, pueden aparecer en una página web para ser vista en un navegador, o pueden

ser reproducidos independientemente por un reproductor Flash. Los archivos de Flash aparecen muy a menudo como animaciones en páginas Web y sitios Web multimedia, y más recientemente Aplicaciones de Internet Ricas. Son también ampliamente utilizados en anuncios de la web.

En versiones anteriores, Macromedia amplió a Flash más allá de las animaciones simples, convirtiéndolo en una herramienta de desarrollo completa, para crear principalmente elementos multimedia e interactivos para Internet.

Fue hasta 2005 perteneciente a la empresa Macromedia conocido hasta entonces como Macromedia Flash® y adquirido por Adobe Systems (desde entonces conocido como Adobe Flash) ampliando con ello su portafolio de productos dentro del mercado.

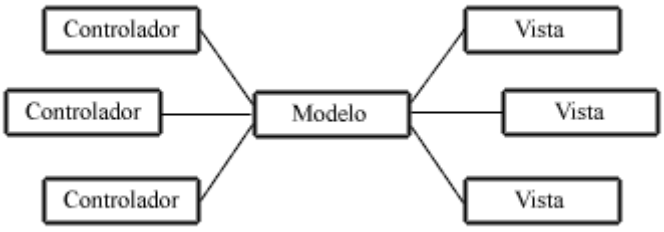
CAPITULO III.- ASPECTOS METODOLOGICOS

3.1 Arquitectura Web

La arquitectura MVC (Model/View/Controller) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Sus características principales son que el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Además del programa ejemplo que hemos presentado al principio y que posteriormente implementaremos, este modelo de arquitectura se puede emplear en sistemas de representación gráfica de datos, como se ha citado, o en sistemas CAD, en donde se presentan partes del diseño con diferente escala de aumento, en ventanas separadas.

En la figura siguiente, vemos la arquitectura MVC en su forma más general. Hay un Modelo, múltiples Controladores que manipulan ese Modelo, y hay varias Vistas de los datos del Modelo, que cambian cuando cambia el estado de ese Modelo.



Este modelo de arquitectura presenta varias ventajas:

- Hay una clara separación entre los componentes de un programa; lo cual nos permite implementarlos por separado
- Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, la Vista o el Controlador, sin aparente dificultad.

- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si uno de los Componentes, posteriormente, se observa que funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas. Este escenario contrasta con la aproximación monolítica típica de muchos programas Java. Todos tienen un Frame que contiene todos los elementos, un controlador de eventos, un montón de cálculos y la presentación del resultado. Ante esta perspectiva, hacer un cambio aquí no es nada trivial.

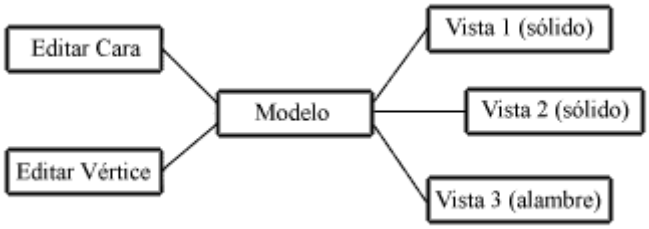
3.1.1 Definición de las partes

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El Controlador es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

Vamos a mostrar un ejemplo concreto. Consideremos como tal el sistema descrito en la introducción a este capítulo, una pieza geométrica en tres dimensiones, que representamos en la figura siguiente:



En este caso, la pieza central de la escena en tres dimensiones es el Modelo. El Modelo es una descripción matemática de los vértices y las caras que componen la escena. Los datos que describen cada vértice o cara pueden modificarse (quizás como resultado de una acción del usuario, o una distorsión de la escena, o un algoritmo de sombreado). Sin embargo, no tiene noción del punto de vista, método de presentación, perspectiva o fuente de luz. El Modelo es una representación pura de los elementos que componen la escena.

La porción del programa que transforma los datos dentro del Modelo en una presentación gráfica es la Vista. La Vista incorpora la visión del Modelo a la escena; es la representación gráfica de la escena desde un punto de vista determinado, bajo condiciones de iluminación determinadas.

El Controlador sabe que puede hacer el Modelo e implementa el interface de usuario que permite iniciar la acción. En este ejemplo, un panel de datos de entrada es lo único que se necesita, para permitir añadir, modificar o borrar vértices o caras de la figura.

De la misma forma en que los arquitectos tradicionales diseñan y coordinan la construcción de edificios, los arquitectos web diseñan y coordinan el desarrollo de sitios web.

Los sitios web son una conjunción muy compleja de distintos sistemas integrados entre sí (Bases de datos, servidores, redes, componentes de backup y seguridad, etc...).

El resultado final será un sitio que pueda resolver las necesidades de negocios: Vender productos y servicios online y servir mejor a las necesidades de los clientes.

En el diseño de sitios web, igual que en el diseño de edificios, se requiere un firme conocimiento de las tecnologías aplicadas.

En el diseño de edificios estos conocimientos son sobre las propiedades estructurales de los materiales, electricidad, mecánica, plomería, etc... En el desarrollo web se requieren de conocimientos de lenguajes programación y estructura de bases de datos, el protocolo TCP/IP, el lenguaje HTML y muchos otros.

En ambos casos es necesario tener conocimientos generales y ser un voraz estudiante de las tecnologías, dado que estas están en continuo desarrollo y avanzan día a día.

El modo de crear los documentos HTML ha variado a los largo de la corta vida de as tecnologías Web pasando desde las primeras paginas escritas en HTML almacenadas en un fichero en el servidor Web hasta aquellas que se generan ala vuelo como respuesta a una acción del cliente y cuyo contenido varía según las circunstancias.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

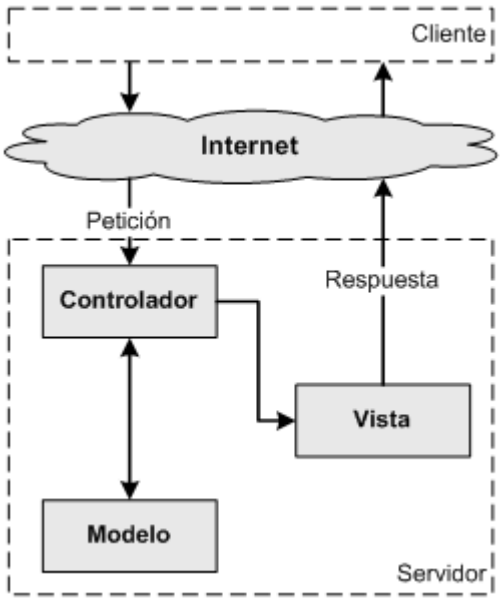


Figura 2.3. El patrón MVC

3.1.1. Las capas de la arquitectura MVC

Para poder entender las ventajas de utilizar el patrón MVC, se va a transformar una aplicación simple realizada con PHP en una aplicación que sigue la arquitectura MVC. Un buen ejemplo para ilustrar esta explicación es el de mostrar una lista con las últimas entradas o artículos de un blog.

3.1.1.1. Programación simple

Utilizando solamente PHP normal y corriente, el script necesario para mostrar los artículos almacenados en una base de datos se muestra en el siguiente listado:

Un script simple

```
<?php

// Conectar con la base de datos y seleccionarla
$conexion = mysql_connect('localhost', 'miusuario', 'micontrasena');
mysql_select_db('blog_db', $conexion);

// Ejecutar la consulta SQL
$resultado = mysql_query('SELECT fecha, titulo FROM articulo',
$conexion);

?>
```

```
<html>
  <head>
    <title>Listado de Artículos</title>
  </head>
  <body>
   <h1>Listado de Artículos</h1>
   <table>
     <tr><th>Fecha</th><th>Titulo</th></tr>
<?php
// Mostrar los resultados con HTML
while ($fila = mysql_fetch_array($resultado, MYSQL_ASSOC))
{
echo "\t<tr>\n";
printf("\t\t<td> %s </td>\n", $fila['fecha']);
printf("\t\t<td> %s </td>\n", $fila['titulo']);
echo "\t</tr>\n";
}
?>
    </table>
  </body>
</html>

<?php

// Cerrar la conexion
mysql_close($conexion);

?>
```

El script anterior es fácil de escribir y rápido de ejecutar, pero muy difícil de mantener y actualizar. Los principales problemas del código anterior son:

- No existe protección frente a errores (¿qué ocurre si falla la conexión con la base de datos?).
- El código HTML y el código PHP están mezclados en el mismo archivo e incluso en algunas partes están entrelazados.
- El código solo funciona si la base de datos es MySQL.

3.1.1.2 Separando la presentación

Las llamadas a `echo` y `printf` del listado 2-3- dificultan la lectura del código. De hecho, modificar el código HTML del script anterior para mejorar la presentación es un follón debido a cómo está programado. Así que el código va a ser dividido en dos partes. En primer lugar, el código PHP puro con toda la

lógica de negocio se incluye en el script del controlador, como se muestra en el script siguiente.

La parte del controlador, en index.php

```
<?php

// Conectar con la base de datos y seleccionarla
$conexion = mysql_connect('localhost', 'miusuario', 'micontrasena');
mysql_select_db('blog_db', $conexion);

// Ejecutar la consulta SQL
$resultado = mysql_query('SELECT fecha, titulo FROM articulo',
$conexion);

// Crear el array de elementos para la capa de la vista
$articulos = array();
while ($fila = mysql_fetch_array($resultado, MYSQL_ASSOC))
{
  $articulos[] = $fila;
}

// Cerrar la conexión
mysql_close($conexion);

// Incluir la lógica de la vista
require('vista.php');
```

El código HTML, que contiene cierto código PHP a modo de plantilla, se almacena en el script de la vista, como se muestra en siguiente script.

La parte de la vista, en vista.php

```
<html>
  <head>
    <title>Listado de Artículos</title>
  </head>
  <body>
    <h1>Listado de Artículos</h1>
    <table>
      <tr><th>Fecha</th><th>Título</th></tr>
    <?php foreach ($articulos as $articulo): ?>
      <tr>
        <td><?php echo $articulo['fecha'] ?></td>
        <td><?php echo $articulo['titulo'] ?></td>
      </tr>
    <?php endforeach; ?>
    </table>
```

```
  </body>
</html>
```

Una buena regla general para determinar si la parte de la vista está suficientemente limpia de código es que debería contener una cantidad mínima de código PHP, la suficiente como para que un diseñador HTML sin conocimientos de PHP pueda entenderla. Las instrucciones más comunes en la parte de la vista suelen ser echo, if/endif, foreach/endforeach y poco más. Además, no se deben incluir instrucciones PHP que generen etiquetas HTML.

Toda la lógica se ha centralizado en el script del controlador, que solamente contiene código PHP y ningún tipo de HTML. De hecho, y como puedes imaginar, el mismo controlador se puede reutilizar para otros tipos de presentaciones completamente diferentes, como por ejemplo un archivo PDF o una estructura de tipo XML.

3.1.1.3 Separando la manipulación de los datos

La mayor parte del script del controlador se encarga de la manipulación de los datos. Pero, ¿qué ocurre si se necesita la lista de entradas del blog para otro controlador, por ejemplo uno que se dedica a generar el canal RSS de las entradas del blog? ¿Y si se quieren centralizar todas las consultas a la base de datos en un único sitio para evitar duplicidades? ¿Qué ocurre si cambia el modelo de datos y la tabla articulo pasa a llamarse articulo_blog? ¿Y si se quiere cambiar a PostgreSQL en vez de MySQL? Para poder hacer todo esto, es imprescindible eliminar del controlador todo el código que se encarga de la manipulación de los datos y ponerlo en otro script, llamado el modelo, tal y como se muestra en el siguiente script.

La parte del modelo, en modelo.php

```
<?php

function getTodosLosArticulos()
{
  // Conectar con la base de datos y seleccionarla
  $conexion = mysql_connect('localhost', 'miusuario', 'micontrasena');
  mysql_select_db('blog_db', $conexion);

  // Ejecutar la consulta SQL
```

```
  $resultado = mysql_query('SELECT fecha, titulo FROM articulo',
$conexion);

  // Crear el array de elementos para la capa de la vista
  $articulos = array();
  while ($fila = mysql_fetch_array($resultado, MYSQL_ASSOC))
  {
    $articulos[] = $fila;
  }

  // Cerrar la conexión
  mysql_close($conexion);

  return $articulos;
}
```

El controlador modificado se puede ver en el siguiente script.

La parte del controlador, modificada, en index.php

```
<?php

// Incluir la lógica del modelo
require_once('modelo.php');

// Obtener la lista de artículos
$articulos = getTodosLosArticulos();

// Incluir la lógica de la vista
require('vista.php');
```

Ahora el controlador es mucho más fácil de leer. La única tarea es la de obtener los datos del modelo y pasárselos a la vista. En las aplicaciones más complejas, el controlador se encarga además de procesar las peticiones, las sesiones de los usuarios, la autenticación, etc. El uso de nombres apropiados para las funciones del modelo hace que sea innecesario añadir comentarios al código del controlador.

El script del modelo solamente se encarga del acceso a los datos y puede ser reorganizado a tal efecto. Todos los parámetros que no dependen de la capa de datos (como por ejemplo los parámetros de la petición del usuario) se deben obtener a través del controlador y por tanto, no se puede acceder a ellos directamente desde el modelo. Las funciones del modelo se pueden reutilizar fácilmente en otros controladores.

3.1.2 Separación en capas más allá del MVC

El principio más importante de la arquitectura MVC es la separación del código del programa en *tres capas*, dependiendo de su naturaleza. La lógica relacionada con los datos se incluye en el modelo, el código de la presentación en la vista y la lógica de la aplicación en el controlador.

La programación se puede simplificar si se utilizan otros patrones de diseño. De esta forma, las capas del modelo, la vista y el controlador se pueden subdividir en más capas.

3.1.2.1 Abstracción de la base de datos

La capa del modelo se puede dividir en la capa de acceso a los datos y en la capa de abstracción de la base de datos. De esta forma, las funciones que acceden a los datos no utilizan sentencias ni consultas que dependen de una base de datos, sino que utilizan otras funciones para realizar las consultas. Así, si se cambia de sistema gestor de bases de datos, solamente es necesario actualizar la capa de abstracción de la base de datos.

El listado 2-3-5 muestra un ejemplo de capa de abstracción de la base de datos y el listado 2-3-6 muestra una capa de acceso a datos específica para MySQL.

La parte del modelo correspondiente a la abstracción de la base de datos

```
<?php

function crear_conexion($servidor, $usuario, $contrasena)
{
  return mysql_connect($servidor, $usuario, $contrasena);
}

function cerrar_conexion($conexion)
{
  mysql_close($conexion);
}

function consulta_base_de_datos($consulta, $base_datos, $conexion)
{
  mysql_select_db($base_datos, $conexion);

  return mysql_query($consulta, $conexion);
}
```

```
function obtener_resultados($resultado)
{
  return mysql_fetch_array($resultado, MYSQL_ASSOC);
}
```

La parte del modelo correspondiente al acceso a los datos

```
function getTodosLosArticulos()
{
  // Conectar con la base de datos
  $conexion = crear_conexion('localhost', 'miusuario', 'micontrasena');

  // Ejecutar la consulta SQL
  $resultado = consulta_base_de_datos('SELECT fecha, titulo FROM
articulo', 'blog_db', $conexion);

  // Crear el array de elementos para la capa de la vista
  $articulos = array();
  while ($fila = obtener_resultados($resultado))
  {
    $articulos[] = $fila;
  }

  // Cerrar la conexión
  cerrar_conexion($conexion);

  return $articulos;
}
```

Como se puede comprobar, la capa de acceso a datos no contiene funciones dependientes de ningún sistema gestor de bases de datos, por lo que es independiente de la base de datos utilizada. Además, las funciones creadas en la capa de abstracción de la base de datos se pueden reutilizar en otras funciones del modelo que necesiten acceder a la base de datos.

Los ejemplos de los listados anteriores no son completos, y todavía hace falta añadir algo de código para tener una completa abstracción de la base de datos (abstraer el código SQL mediante un constructor de consultas independiente de la base de datos, añadir todas las funciones a una clase, etc.) El propósito de escribirlos nos ayuda a mirar los ejemplos de los niveles de la Arquitectura MVC.

3.1.2.2. Los elementos de la vista

La capa de la vista también puede aprovechar la separación de código. Las páginas web suelen contener elementos que se muestran de forma idéntica a lo largo de toda la aplicación: cabeceras de la página, el layout genérico, el pie de página y la navegación global. Normalmente sólo cambia el interior de la página. Por este motivo, la vista se separa en un layout y en una plantilla. Normalmente, el layout es global en toda la aplicación o al menos en un grupo de páginas. La plantilla sólo se encarga de visualizar las variables definidas en el controlador. Para que estos componentes interaccionen entre sí correctamente, es necesario añadir cierto código. Siguiendo estos principios, la parte de la vista del código anterior se puede separar en tres partes, como se muestra en los scripts siguientes.

La parte de la plantilla de la vista, en miplantilla.php

```
<h1>Listado de Artículos</h1>
<table>
<tr><th>Fecha</th><th>Título</th></tr>
<?php foreach ($articulos as $articulo): ?>
  <tr>
    <td><?php echo $articulo['fecha'] ?></td>
    <td><?php echo $articulo['titulo'] ?></td>
  </tr>
<?php endforeach; ?>
</table>
```

La parte de la lógica de la vista

```
<?php

$titulo = 'Listado de Artículos';
$contenido = include('miplantilla.php');
```

La parte del contenedor de la vista

```
<html>
  <head>
    <title><?php echo $titulo ?></title>
  </head>
  <body>
    <?php echo $contenido ?>
  </body>
</html>
```

3.1.2.3. Acciones y controlador frontal

En el ejemplo anterior, el controlador no se encargaba de realizar muchas tareas, pero en las aplicaciones web reales el controlador suele tener mucho trabajo. Una parte importante de su trabajo es común a todos los controladores de la aplicación. Entre las tareas comunes se encuentran el manejo de las peticiones del usuario, el manejo de la seguridad, cargar la configuración de la aplicación y otras tareas similares. Por este motivo, el controlador normalmente se divide en un controlador frontal, que es único para cada aplicación, y las acciones, que incluyen el código específico del controlador de cada página.

Una de las principales ventajas de utilizar un controlador frontal es que ofrece un punto de entrada único para toda la aplicación. Así, en caso de que sea necesario impedir el acceso a la aplicación, solamente es necesario editar el script correspondiente al controlador frontal. Si la aplicación no dispone de controlador frontal, se debería modificar cada uno de los controladores.

3.1.2.4. Orientación a objetos

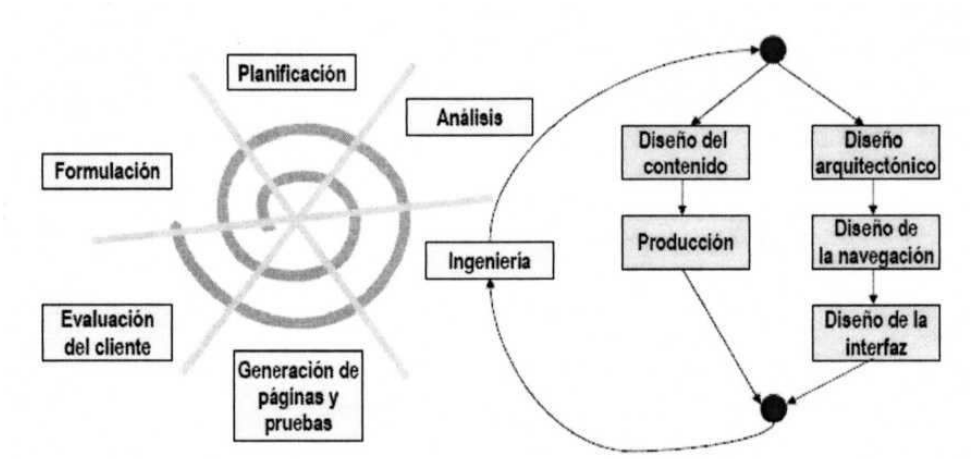
Los ejemplos anteriores utilizan la programación procedimental. Las posibilidades que ofrecen los lenguajes de programación modernos para trabajar con objetos permiten simplificar la programación, ya que los objetos pueden encapsular la lógica, pueden heredar métodos y atributos entre diferentes objetos y proporcionan una serie de convenciones claras sobre la forma de nombrar a los objetos.

La implementación de una arquitectura MVC en un lenguaje de programación que no está orientado a objetos puede encontrarse con problemas de *namespaces* y código duplicado, dificultando la lectura del código de la aplicación.

La orientación a objetos permite a los desarrolladores trabajar con objetos de la vista, objetos del controlador y clases del modelo, transformando las funciones de los ejemplos anteriores en métodos. Se trata de un requisito obligatorio para las arquitecturas de tipo MVC.

3.2 PARADIGMA DE DESARROLLO ESPIRAL ORIENTADO A LA WEB

El paradigma de desarrollo en espiral incluyo lo mejor de los paradigmas del ciclo de vida clásico y prototipeo e introdujo el concepto de evaluación del riesgo del proyecto, el cual incluía el riesgo de culminar en el tiempo programado, de no gastar más de lo presupuestado y de lograr satisfacer los requerimientos de los usuarios. Mas no se entregaba nada concreto a los usuarios sino hasta la finalización del proyecto. Es el más versátil y flexible, pero también el más complejo. Cada vuelta de la espiral (ciclo) supone una refinación en el desarrollo.



A continuación se indican las etapas que presenta este Modelo:

Formulación. Actividad que identifica las metas y los objetivos de las web y establece el ámbito de primer incremento.

Planificación. Estima el costo global del proyecto, evalúa los riesgos asociados con el esfuerzo del desarrollo bien granulado para el incremento final de la web con la determinación de objetivos, alternativas y restricciones.

Análisis. Establecimiento de los requisitos técnicos y de diseño (estéticos) e identificación de los elementos de contenido.

Ingeniería. Dos tareas paralelas.

Diseño del contenido y producción. Realizadas por personal NO técnico. Recopilación de información, medios audiovisuales, a integrar en la App.

Diseño arquitectónico, de navegación y del interfaz: hecho por técnicos.

Generación de páginas. Es una actividad de construcción que hace mucho uso de las herramientas automatizadas para la creación de la web. El contenido definido en la actividad de la ingeniería se fusiona con los diseños arquitectónicos, de navegación y de la interfaz para elaborar páginas web ejecutables en HTML, XML y otros lenguajes orientados a procesos.

Durante esta actividad se lleva a cabo la integración con el software intermedio (Middle ware) de componentes es decir (CORBA, DCOM o JAVA BEANS).

Pruebas. Se hace una navegación intensiva sobre la aplicación para descubrir errores, visualizarla en otros navegadores y ser consciente cuanto menos de las limitaciones y posibles “bugs”.

Evaluación del cliente. Cada incremento producido como parte del proceso Web se revisa durante la actividad de evaluación del cliente en este punto en donde se solicitan cambios (tiene lugar ampliaciones del ámbito). Los cambios se hacen efectivos por el flujo incremental del proceso.

3.3 UML, “Lenguaje Unificado de Modelado”

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modeling Language) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML 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.

Es importante resaltar que UML es un “lenguaje” para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. Se puede aplicar en una gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional), pero no especifica en sí mismo qué metodología o proceso usar.

UML no puede compararse con la programación estructurada, pues UML significa (Lenguaje de Modelación Unificada), no es programación, solo se diagrama la realidad de una utilización en un requerimiento. Mientras que, programación estructurada, es una forma de programar como lo es la orientación a objetos, sin embargo, la orientación a objetos viene siendo un complemento perfecto de UML, pero no por eso se toma UML sólo para lenguajes orientados a objetos.

3.3.1 Diagramas

Jerarquía de los diagramas UML 2.0, mostrados como un diagrama de clases.

En UML 2.0 hay 13 tipos diferentes de diagramas. Para comprenderlos de manera concreta, a veces es útil categorizarlos jerárquicamente, como se muestra en la figura de la derecha.

Los Diagramas de Estructura enfatizan en los elementos que deben existir en el sistema modelado:

- Diagrama de clases
- Diagrama de componentes
- Diagrama de objetos
- Diagrama de estructura compuesta (UML 2.0)
- Diagrama de despliegue
- Diagrama de paquetes

Los Diagramas de Comportamiento enfatizan en lo que debe suceder en el sistema modelado:

- Diagrama de actividades
- Diagrama de casos de uso
- Diagrama de estados

Los Diagramas de Interacción son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado:

- Diagrama de secuencia
- Diagrama de colaboración
- Diagrama de tiempos (UML 2.0)
- Diagrama de vista de interacción (UML 2.0)

Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.

3.4 Modelo oohdm (object-oriented hypermedia design model) o método de diseño de hipermedia orientado a objetos

El modelo OOHDM u Object Oriented Hypermedia Design Methodology, para diseño de aplicaciones hipermedia y para la Web, fue diseñado por D. Schwabe, G. Rossi, and S. D. J. Barbosa y es una extensión de HDM (Modelo

de diseño de Hipermedia) con orientación a objetos, que se está convirtiendo en una de las metodologías más utilizadas. Ha sido usada para diseñar diferentes tipos de aplicaciones hipermedia como galerías interactivas, presentaciones multimedia y, sobre todo, numerosos Sitios Web.

OOHDM propone el desarrollo de aplicaciones hipermedia mediante un proceso de 4 etapas:

- Diseño Conceptual
- Diseño Navegacional
- Diseño de la Interface Abstracta
- Implementación

3.4.1 Diseño Conceptual

Durante la primera fase, denominada **Diseño Conceptual,** se realiza el modelado del dominio del hiperdocumento utilizando algún método análisis orientado a objetos de Sistemas de Información, por ejemplo OMT, obteniendo un *esquema conceptual de clases* en el que, además de clases abstractas y objetos, se representan las relaciones entre ellas, incluidas las de herencia y agregación, y los correspondientes atributos (y métodos asociados a las clases.

3.4.2 Diseño Navegacional

Una vez obtenido el esquema conceptual la metodología OOHDM establece una segunda fase de Diseño Navegacional en la que se ha de definir la estructura de navegación a través del hiperdocumento mediante la realización de modelos navegacionales que representen diferentes vistas del esquema conceptual de la fase anterior. El Diseño Navegacional se expresa, también con un enfoque orientado a objetos, a través de dos tipos de esquemas o modelos: el denominado *esquema de clases navegacionales,* con las posibles vistas del hiperdocumento a través de unos tipos predefinidos de clases, llamadas navegacionales, como son los "nodos", los "enlaces", y otras clases que representan estructuras o formas alternativas de acceso a los nodos, como los "índices" y los "recorridos guiados"; y el *esquema de contexto navegacional,*

que permite la estructuración del hiperespacio de navegación en subespacios para los que se indica la información que será mostrada al usuario y los enlaces que estarán disponibles cuando se acceda a un objeto (nodo) en un contexto determinado.

3.4.3 Diseño de Interfaz Abstracta

La metodología OOHDM contempla una tercera fase de diseño, denominada Diseño de la Interface Abstracta, en la que se realiza un modelo, también orientado a objetos, para especificar la estructura y el comportamiento de la interface del sistema hipermedia con el usuario. Este modelo es abstracto y, por tanto, independiente de la implementación final del sistema. Sin embargo, se basa en las ideas que actualmente se aplican en las Interfaces Gráficas de Usuario (IGUs), por lo que como la mayor parte de entornos hipermedia comerciales trabajan con IGUs, su implantación en un entorno de este tipo debe ser una tarea sencilla.

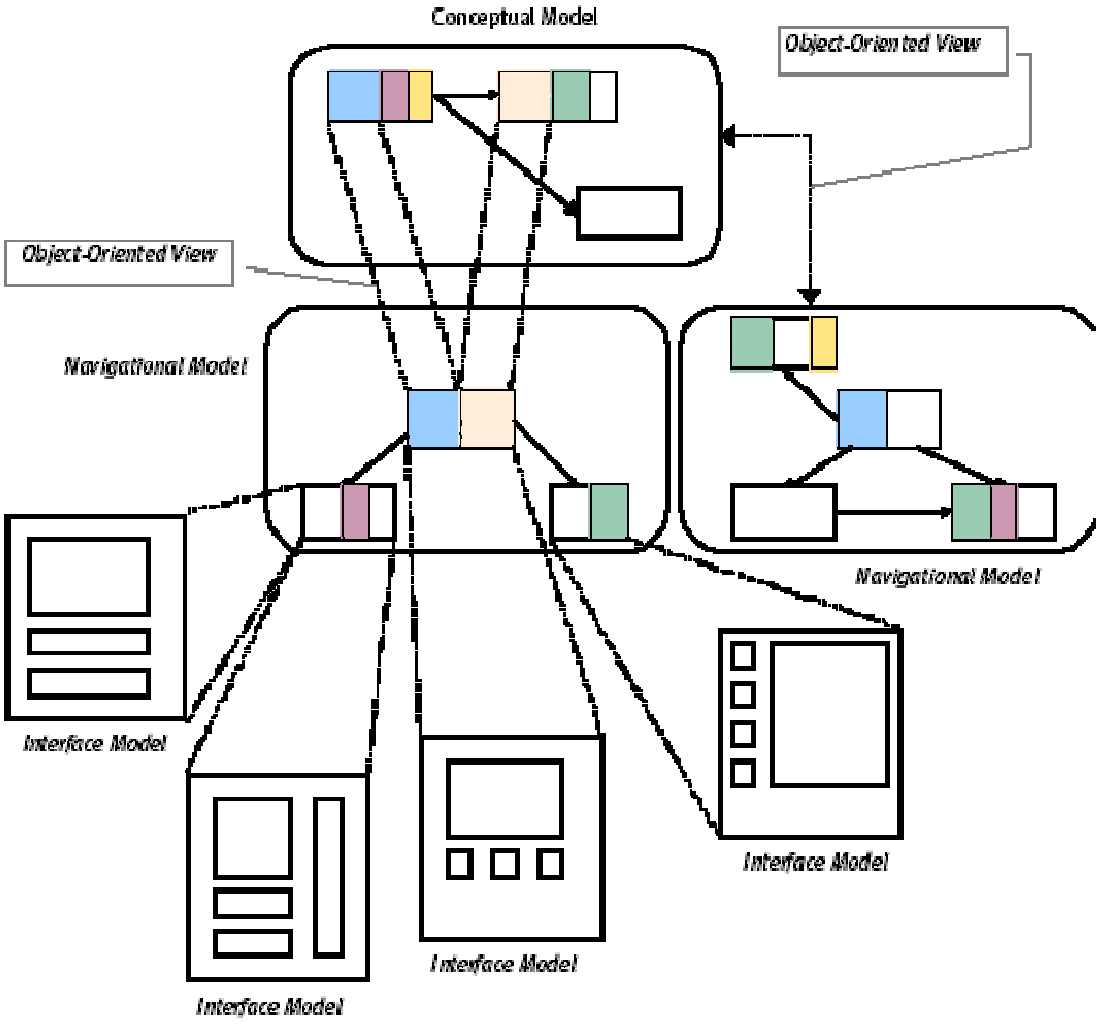
El modelo de la interface abstracta se expresa a través de tres tipos de diagramas que se complementan entre sí. En primer lugar se deben crear los denominados diagramas de Vistas de Datos Abstractos (ADVs), Diagrama de Configuración, Diagramas de Estados.

3.4.4 Implementación

En esta fase, el diseñador debe implementar el diseño. Hasta ahora, en esta fase es tenido en cuenta el entorno particular en el cual se va a correr la aplicación. Al llegar a esta fase, el primer paso que debe realizar el diseñador es definir los ítems de información que son parte del dominio del problema. Debe identificar también, cómo son organizados los ítems de acuerdo con el perfil del usuario y su tarea; decidir qué interfaz debería ver y cómo debería comportarse. A fin de implementar todo en un entorno web, el diseñador debe decidir además qué información debe ser almacenada.

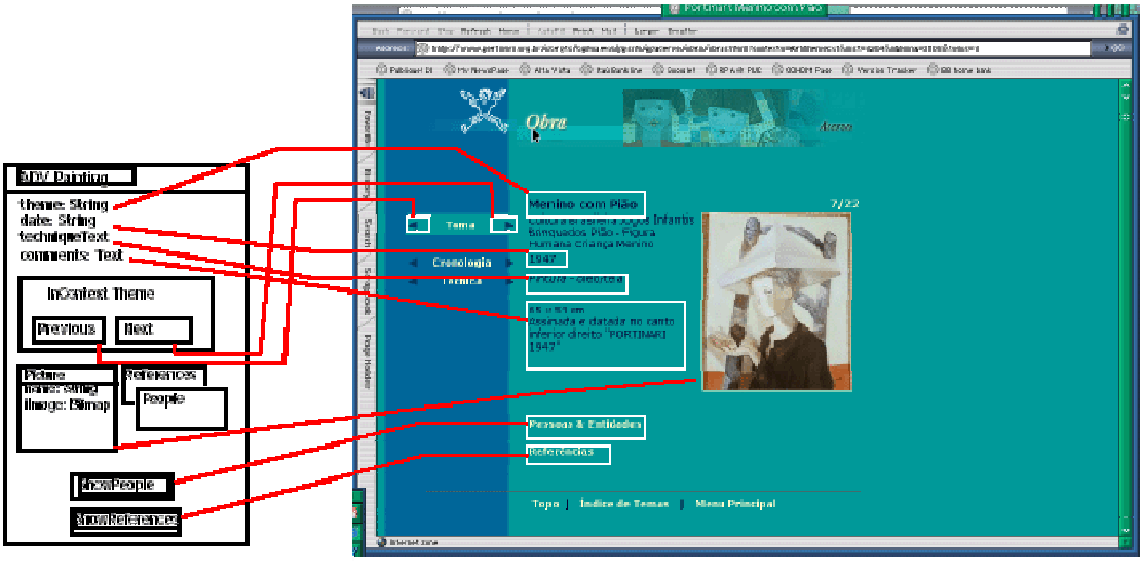
En la siguiente imagen se muestran las relaciones entre los esquemas conceptual, navegacional y los objetos de interfaz en OOHDM. (Fuente: Daniel

Schwabe y Gustavo Rossi: The Object-Oriented Hypermedia Design Model (OOHDM).



Aunque los ejemplos que ilustran el método sean siempre del mismo tipo, OOHDM es un método abierto porque, por una parte, el modelo del dominio no viene impuesto y por otra parte, el soporte en objetos del método permite la especialización de las clases navegacionales y de los contextos navegacionales. El objetivo de OOHDM es cubrir la concepción de todo tipo de aplicaciones hipermedia.

La siguiente imagen muestra una vista abstracta de datos puesta en relación con la interfaz real de objetos.



3.4 Herramientas y aplicaciones auxiliares.

Javascript: es un lenguaje sencillo pero poderosamente expresivo, su capacidad de interactuar con la ventana de navegador y con cada uno de sus componentes html nos permite manipular prácticamente cualquier componente que se desea e interactuar con los eventos del usuario.

Php: surgió para quedarse ayer convertido en una herramienta de juguete para realizar algunas tareas orientadas a la Web hasta hoy en día convertirse en un lenguaje bastante robusto orientado a objetos y base de framework muy respetables como zend, cakephp, joomla 1.5 y el mismo symfony que han marcado una nueva era del desarrollo rápido RAD basados en el Modelo Vista Controlador.

Html: ha sido y es el lenguaje de hiper texto base del contenido de una página o sitio Web es bastante práctico aunque en la red daría la impresión que su solo uso no basta para satisfacer la funcionalidad o vistosidad de los sitios modernos requeridos, por esta razón html hoy en día va acompañada de otras tecnologías como php, javascript, flash por un lado o trata de llevarselo a otro nivel a través del desarrollo de otras herramientas más interactivas como flex.

Aun así es el lenguaje por así decirlo de cajón al iniciar el desarrollo de una prpyecto Web.

Apache: es un sistema del lado del servidor que según netcraft.com ha sido el primero en el mundo del desarrollo Web sobrepasando incluso al poderoso marketing de empresas como Microsoft con el IIS. Es el más rápido, eficiente y el que evoluciona a mayor velocidad. Y Apache, por su naturaleza de software abierto, es ideal para instalar en máquinas GNU/Linux, que aseguran un S.O. con unas comunicaciones excelentes. Apache y GNU/Linux es una combinación que se está utilizando en el mundo empresarial, Apache ha ayudado a que el campo de GNU/Linux se amplíe de forma muy sólida en el mundo Internet, creando una Internet-box que difícilmente puede ser superada por otra plataforma en los sistemas actuales, tanto en coste como en potencia.

3.4.1 Herramientas de Apoyo

DreamWeaver: es la herramienta de diseño de páginas Web más avanzada, tal como se ha afirmado en muchos medios. Aunque se sea un experto programador de tecnologías orientadas a la red el usuario que lo maneje, siempre se encontrarán en este programa razones para utilizarlo, sobretodo en lo que a productividad se refiere.

Cumple perfectamente el objetivo de diseñar páginas con aspecto profesional, y soporta gran cantidad de tecnologías, además muy fáciles de usar, esta apegado a las convenciones de programación para entregar el código totalmente validado y también reconoce fortalezas y debilidades de distintos navegadores que no siempre respetan los estándares haciendo que se cree código alternativo que cuide los resultados de lo que se va a mostrar en pantalla. Dreamweaver ya no solo se trata de un editor de código sino de todo un conjunto de herramientas acopladas que rigen las mejores prácticas de desarrollo.

Editor Case Power Designer: diseñado para arquitectura empresarial y su coste muy alto se identifica con esta clase de proyectos también brinda un enfoque basado en modelos, el cual permite alinear al negocio con la tecnología de

información, facilitando la implementación de arquitecturas efectivas de información empresarial. Brinda potentes técnicas de análisis, diseño y gestión de metadatos a la empresa. En el presente proyecto sin embargo nos ha entregado las bases y la vista general de cómo queremos que nuestro proyecto actúe. Existen herramientas que pueden acoplarse con Power Designer y que permitan integrar el planteamiento con el desarrollo a través de la creación de las librerías, clases e interfaces necesarias para iniciar el desarrollo pero estas necesitan prácticamente de un desarrollo a parte que no sería viable en el presente trabajo. Es así que la herramienta ha sido usada para crear los diagramas y en base a los mismos ir creando las clases propias del proyecto.

Editor Multimedia Adobe Flash: Sin duda alguna se trata hoy en día una de las herramientas estrella en cuanto al desarrollo de aplicaciones interactivas su integración con Action Script y su fácil uso para las animaciones ha sido la clave para ser escogida en el diseño iteractivo aunque en este caso solo lo hemos usado para el desarrollo de los encabezados.

Editor de Gráficos Adobe FireWorks: es un editor sencillo y fácil uso aunque muy potente ha sido seleccionado por estas características en la manipulación de nuestros gráficos.

CAPITULO IV.- CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- El desarrollo basado en un lenguaje muy maduro como lo es php y respaldado en una base de datos potente como lo es mysql hace que la aplicación este a la altura de un proyecto de información importante como lo es Nanegal.
- Con symfony que guarda muchas de las buenas prácticas de desarrollo y sistematización de las tareas comunes a la misma creación del proyecto ha facilitado mucho el desarrollo del sitio.
- El framework symfony guarda estándares por lo que será fácil incrementar la capacidad del sistema.
- Las limitaciones en el acceso a la información de los proyectos de Nanegal ha impedido que el sistema automatice a niveles más profundos los procesos que se llevan adelante.

4.2 Recomendaciones

- El sitio al estar expuesto a la Internet podría ser víctima de un ataque por lo que es importante llevar un backup inicial y realizar respaldos periódicos del sitio o cuando se realicen actualizaciones importantes.
- Al ser un sitio enfocado en informar sobre los avances más importantes en Nanegal es recomendable que una vez que existan estándares sobre la forma de llevar los proyectos se creen los módulos correspondientes que faciliten el seguimiento y control de los mismos.
- Los usuarios que utilicen el sitio tienen distintos accesos y en un futuro cuando los proyectos tengas distintas tareas y actividades los niveles de acceso de usuarios podrían ser extendidos para que sea muy enfocados en el usuario.

Bibilografía:

http://www.hipertext.net/web/pag206.htm#Diseño

http://es.wikipedia.org/wiki/Servidor_HTTP_Apache

http://es.wikipedia.org/wiki/HTML

http://es.wikipedia.org/wiki/JavaScript

http://es.wikipedia.org/wiki/PHP

http://www.mtbase.com/productos/modelamientometadatos/powerdesigner

http://es.wikipedia.org/wiki/Adobe_Fireworks

http://es.wikipedia.org/wiki/Adobe_Flash

http://www.ati.es/gt/LATIGOO/OOp96/Ponen6/atio606.html

http://www.hipertexto.info/documentos/oohdm.htm

http://news.netcraft.com/

ANEXOS