UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SITIO EN INTERNET PARA LA DIFUSIÓN DE LA CULTURA

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN

P R E S E N T A

VERÓNICA HELADIA. SÁNCHEZ ZARCO

Director de Tesis:
Ing. Alejandro Velazquez Mena

Ciudad Universitaria junio 2005

Dedicatoria

A la memoria de mí Padre:

Amigo de mi vida todo el tiempo, protector de mi miedo, brazo mío, palabra clara, corazón resuelto.

"Padre mío, señor mío, hermano mío,
Amigo de mi alma, tierno y fuerte,
Saca tu cuerpo viejo, viejo mío,
Saca tu cuerpo de la muerte.

Saca tu corazón igual que un río,
Tu frente limpia en que aprendí a quererte,
Tu brazo como un árbol en el frío
Saca todo tu cuerpo de la muerte. "

Esta carrera, tu lo sabes papá, es tuya, por aquella promesa que te hice a los 7 años, más que mía, tuya. Te extraño.

Agradecimientos

A Dios, por darme la oportunidad de estar aquí, en este misterio de la vida.
A mí madre, por haberme hecho llegar a la vida.
A mí padre, por no haberse ido y estar conmigo a cada instante.
A mí abuela, por su ejemplo, sus enseñanzas, su fortaleza y sobre todo, por ese gran amor que siempre me ha dado.

A mis hermanas:
Luz María, por ser mí amiga y compañera durante toda la vida.
Susana, por abrazarme siempre que lo necesito, y recordarme quien soy verdaderamente.
Norma, por alentarme y cuidarme cuando lo necesite.
Yolanda, por escucharme.
A mis sobrinos: Alex y Axel, por los juegos, por la risa, por la inocencia.

A mí amiga Tania y a mí amigo Guillermo, por su ayuda "mágica" y por todo lo que vivimos.
A mí amiga Rosario por ayudarme a ser mejor persona.

A mí compañero de viaje, este viaje que es la vida, gracias Christian, por sanar mis alas, impulsarme y hacerme volver a volar. Gracias por estar conmigo.

A mí ángel, viajero en el tiempo, por la esperanza.

A mí director de tesis, Ing. Alejandro Velazquez Mena. Por su valioso apoyo y asesoría durante el desarrollo de esta tesis y a cada uno de los profesores que firmaron como sinodales, por hacer posible este proceso de titulación.

Gracias, a todas las personas que directa o indirectamente hicieron posible que yo terminara este trabajo.

"El mundo no está amenazado por las malas personas, sino por aquellos que permiten la maldad."

"Pocos son los que ven con sus propios ojos y sienten con sus propios corazones".

"Si no chocamos contra la razón nunca llegaremos a nada."

"Un ser humano es parte de un todo, llamado por nosotros universo, una parte limitada en el tiempo y el espacio. Se experimenta a sí mismo, sus pensamientos y sentimientos como algo separado del resto... algo así como una ilusión óptica de su conciencia. Esta falsa ilusión es para nosotros como una prisión que nos restringe a nuestros deseos personales y al afecto que profesamos a las pocas personas que nos rodean. Nuestra tarea debe ser el liberarnos de esta cárcel ampliando nuestro círculo de compasión para abarcar a todas las criaturas vivas y a la naturaleza en conjunto en toda su belleza"

Albert Einstein

INDICE

Introducción.

Difusión Cultural UNAM (DCUNAM) es una dependencia de la Universidad Nacional Autónoma de México, cuya principal función es la de promover y difundir las actividades culturales en los diversos recintos con los que cuenta la UNAM.

Si algo identifica a la difusión cultural de la UNAM es su responsabilidad de *formar* a la comunidad, ofreciendo actividades que complementen la formación académica tanto de los estudiantes, maestros, investigadores y empleados, como del público en general. Asume el compromiso de generar las condiciones que sirvan para estimular la imaginación creadora y la sensibilidad, de donde surgirán propuestas culturales y artísticas que se vinculen con el resto del país.

Para llevar a cabo esta función, Difusión Cultural UNAM se sirve de los diversos medios de publicidad, tales como anuncios de radio y televisión, carteles, boletines, folletos y una página Web. En la actualidad, esta página web carece de un sistema que permita el registro de usuarios y el envío de la cartelera vía correo electrónico, así como de una sección que permita al personal de DCUNAM el mantenimiento del sitio.

En la actualidad, Internet y el World Wide Web (WWW) como idea general de Internet, constituye el soporte natural para aplicaciones en red. El navegador de Internet (Microsoft Explorer o Netscape Communicator, por ejemplo) oculta a los usuarios la complejidad de las redes y los servidores, lo que lleva a éstos a pensar que Internet es simplemente un servicio con muchas y variadas posibilidades.

En nuestro caso, se pretende que Internet aparezca como un servicio de difusión de eventos culturales, a través del cual se pueda acceder a información referente a los eventos y centros de difusión cultural, así como a la cartelera actualizada, además de proporcionar la oportunidad de participar en promociones.

Este trabajo comienza con un poco de la historia de Internet, más adelante se hace una presentación de las tecnologías propuestas para la implementación de páginas Web, como las diferentes arquitecturas, algunos métodos de diseño de páginas, lenguajes de programación y protocolos. Posteriormente se realizará un análisis de cada una de estas, a través del cual se determinara cuales serán utilizadas en el desarrollo de esta tesis. Posteriormente se implementan las tecnologías elegidas para la elaboración del sitio Web de Difusión Cultural UNAM. Al final se presenta un apéndice, con el procedimiento para la instalación de TomCat y Sybase, así como un glosario de términos utilizados en este trabajo.

Para su desarrollo este trabajo se dividió en los siguientes capítulos:

En el primer capitulo se presenta una breve historia de Internet, de su origen, desarrollo y evolución, y se plantean los objetivos del presente trabajo.

En el segundo capitulo se muestran algunas de las tecnologías que están involucradas en el desarrollo de aplicaciones Web: Arquitectura Cliente/Servidor, el protocolo TCP/IP HTTP, HTML, lenguajes de programación, servidores de Bases de Datos y algunos servidores Web.

En el tercer capítulo, una vez seleccionadas las tecnologías para el desarrollo de sitios Web, lo siguiente será describir la estructura general de una aplicación Web y la problemática que se presenta en su desarrollo. Se hablará del acceso a Bases de Datos, así mismo se describirán

los aspectos importantes en el diseño de una aplicación Web, del análisis de requerimientos, se describirán también algunos lineamientos para el diseño de la interfaz de usuario.

En el cuarto capítulo, se habla de las herramientas de software necesarias, desde el manejador de bases de datos, el lenguaje SQL, el lenguaje JSP como script incrustado en el HTML, JAVA como lenguaje para implementar Servlets. Además se muestran ciertos lineamientos para el diseño de la base de datos y el acceso a ella. Por último se describen algunas metodologías para el proceso de desarrollo de sistemas.

En el quinto capítulo se presentará una aplicación Web completa, desarrollada bajo los lineamientos expuestos en los capítulos anteriores. En ella se incluye todo el código fuente así como los apéndices en los que se describen los pasos para la configuración del servidor Web, servidor de Base de Datos y Java.

CAPÍTULO 1

Antecedentes

1.1 Antecedentes

1.2 Objetivo General

1.3 Objetivos Específicos

1.4 Justificación

CAPITULO 1

• 1.1 Antecedentes

"Internet es una red de computadoras que utiliza convenciones comunes a la hora de nombrar y direccionar sistemas. Es una colección de redes independientes interconectadas; no hay nadie que sea dueño o active Internet al completo."

A principios de los años sesenta, con la aparición de las computadoras digitales de uso general, la tensión producida por la guerra fría y el acelerado crecimiento en los medios masivos de comunicación, comenzó a germinar la idea de transmitir datos a través de los primeros *mainframes* a muy largas distancias. Fue al final de 1969 cuando, bajo el auspicio de Advanced Research Projects Agency (**D.A.R.P.A**), se puso en funcionamiento el primer nodo de *ARPANET* en la University of California, Los Angeles (**UCLA**). La idea de unir a las computadoras y medios de transmisión de los datos a altas velocidades y de manera confiable, comenzó a extenderse de manera insospechada: en menos de dos décadas el *ARPANET* dejó su lugar a lo que ahora conocemos como Internet, para masificarse a tal punto, que podría compararse con otros medios de comunicación de mayor arraigo como la televisión.

En 1991, Tim Berners-Lee, en los laboratorios del European Laboratory for Particle Physics (**CERN**), saca a la luz pública una nueva forma de transmitir información escrita a través de computadoras: el hipertexto. Aunque no totalmente de su autoría, sí propuso un protocolo (HTTP) y un lenguaje de marcado (HTML) muy específicos, que permitieron el desarrollo de software orientados hacia esta aplicación. A la información que circulaba por Internet mediante este protocolo y este lenguaje se le conoce como World Wide Web, o simplemente WWW.

El crecimiento de Internet, en gran parte gracias al empuje ejercido por el World Wide Web, fue exponencial. Simplemente observemos que en 1993 había únicamente 130 sitios Web o servidores Web, y para octubre del 2000 se contabilizaban 22,287,727 sitios Web; es decir un crecimiento del 17,000% en sólo 7 años.

El mismo World Wide Web no se ha estancado en su idea original: mostrar información estática, previamente escrita; ha evolucionado hacia la generación de información a partir de una solicitud no determinada, como lo podemos observar en los motores de búsqueda, y en la banca electrónica.

Internet fue desarrollado dentro de una comunidad académica, entre universidades y centros de investigación. En estos lugares, el desarrollo de software se comporta de manera muy diferente a lo que estamos acostumbrados dentro del entorno comercial, el software no se escribe con el fin específico de obtener una ganancia económica. Se escriben programas para resolver los problemas de investigación y se comparten estos programas con los colegas para ayudarles a resolver los suyos y viceversa.

El software con el cual se erige Internet, es descendiente directo de éste software académico y de investigación, fundamentado en licencias poco restrictivas y respetuosas de la libertad de los usuarios de dicho software. Con el avance del tiempo y de Internet, esta idea no se ha perdido

ni debilitado, sino todo lo contrario, el Software Libre ha tomado importancia en casi todos los ámbitos donde el software en general es utilizado. El Software Libre ha tenido un gran desarrollo relacionado con el área de servidores: aplicaciones de servidores de nombres, servidores de HTTP, de correo electrónico, éstos han tenido muchísimo éxito tanto en el desarrollo académico como en el comercial.

Como ya se señaló, el WWW ha evolucionado de maneras inicialmente insospechadas. Originalmente el HTTP y el HTML fueron desarrollados con el objetivo de que los científicos pudieran compartir sus reportes y documentos en general con otros colegas y asistentes. Pero con la creciente popularidad, rápidamente comenzaron a aparecer nuevas formas de utilizar el WWW, en base a las distintas posibles solicitudes de un cliente permitidas por el protocolo.

Actualmente existen dos tipos de sitios Web: los que se comportan como revistas, donde uno simplemente lee la información ahí contenida; y las que se comportan como el software, donde uno hace un grupo de tareas específicas. Estas últimas se conocen como aplicaciones Web, y son desarrolladas por grupos de desarrollo de software, tal como en las aplicaciones de escritorio.

Una de las formas más populares para usar el WWW es a través de las aplicaciones Web. Es la razón de la aparición de un nuevo modelo de negocios en base a estas aplicaciones.

La creciente popularidad de las aplicaciones Web se debe a sus múltiples ventajas, entre las cuales podemos citar:

Multiplataforma: Con un solo programa, un único ejecutable, las aplicaciones pueden ser utilizadas a través de múltiples plataformas, tanto de hardware como de software.

Actualización instantánea: Debido que todos los usuarios de la aplicación hacen uso de un sólo programa que radica en el servidor, los usuarios siempre utilizarán la versión más actualizada del sistema.

Suave curva de aprendizaje: Los usuarios, como utilizan la aplicación a través de un navegador, hacen uso del sistema tal como si estuvieran navegando por Internet, por lo cual su acceso es más intuitivo.

Fácil de integrar con otros sistemas: Debido a que se basa en protocolos estándares, la información manejada por el sistema puede ser accedida con mayor facilidad por otros sistemas.

Acceso móvil: El usuario puede acceder a la aplicación con la única restricción de que cuente con un acceso a la red privada de la organización o a Internet, dependiendo de las políticas de dicha organización; puede hacerlo desde una computadora de escritorio, o una laptop, hogar u otra parte del mundo.

• 1.2 Objetivo general

Documentar y estructurar el desarrollo de una aplicación Web para DCUNAM, cuya funcionalidad será el registro de usuarios del sitio y el envío de la cartelera vía correo electrónico, utilizando para esto la metodología de desarrollo RUP y en su mayoría software libre.

• 1.3 Objetivos específicos

1.- Proporcionar un panorama de los estándares, protocolos y lenguajes de programación involucrados en el desarrollo de aplicaciones Web.

2.- Proponer una metodología de desarrollo para aplicaciones Web.

3.- Modificar totalmente el diseño de la página principal de DCUNAM. Con la utilización de herramientas como Flash 5.0 de macromedia, DreamWeaver y PhotoShop.

4.-Implementar un sistema de registro de usuarios con la finalidad de almacenar la dirección de correo electrónico, con la finalidad de enviar automáticamente la cartelera actualizada.

5.- Programar un buscador en la página principal con la finalidad de realizar búsquedas en la agenda de eventos.

6.- Elegir y justificar el uso del software necesario para realizar la aplicación.

• 1.4 Justificación

El desarrollo de aplicaciones Web crece día con día, existe gran cantidad de tecnologías que resultan abrumadoras para el desarrollador al momento de elegir cual usar y cómo hacerlo.

Por otra parte la documentación específica para el desarrollo de aplicaciones Web es muy escasa. A pesar de que existe documentación para las herramientas que pueden ser utilizadas, no existe un documento o manual que nos diga cómo integrar cada una de ellas en un desarrollo Web.

Con este documento se pretende mostrar de manera formal, la arquitectura de una aplicación Web en general, además de ciertos lineamientos para la integración de las tecnologías involucradas.

Con la construcción de este sitio Web, se apoyará a Difusión Cultural UNAM en su labor de la difusión de los eventos culturales, contribuyendo, como se menciono arriba, con la formación cultural de los universitarios, facilitándoles el acceso a la información referente a las actividades culturales que elijan, así como un acceso automático a la cartelera de los diversos eventos culturales.

CAPÍTULO 2

Marco Teórico

2.1 Arquitecturas

2.2 Protocolos

2.3 Lenguajes de Programación

2.4 Servidores

CAPITULO 2

Este capítulo trata acerca de las tecnologías y protocolos involucrados en el desarrollo de aplicaciones Web. Primero se revisaran los conceptos de la arquitectura Cliente/Servidor, definición, componentes y clasificación. Se describirán características generales de los protocolos y lenguajes que son el fundamento de las aplicaciones Web, entre ellos el HTML, y su medio de transporte, el HTTP. También se hablará acerca del lenguaje SQL, el cual es imprescindible si es que se desea desarrollar aplicaciones con acceso a bases de datos, como es nuestro caso.

El objetivo de este capítulo no es hacer una descripción minuciosa de cada tecnología, sino simplemente dar una descripción general, ya que se puede ahondar más en los temas utilizando la bibliografía señalada.

• 2.1 Arquitecturas

Arquitectura cliente/servidor

Cuando se habla de arquitectura cliente/servidor, se habla de una plataforma abierta. Ciertamente las posibilidades de igualar distintos productos, aplicaciones o componentes de distintos proveedores nos brinda la oportunidad de hacer una gran variedad de combinaciones de clientes y servidores.

Pero esta gran variedad implica también una gran cantidad de elementos a considerar y evaluar al momento de buscar una solución basada en la arquitectura cliente/servidor.

Básicamente tendríamos que responder dos preguntas.

- ¿Qué plataforma?
- ¿Qué herramientas de desarrollo elegir?

La primera pregunta tiene relación con la respuesta a cuestiones aun más específicas como pueden ser: ¿Qué plataforma cliente elegir? ¿Qué plataforma servidor? ¿Qué clase de middleware? ¿Qué administrador o servidor de base de datos? ¿Sobre qué arquitectura de computación distribuida se tendrá que montar la solución?

El segundo aspecto tiene relación con la toma de decisiones sobre el área de desarrollo y herramientas de cliente/servidor.

Aunque la ventaja de esta tecnología es la flexibilidad que tenemos al elegir entre muchas opciones, también nos obliga a tener amplios conocimientos para la integración de las mismas, dado que el desarrollo de aplicaciones cliente/servidor requiere del manejo de elementos en el área de diseño de bases de datos, comunicación entre procesos, procesamiento de transacciones, generación de GUI (interfaces gráficas de usuarios) y de Internet, con clientes y servidores distribuidos a lo largo de la Web.

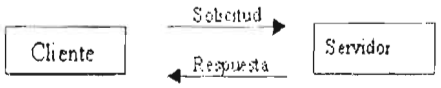
Aunque desventajas, esta tecnología ofrece beneficios en cuanto a que permite sistemas abiertos, esto mismo le da un alto grado de complejidad, característica propia de los sistemas abiertos, con una gran cantidad de componentes que no siempre tienen un compromiso firme o permanente entre sí, a diferencia de los sistemas propietarios. Si a esto se agregan las presiones comerciales de los distintos proveedores con objetivos distintos la cosa se complica

aún más. También se menciono anteriormente que se requieren conocimientos de varias áreas, y que las decisiones tomadas respecto del diseño deben ser las adecuadas, de otra manera resultan más críticas con relación a desarrollos tradicionales, lo que obliga siempre a manejarlos en forma mucho más coherente y a tener un gran dominio de herramientas para lograr una adecuada implantación y explotación.

Definición del modelo cliente/servidor

La tecnología Cliente/Servidor, desde el punto de vista funcional, se puede definir como una arquitectura distribuida que permite un procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual, múltiples clientes distribuidos geográficamente, solicitan requerimientos a uno o mas servidores centrales, teniendo de esta manera, acceso a la información en forma transparente aun en entornos multiplataforma.

En el modelo cliente/servidor, el cliente envía un mensaje solicitando un determinado servicio a un servidor, y este envía uno o varios mensajes con la respuesta. En un sistema distribuido cada máquina puede cumplir el rol de servidor para algunas tareas y el rol de cliente para otras.



Modelo cliente servidor

fig. 2.1

La idea es tratar a una computadora como un instrumento, que por sí sola pueda realizar muchas tareas, pero con la consideración de que realice aquellas que son más adecuadas a sus características. Si esto se aplica tanto a clientes como servidores se entiende que la forma más estándar de aplicación y uso de sistemas clientes/servidores es mediante la explotación de las PC a través de interfaces gráficas de usuario; mientras que la administración de datos y su seguridad e integridad se deja a cargo de computadoras centrales.

Se concluye que tanto clientes como servidores son entidades independientes que operan conjuntamente a través de una red para realizar una tarea. Pero para hacer la distinción respecto de otras formas de arquitecturas o software distribuidos, se presenta a continuación una lista de características que debieran cumplir los sistemas cliente/servidor:

- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- Las plataformas de software y hardware entre clientes y servidores son independientes. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.

- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema cliente/servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

Componentes del modelo Cliente/Servidor

Cliente/servidor es un modelo basado en la idea del servicio, en el que el cliente es un proceso consumidor de servicios y el servidor es un proceso proveedor de servicios. Además esta relación está establecida en función del intercambio de mensajes que es el único elemento de acoplamiento entre ambos. De esto se desprenden los tres elementos fundamentales sobre los cuales se desarrollan e implantan los sistemas cliente/servidor:

Cliente

El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término ***front-end***. Este normalmente maneja todas las funciones relacionadas con la manipulación y despliegue de datos, por lo que están desarrollados sobre plataformas que permiten construir interfaces gráficas de usuario (GUI), además de acceder a los servicios distribuidos en cualquier parte de la red.

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

- Administrar la interfaz de usuario.
- Interactuar con el usuario.
- Procesar la lógica de la aplicación y hacer validaciones locale
- Generar requerimientos de bases de datos.
- Recibir resultados del servidor.
- Formatear resultados.

Servidor

Es el proceso encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término ***back-end.***

El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos.

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

- Aceptar los requerimientos de bases de datos que hacen los clientes.
- Procesar requerimientos de bases de datos.

Middleware

En su definición más simple, ***middleware*** es la interfaz que provee la conectividad entre aplicaciones clientes y aplicaciones servidoras, y entre aplicaciones y bases de datos. Es una capa de software que protege a los desarrolladores de tener que manejar detalles de bajo nivel

de diferentes protocolos de comunicación, sistemas operativos y arquitecturas de bases de datos. Este tipo de interfaces incluyen API's, PRC's, Pipes, mensajería de red y accesos a bases de datos.

- El proceso cliente que es quien inicia el diálogo.
- El proceso servidor que pasivamente espera a que lleguen peticiones de servicio
- El middleware que corresponde a la interfaz que provee la conectividad entre el cliente y el servidor para poder intercambiar mensajes.

Para entender en forma más ordenada y clara los conceptos y elementos involucrados en esta tecnología se puede aplicar una descomposición o arquitectura de niveles. Esta descomposición principalmente consiste en separar los elementos estructurales de esta tecnología en función de aspectos más funcionales de la misma:

- **Nivel de Presentación:** Agrupa a todos los elementos asociados al componente Cliente.

- **Nivel de Aplicación:** Agrupa a todos los elementos asociados al componente Servidor.

- **Nivel de Comunicación:** Agrupa a todos los elementos que hacen posible la comunicación entre los componentes Cliente y servidor
- **Nivel de base de datos:** Agrupa a todas las actividades asociadas al acceso de los datos.

Este modelo de descomposición en niveles, como se verá más adelante, permite introducir más claramente la discusión del desarrollo de aplicaciones en arquitecturas de hardware y software en planos.

Clasificación de modelos Cliente/Servidor

Es importante analizar ciertas relaciones entre Cliente/ Middleware /Servidor, que pueden definir el tipo de solución que se ajusta mejor a los requerimientos de información que se obtuvieron en la etapa de análisis de un determinado proyecto.

Por lo anterior se presenta en primer lugar, un esquema de clasificación basado en los conceptos de Fat Client/Thin Client, Fat Server/Thin Server, Two Tier, Three Tier, los cuales están generalizados de definiciones, pero que se consideran necesarios y útiles para la aplicación del modelo cliente/servidor.

Este tipo de clasificación se basa en los grados de libertad que brinda el modelo cliente/servidor para balancear la carga de proceso entre los niveles de presentación, aplicación y base de datos. Dependiendo de que segmento de las capas de software tenga que soportar la mayor o menor carga de procesamiento, se habla de Fat Cliente (Thin Server) o Fat server (Thin Client). Consideraciones de este tipo son importantes al momento de decidir una plataforma de desarrollo, al punto que pueden definir la viabilidad o no de las mismas para enfrentar un cierto número de restricciones impuestas por una problemática a resolver.

Fat Client (Thin Server)

En este esquema de arquitectura el grueso de la aplicación es ejecutada en el cliente, es decir, el nivel de presentación y el nivel de aplicación corren en un único proceso cliente, y el servidor es relegado a realizar las funciones que provee un administrador de base de datos

En general este tipo de arquitectura tiene mejor aplicación en sistemas de apoyo de decisiones (DSS: Decision Support System) y sistemas de información ejecutiva (EIS: Executive Information System), y como se concluirá más adelante, tiene pocas posibilidades de aplicarse en sistemas de misión crítica

Fat Server (Thin Client)

Este es el caso opuesto al anterior, el proceso cliente es restringido a la presentación de la interfaz de usuario, mientras que el grueso de la aplicación corre por el lado del servidor de aplicación.

En general este tipo de arquitectura presenta una flexibilidad mayor como para desarrollar un gran espectro de aplicaciones, incluyendo los sistemas de misión crítica a través de servidores de transacciones.

Por planos o capas (Tier)

Una de las más comunes y discutidas distinciones entre las diferentes arquitecturas cliente/servidor se basan en la idea de planos (tier), la cual es una variación sobre la división o clasificación por tamaño de componentes (clientes grandes y servidores amplios). Esto se debe a que se trata de definir el modo en que las funciones de la aplicación serán asignadas, y en que proporción, tanto al cliente como al servidor. Dichas funciones se deben agrupar entre los tres componentes clásicos para cliente/servidor:

- Interfaz de usuario.
- Lógica de negocios.
- Datos compartidos.

Cada uno de los cuales corresponde a un plano

Dentro de esta categoría tenemos las aplicaciones en dos planos (two-tier), tres planos (three-tier) y multi planos (multi-tier). Dado que este término ha sido sobrecargado de significados por cuanto se lo utiliza indistintamente para referirse tanto a aspectos lógicos (Software) como físicos (Hardware), aquí se muestran ambos.

Planos a niveles de software

Este enfoque o clasificación es el más generalizado y el que más se ajusta a los enfoques modernos, dado que se fundamenta en los componentes lógicos de la estructura cliente/servidor. Por ejemplo, esto permite hablar de servidores de aplicación distribuidos a lo largo de una red, y no tiene mucho sentido identificar a un equipo de hardware como servidor, si no más bien entenderlo como una plataforma física sobre la cual pueden operar uno o más servidores de aplicaciones.

Cliente/Servidor Dos Planos

Esta estructura se caracteriza por la conexión directa entre el proceso cliente y un administrador de bases de datos. Dependiendo de donde se localice el grupo de tareas correspondientes a la lógica de negocios se pueden tener a su vez dos tipos distintos dentro de esta misma categoría:

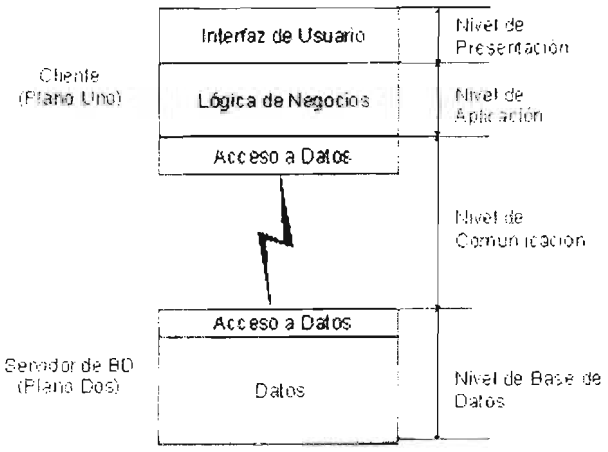


fig. 2.2

En este esquema el cliente envía mensajes con solicitudes SQL al servidor de bases de datos y el resultado de cada instrucción SQL es devuelto por la red, no importando si son uno, diez, cien o mil registros. Es el mismo cliente quien debe procesar todos los registros que le fueron devueltos por el servidor de base de datos, según el requerimiento que él mismo hizo.

Esto hace que este tipo de estructura se adecue a los requerimientos de aplicaciones orientadas a los sistemas de apoyo y gestión, pero resultan inadecuados para los sistemas críticos en que se requieran bajos tiempos de respuesta.

Ventajas

Presenta una estructura de desarrollo bastante simple por cuanto el programador típicamente maneja un único ambiente de desarrollo (es más simple respecto de cliente/servidor en tres planos, puesto que reduce una capa de programación, como se verá más adelante).

- Por su bajo rendimiento esta estructura tiene un bajo espectro de aplicación, limitándose a la construcción de sistemas no críticos.

Cliente/Servidor Tres Planos.

Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se tomen, se puede balancear la carga de trabajo entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

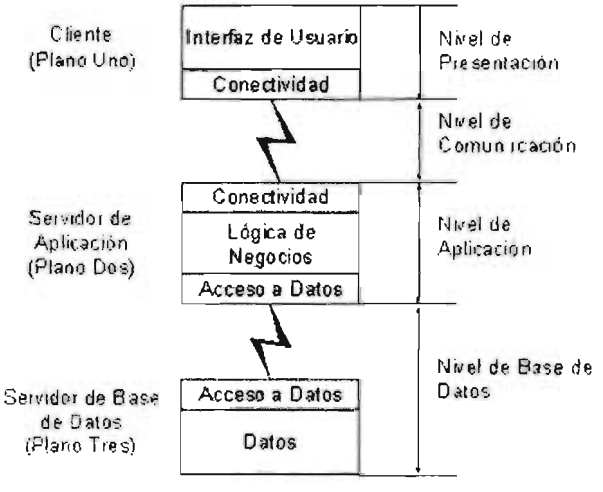


fig. 2.3

En este esquema el cliente envía mensajes directamente al servidor de aplicación el cual debe administrar y responder todas las solicitudes. Es el servidor, dependiendo del tipo de solicitud, quien accede y se conecta con la base de datos.

Ventajas

- Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto de la estructura en dos planos)
- Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones.
- Provee escalabilidad horizontal y vertical.
- Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.
- Los lenguajes sobre los cuales se desarrollan las aplicaciones son estándares lo que hace más exportables las aplicaciones entre plataformas.
- Dado que mejora el rendimiento al optimizar el flujo de información entre componentes, permite construir sistemas críticos de alta confiabilidad.
- El mismo hecho de localizar las reglas del negocio en su propio ambiente, en vez de distribuirlos en la capa de interfaz de usuario, permite reducir el impacto de hacer mantenimiento, cambios urgentes de última hora o mejoras al sistema.
- Disminuye el número de usuarios (licencias) conectados a la base de datos.

Desventajas

- Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con cliente/servidor dos planos.
- Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo cliente/servidor dos planos, y normalmente son de alto costo.

Planos a niveles de software

Este enfoque o clasificación es el más generalizado y el que más se ajusta a los enfoques modernos, dado que se fundamenta en los componentes lógicos de la estructura cliente/servidor y en la madurez y popularidad de la computación distribuida. Por ejemplo, esto permite hablar de servidores de aplicación distribuidos a lo largo de una red, y no tiene mucho sentido identificar a un equipo de hardware como servidor, si no más bien entenderlo como una plataforma física sobre la cual pueden operar uno o más servidores de aplicaciones.

Cliente/Servidor Dos Planos.

Esta estructura se caracteriza por la conexión directa entre el proceso cliente y un administrador de bases de datos. Dependiendo de donde se localice el grupo de tareas correspondientes a la lógica de negocios se pueden tener a su vez dos tipos distintos dentro de esta misma categoría:

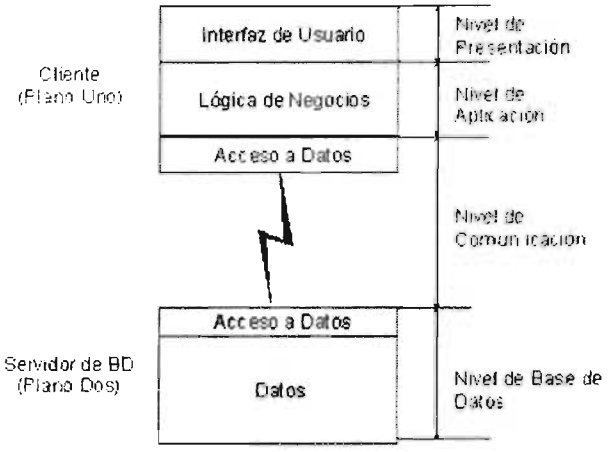


fig. 2.4

En este esquema el cliente envía mensajes con solicitudes SQL al servidor de bases de datos y el resultado de cada instrucción SQL es devuelto por la red, no importando si son uno, diez, cien o mil registros. Es el mismo cliente quien debe procesar todos los registros que le fueron devueltos por el servidor de base de datos, según el requerimiento que él mismo hizo.

Esto hace que este tipo de estructura se adecue a los requerimientos de aplicaciones orientadas a los sistemas de apoyo y gestión, pero resultan inadecuados para los sistemas críticos en que se requieran bajos tiempos de respuesta.

Ventajas

Presenta una estructura de desarrollo bastante simple por cuanto el programador típicamente maneja un único ambiente de desarrollo (es más simple respecto de cliente/servidor en tres planos, puesto que reduce una capa de programación, como se verá más adelante).

- Por su bajo rendimiento esta estructura tiene un bajo espectro de aplicación, limitándose a la construcción de sistemas no críticos.

Implementado con Procedimientos Almacenados

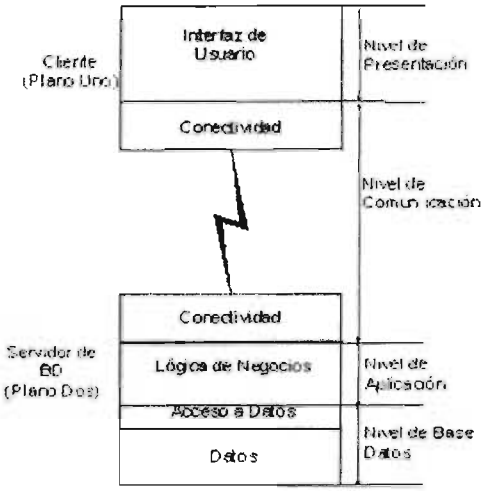


fig. 2.5

En este esquema el cliente envía llamadas a funciones que residen en la base de datos, y es ésta quien resuelve y procesa la totalidad de las instrucciones SQL agrupadas en la mencionada función.

Ventajas

- Presenta las mismas ventajas de una arquitectura dos planos con procedimientos almacenados, pero mejora considerablemente el rendimiento sobre ésta, dado que reduce el tráfico por la red al procesar los datos en la misma base de datos, haciendo viajar sólo el resultado final de un conjunto de instrucciones SQL.

Desventajas

- Si bien la complejidad de desarrollo se ve disminuida, se pierde flexibilidad y escalabilidad en las soluciones implantadas (especialmente respecto de cliente/servidor en tres planos, como se verá mas adelante).
- Obliga a basar el grueso de la aplicación en SQL extendido, propios del proveedor de la base de datos que se elija. Debiera considerarse que sí bien los procedimientos almacenados (stored procedures), los desencadenantes (triggers) y las reglas (constraint) son útiles, en rigor son ajenos al estándar de SQL:
- No existen dos implementaciones de proveedores iguales.
- El lenguaje para la descripción de los procedimientos almacenados y probablemente su funcionalidad varía de un proveedor a otro. Lo que implica que los procedimientos almacenados no son totalmente exportables entre plataformas de distintos proveedores.
- Se pierde la independencia entre el código de la aplicación (conocimiento y reglas del negocio) y los datos.
-

Cliente/Servidor Tres Planos.

Esta estructura se caracteriza por elaborar la aplicación en base a dos capas principales de software, más la capa correspondiente al servidor de base de datos. Al igual que en la arquitectura dos capas, y según las decisiones de diseño que se tomen, se puede balancear la

carga de trabajo entre el proceso cliente y el nuevo proceso correspondiente al servidor de aplicación.

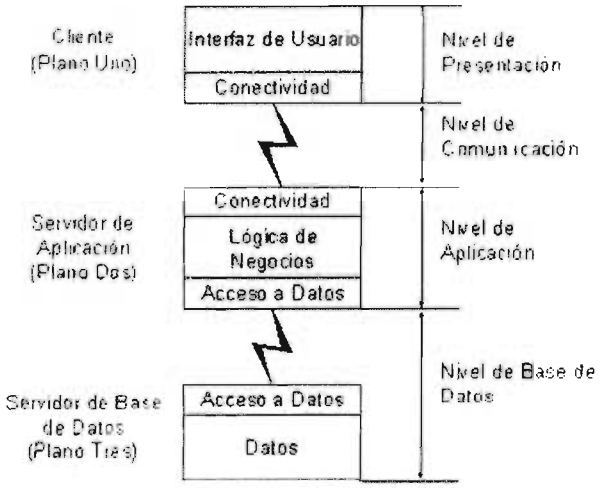


fig. 2.6

En este esquema el cliente envía mensajes directamente al servidor de aplicación el cual debe administrar y responder todas las solicitudes. Es el servidor, dependiendo del tipo de solicitud, quien accede y se conecta con la base de datos.

Ventajas

- Reduce el tráfico de información en la red por lo que mejora el rendimiento de los sistemas (especialmente respecto de la estructura en dos planos)
- Brinda una mayor flexibilidad de desarrollo y de elección de plataformas sobre la cual montar las aplicaciones.
- Provee escalabilidad horizontal y vertical.
- Se mantiene la independencia entre el código de la aplicación (reglas y conocimiento del negocio) y los datos, mejorando la portabilidad de las aplicaciones.
- Los lenguajes sobre los cuales se desarrollan las aplicaciones son estándares lo que hace más exportables las aplicaciones entre plataformas.
- Dado que mejora el rendimiento al optimizar el flujo de información entre componentes, permite construir sistemas críticos de alta confiabilidad.
- El mismo hecho de localizar las reglas del negocio en su propio ambiente, en vez de distribuirlos en la capa de interfaz de usuario, permite reducir el impacto de hacer mantenimiento, cambios urgentes de última hora o mejoras al sistema.
- Disminuye el número de usuarios (licencias) conectados a la base de datos.

Desventajas

- Dependiendo de la elección de los lenguajes de desarrollo, puede presentar mayor complejidad en comparación con cliente/servidor dos planos.
- Existen pocos proveedores de herramientas integradas de desarrollo con relación al modelo cliente/servidor dos planos, y normalmente son de alto costo.

2.2 Protocolos

Los protocolos establecen una descripción formal de los formatos que deberán presentar los mensajes para poder ser intercambiados por equipos de cómputo; además definen las reglas que ellos deben seguir para lograrlo.

Los protocolos están presentes en todas las etapas necesarias para establecer una comunicación entre equipos de cómputo, desde aquellas de más bajo nivel (e.g. la transmisión de flujos de bits a un medio físico) hasta aquellas de más alto nivel (e.g. el compartir o transferir información desde una computadora a otra en la red).

Tomando al modelo Open System Interconection **(OSI)** como referencia podemos afirmar que para cada capa o nivel que él define existen uno o más protocolos interactuando. Los protocolos son entre pares (peer-to-peer), es decir, un protocolo de algún nivel dialoga con el protocolo del mismo nivel en la computadora remota.

En esta sección se describirá el funcionamiento y características de los protocolos http y TCP/IP que son los dos protocolos necesarios para la transmisión de información vía Internet.

Nivel	**Protocolo**
Nivel físico	RJ45
Nivel de enlace	Ethernet
Nivel de red	IP , ICMP , IGMP
Nivel de transporte	TCP , UDP
Nivel de aplicación	FTP , SSH , SMTP , HTTP

Tabla 2.1

Descripción del modelo Open System interconnection (OSI)

Antes que nada, se describirán algunas características generales de OSI, con el fin de lograr un mejor entendimiento del modelo de capas y los niveles en los que trabaja cada protocolo.

El modelo de referencia OSI es la arquitectura de red predominante actualmente. El objetivo de éste es el de desarrollar estándares para la interconexión de sistemas abiertos. El término OSI es el nombre dado a un conjunto de estándares para las comunicaciones entre computadoras, terminales y redes. OSI es un modelo de 7 capas, donde cada capa define los procedimientos y las reglas (protocolos normalizados) que los subsistemas de comunicaciones deben seguir, para poder comunicarse con sus procesos correspondientes de los otros sistemas. Esto permite

que un proceso que se ejecuta en una computadora, pueda comunicarse con un proceso similar en otra computadora, si tienen implementados los mismos protocolos de comunicaciones de capas OSI.

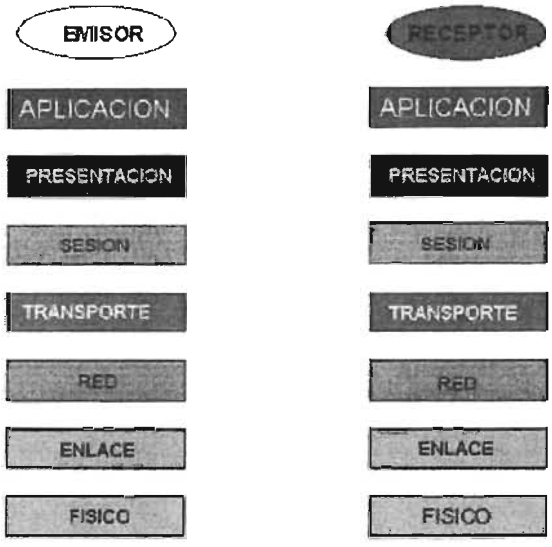


fig. 2.7

Las características generales de las capas son las siguientes

Cada una de las capas desempeña funciones bien definidas.

- Los servicios proporcionados por cada nivel son utilizados por el nivel superior.
- Existe una comunicación virtual entre 2 mismas capas, de manera horizontal.
- Existe una comunicación vertical entre una capa de nivel N y la capa de nivel N+1.
- La comunicación física se lleva a cabo entre las capas de nivel 1.

Algunas de las funciones de cada capa o nivel se describen a continuación:

Nivel de Aplicación. Relacionado con las funciones de mas alto nivel que proporcionan soporte a las aplicaciones o actividades del sistema. Por ejemplo, control de transferencia de archivos, soporte al operador funciones de dialogo de alto nivel, actividades de bases de datos de alto nivel. Los tres primeros niveles proporcionan una variedad de servicios que son empleados en la sesión de los usuarios, a este subconjunto se le denomina subsistema de la sesión de servicios.

Transferencia de archivos (FTP)

- Intercambio de mensajes (correo electrónico).
- Login remoto (rlogin, telnet).
- Acceso a bases de datos, etc.

Nivel de Presentación. Sus funciones están relacionadas con el conjunto de caracteres o códigos de datos que son usados, o la manera como van a ser presentados en pantalla o como van a ser impresos, cuando un conjunto de caracteres llega a una pantalla, se dan ciertas acciones para una presentación buena de la información. Esta capa también tiene que ver con el conjunto de caracteres que debe presentar una edición de datos, salto de línea, colocación de datos en columnas, adición de encabezados fijos para las columnas etc.

En esta capa se realizan las siguientes funciones:

- Se da formato a la información para visualizarla o imprimirla.
- Se interpretan los códigos que estén en los datos (conversión de código).
- Se gestiona la encripción de datos.
- Se realiza la compresión de datos.
- Establece una sintaxis y semántica de la información transmitida.
- Se define la estructura de los datos a transmitir (v.g. define los campos de un registro: nombre, dirección, teléfono, etc).

Define el código a usar para representar una cadena de caracteres (ASCII, EBCDIC, etc).

- Compresión de datos.
- Criptografía.

Nivel de Sesión. Estandariza el proceso de establecimiento y terminación de una sesión, si por algún motivo esta sesión falla este restaura la sesión sin perdida de datos o si esto no es posible termina la sesión de una manera ordenada chequeando y recuperando todas sus funciones. Establece las reglas o protocolos para el dialogo entre maquinas y así poder regular quien habla y por cuanto tiempo o si hablan en forma alterna es decir las reglas del dialogo que son acordadas. Provee mecanismos para organizar y estructurar diálogos entre procesos de aplicación. Actúa como un elemento moderador capaz de coordinar y controlar el intercambio de los datos. Controla la integridad y el flujo de los datos en ambos sentidos. Algunas de las funciones que realiza son las siguientes:

- Establecimiento de la conexión de sesión.
- Intercambio de datos.
- Liberación de la conexión de sesión.
- Sincronización de la sesión.
- Administración de la sesión.
- Permite a usuarios en diferentes máquinas establecer una sesión.
- Una sesión puede ser usada para efectuar un login a un sistema de tiempo compartido remoto, para transferir un archivo entre 2 máquinas, etc.
- Controla el diálogo (quién habla, cuándo, cuánto tiempo, half duplex o full duplex).
- Función de sincronización.

Nivel de Transporte. Controla la interacción entre procesos usuarios, incluye controles de integración entre usuarios de la red para prevenir perdidas o doble procesamiento de transmisiones, controla el flujo de transacciones y direccionamiento de maquinas a procesos de usuario.
Esta capa asegura que se reciban todos los datos y en el orden adecuado. Realiza un control de extremo a extremo. Algunas de las funciones realizadas son:

- Acepta los datos del nivel de sesión, fragmentándolos en unidades más pequeñas en caso necesario y los pasa al nivel de red.
- Multiplexaje.
- Regula el control de flujo del tráfico de extremo a extremo.
- Reconoce los paquetes duplicados.
- Establece conexiones punto a punto sin errores para el envío de mensajes.
- Permite multiplexar una conexión punto a punto entre diferentes procesos del usuario (puntos extremos de una conexión).
- Provee la función de difusión de mensajes (broadcast) a múltiples destinos.
- Control de Flujo.

Nivel de Red. Relaciona los circuitos virtuales, estos circuitos son imaginarios y aunque no existen figuran e interactúan con los niveles mas altos y dan la impresión de existencia en este nivel están los procedimientos de interfaces estándar para el circuito virtual y los mecanismos complejos de operación están ocultos a los niveles mas altos de software tanto como sea posible.

En esta capa se determina el establecimiento de la ruta.

- Esta capa mira las direcciones del paquete para determinar los métodos de conmutación y enrutamiento.
- Realiza control de congestión.
- Divide los mensajes de la capa de transporte en paquetes y los ensambla al final.
- Utiliza el nivel de enlace para el envío de paquetes: un paquete es encapsulado en una trama.
- Enrutamiento de paquetes.
- Envía los paquetes de nodo a nodo usando ya sea un circuito virtual o como datagramas.
- Control de Congestión.

Nivel de Enlace de Datos. Este nivel se relaciona con el envío de bloque de datos sobre una comunicación física, determina el principio y el fin de un bloque de datos transmitido, detecta errores de transmisión, controla muchas maquinas que comparten un circuito físico para que sus transmisiones no sufran mezclas, direcciona un mensaje a una maquina entre varias.

- Detección y control de errores (mediante el empleo del CRC).
- Control de secuencia
- Control de flujo
- Control de enlace lógico
- Control de acceso al medio
- Sincronización de la trama
- Estructura el flujo de bits bajo un formato predefinido llamado trama
- Para formar una trama, el nivel de enlace agrega una secuencia especial de bits al principio y al final del flujo inicial de bits
- Transfiere tramas de una forma confiable libre de errores (utiliza reconocimientos y retransmisión de tramas)
- Provee control de flujo

Nivel Físico. Relaciona la agrupación de circuitos físicos a través de los cuales los bits son movidos y que encierran las características físicas, eléctricas funcionales y procedimentales, para el envío y recepción de bits.

- Define las características físicas (componentes y conectores mecánicos).
- Define las características eléctricas (niveles de tensión).
- Define las características funcionales de la interfaz (establecimiento, mantenimiento y liberación del enlace físico).
- Solamente reconoce bits individuales, no reconoce caracteres ni tramas multicaracter. Por ejemplo RS-232 y RS-449.
- Transmisión de flujo de bits a través del medio. No existe estructura alguna.
- Maneja voltajes y pulsos eléctricos.
- Especifica cables, conectores y componentes de interfaz con el medio de transmisión.

Niveles	Nombre	Características
Nivel 7	Aplicación	Provee servicios generales relacionados con aplicaciones (Transferencias de Archivos)
Nivel 6	Presentación	Formato de datos (ASCII)
Nivel 5	Sesión	Coordina la interacción en la sesión (Dialogo) de los usuarios.
Nivel 4	Transporte	Provee una transmisión de datos confiable punto a punto
Nivel 3	Red	Enruta unidades de información
Nivel 2	Enlace de datos	Provee intercambio de datos entre dispositivos del mismo medio.
Nivel 1	Físico	Transmite un flujo de bits a través de un medio físico

Tabla 2.2

El protocolo HTTP

El Protocolo de Transferencia de Hipertexto es un protocolo de nivel de aplicación para sistemas de información distribuidos. Es un protocolo genérico y sin manejo de estados.

Descripción de su operación

El protocolo HTTP es un protocolo basado en el esquema (request/response).

El servidor envía una respuesta al cliente basado en la solicitud hecha por éste, cerrando inmediatamente la conexión entre ambos; por ello se le conoce como un protocolo sin manejo de estados; no existe el concepto de sesión como ocurre en otros protocolos.

Una conexión bajo el protocolo HTTP puede expresarse de la siguiente manera:

1. Un cliente envía una solicitud al servidor en forma de un método de solicitud, una *URL* y una versión del protocolo utilizado, seguido de un mensaje tipo MIME conteniendo los modificadores de la solicitud, la información del cliente, y un posible cuerpo de contenido a través de la conexión con el servidor.
2. El servidor responde con una línea de estado, incluyendo la versión del protocolo del mensaje y un código de éxito o error, seguido por un mensaje tipo MIME

conteniendo la información del servidor, entidades de meta información, y posible contenido cuerpo-entidad.

Mecanismo para el manejo de estados en el HTTP

El estándar HTTP/1.0 recoge únicamente tres comandos, que representan las operaciones de recepción y envío de información y chequeo de estado:

GET Se utiliza para recoger cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento correspondiente a la URL seleccionada, o bien activa un módulo CGI, que generará a su vez la información de retorno.

HEAD Solicita información sobre un objeto (fichero): tamaño, tipo, fecha de modificación. Es utilizado por los gestores de cachés de páginas o los servidores proxy, para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.

POST Sirve para enviar información al servidor, por ejemplo los datos contenidos en un formulario. El servidor pasará esta información a un proceso encargado de su tratamiento. La operación que se realiza con la información proporcionada depende de la URL utilizada. Se utiliza, sobre todo, en los formularios.

Un cliente Web selecciona automáticamente los comandos HTTP necesarios para recoger la información requerida por el usuario. Así, ante la activación de un enlace, siempre se ejecuta una operación GET para recoger el documento correspondiente. El envío del contenido de un formulario utiliza GET o POST, en función del atributo de <FORM METHOD="...">. Además, si el cliente Web tiene un caché de páginas recientemente visitadas, puede utilizar HEAD para comprobar la última fecha de modificación de un fichero, antes de traer una nueva copia del mismo.

Posteriormente se han definido algunos comandos adicionales, que sólo están disponibles en determinadas versiones de servidores HTTP, con motivos eminentemente experimentales. La última versión de HTTP, denominada 1.1, recoge estas y otras novedades, que se pueden utilizar, por ejemplo, para editar las páginas de un servidor Web trabajando en remoto.

PUT Actualiza información sobre un objeto del servidor. Es similar a POST, pero en este caso, la información enviada al servidor debe ser almacenada en la URL que acompaña al comando. Así se puede actualizar el contenido de un documento.

DELETE Elimina el documento especificado del servidor.

LINK Crea una relación entre documentos.

UNLINK Elimina una relación existente entre documentos del servidor.

Los servidores HTTP responden a cada solicitud del cliente sin relacionar tal solicitud con alguna solicitud previa o subsecuente. Para solucionar este problema Netscape introdujo el concepto de *cookie*, el cual es un grupo de cabeceras que se agrega a las solicitudes de los clientes y a las respuestas de los servidores, llevando consigo información.

Esta técnica permite a los clientes y a los servidores intercambiar información sobre el estado y colocar las solicitudes y respuestas del HTTP dentro de un contexto más largo, lo cual llamaremos *sesión*.

No obstante existen diferentes contextos potenciales y por lo tanto hay varios tipos potenciales de sesión. El diseño del mecanismo para el manejo de estados a través del intercambio de *cookies* tiene los siguientes atributos:

- Cada sesión tiene un inicio y un fin.
- Cada sesión tiene un tiempo de vida relativamente corto.

- Ya sea el agente del usuario o el servidor original puede terminar una sesión.
- La sesión está implícita en el intercambio de la información del estado.

Ante cada transacción con un servidor HTTP, éste devuelve un código numérico que informa sobre el resultado de la operación, como primera línea del mensaje de respuesta. Estos códigos aparecen en algunos casos en la pantalla del cliente, cuando se produce un error. El formato de la línea de estado es:

Existen cinco categorías de mensajes de estado, organizadas por el primer dígito del código numérico de la respuesta:

1xx: Mensajes informativos. En el HTTP/1.0) no se utilizan, y están reservados para un futuro uso.

2xx: Mensajes asociados con operaciones realizadas correctamente.

3xx: Mensajes de redirección, que informan de operaciones complementarias que se deben realizar para finalizar la operación.

4xx: Errores del cliente; el requerimiento contiene algún error, o no puede ser realizado.

5xx: Errores del servidor, que no ha podido llevar a cabo una solicitud.

Los más comunes se recogen en la siguiente tabla

Código	Comentario	Descripción
200	OK	Operación realizada satisfactoriamente.
201	Created	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. Este nuevo objeto ya está disponible. Puede ser utilizado en sistemas de edición de documentos.
202	Accepted	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. El nuevo objeto no está disponible por el momento. En el cuerpo de la respuesta se debe

		informar sobre la disponibilidad de la información.
204	No Content	La operación ha sido aceptada, pero no ha producido ningún resultado de interés. El cliente no deberá modificar el documento que está mostrando en este momento.
301	Moved Permanently	El objeto al que se accede ha sido movido a otro lugar de forma permanente. El servidor proporciona, además, la nueva URL en la variable Location de la respuesta. Algunos browsers acceden automáticamente a la nueva URL. En caso de tener capacidad, el cliente puede actualizar la URL incorrecta, por ejemplo, en la agenda de bookmarks.

Tabla 2.8

Etapas de una transacción HTTP

Para ejemplificar una transacción HTTP; a continuación se mostrará una con la descripción de cada uno de los pasos. Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo Location del cliente Web.
2. El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
4. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada (casi siempre HTTP/1.0) y un conjunto variable de información, que incluye datos sobre las capacidades del browser, datos opcionales para el servidor,...
5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
6. Se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes. En la actualidad se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo, denominado ***HTTP Keep Alive***, es empleado por la mayoría de los clientes y servidores modernos. Esta mejora es imprescindible en una Internet saturada, en la que el establecimiento de cada nueva conexión es un proceso lento y costoso.

Veamos el proceso completo con un ejemplo.

1. Desde un cliente se solicita la URL http://www.unican.es/invest/default.html
2. Se abre una conexión TCP/IP con el puerto 80 del sistema www.unican.es.
3. El cliente realiza la solicitud, enviando algo similar a esto:

Accept: text/plain Lista de tipos MIME que acepta o entiende
Accept: text/html el cliente
*Accept: audio/**
Accept: video/mpeg
.. .. ..
Accept: */* Indica que acepta otros posibles tipos MIME
User-Agent: Mozilla/3.0 (WinNT; I) Información sobre el tipo de cliente
Línea en blanco, indica el final de la petición

4. El servidor responde con la siguiente información:

HTTP/1.0 200 OK Status de la operación; en este caso, correcto
Date: Monday, 7-Oct-96 18:00:00 Fecha de la operación
Server: NCSA 1.4 Tipo y versión del servidor
MIME-version: 1.0 Versión de MIME que maneja
Content-type: text/html Definición MIME del tipo de datos a devolver
Content-length: 254 Longitud de los datos que siguen
Last-modified: 6-Oct-96 12:30:00 Fecha de modificación de los datos
Línea en blanco
<HTML> Comienzo de los datos
<HEAD><TITLE>Recursos de investigación en UNICAN</TITLE></HEAD>
<BODY>
.. .. ..
.. .. ..
</HTML>
Se cierra la conexión.

Descripción de las cabeceras

Las cabeceras son un conjunto de variables que se incluyen en los mensajes HTTP, para modificar su comportamiento o incluir información de interés. En función de su nombre, pueden aparecer en los requerimientos de un cliente, en las respuestas del servidor o en ambos tipos de mensajes. El formato general de una cabecera es:

Los nombres de variables se pueden escribir con cualquier combinación de mayúsculas y minúsculas. Además, se debe incluir un espacio en blanco entre el signo **":"** y su valor. En caso de que el valor de una variable ocupe varias líneas, éstas deberán comenzar, al menos, con un espacio en blanco o un tabulador.

Cabeceras comunes para peticiones y respuestas

Content-Type: descripción MIME de la información contenida en este mensaje. Es la referencia que utilizan las aplicaciones Web para dar el correcto tratamiento a los datos que reciben.

Content-Length: longitud en bytes de los datos enviados, expresado en base decimal.

Content-Encoding: formato de codificación de los datos enviados en este mensaje. Sirve, por ejemplo, para enviar datos comprimidos (x-gzip o x-compress) o encriptados.

Date: fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación. Por ejemplo: Sunday, 12-Dec-96 12:21:22 GMT+01. No existe un formato único en las fechas; incluso es posible encontrar casos en los que no se dispone de la zona horaria correspondiente, con los problemas de sincronización que esto produce. Los formatos de fecha a emplear están recogidos en los RFC 1036 y 1123.

Pragma: permite incluir información variada relacionada con el protocolo HTTP en el requerimiento o respuesta que se está realizando. Por ejemplo, un cliente envía un Pragma: no-cache para informar de que desea una copia nueva del recurso especificado.

Cabeceras sólo para peticiones del cliente

Accept: campo opcional que contiene una lista de tipos MIME aceptados por el cliente. Se pueden utilizar * para indicar rangos de tipos de datos; tipo/* indica todos los subtipos de un determinado medio, mientras que */* representa a cualquier tipo de dato disponible.

Authorization: clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado. La información incluye el formato de autorización empleado, seguido de la clave de acceso propiamente dicha. La explicación se incluye más adelante.

From: campo opcional que contiene la dirección de correo electrónico del usuario del cliente Web que realiza el acceso.

If-Modified-Since: permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada. Puede ser utilizada por los sistemas de almacenamiento temporal de páginas. Es equivalente a realizar un HEAD seguido de un GET normal.

Referer: contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene. No todos los clientes lo envían.

User-agent: cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los *browsers* de Netscape envían cadenas del tipo User-Agent: Mozilla/3.0 (WinNT; I)

Cabeceras sólo para respuestas del servidor HTTP

Allow: informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere esta respuesta. Por ejemplo, Allow: GET, POST.

Expires: fecha de expiración del objeto enviado. Los sistemas de cache deben descartar las posibles copias del objeto pasada esta fecha. Por ejemplo, Expires: Thu, 12 Jan 97 00:00:00 GMT+1. No todos los sistemas lo envían. Puede cambiarse utilizando un <META EXPIRES> en el encabezado de cada documento.

Location: informa sobre la dirección exacta del recurso al que se ha accedido. Cuando el servidor proporciona un código de respuesta de la serie 3xx, este parámetro contiene la URL necesaria para accesos posteriores a este recurso.

Last-modified: fecha local de modificación del objeto devuelto. Se puede corresponder con la fecha de modificación de un fichero en disco, o, para información generada dinámicamente desde una base de datos, con la fecha de modificación del registro de datos correspondiente.

Server: cadena que identifica el tipo y versión del servidor HTTP. Por ejemplo, Server: NCSA 1.4.

WWW-Autenticate: cuando se accede a un recurso protegido o de acceso restringido, el servidor devuelve un código de estado 401, y utiliza este campo para informar de los modelos de autentificación válidos para acceder a este recurso.

- **2.3 Lenguajes de Programación**

Un lenguaje de programación es un conjunto de sintaxis y reglas semánticas que definen los programas de en una computadora. Se puede entender como una técnica estándar de comunicación para entregarle instrucciones a la computadora, qué tipo de datos actúan y que acciones tomar bajo diferentes circunstancias.

Un programa escrito en un lenguaje de programación necesita pasar por un proceso de compilación, es decir, ser traducido al lenguaje de máquina para que pueda ser ejecutado por el ordenador. Existen también los llamados, lenguajes interpretados, que no necesitan ser compilados, de estos se hablará más adelante.

Lenguaje de marcado HTML

SGML y HTML

SGML significa Standard Generalized Mark-up Language, *algo así como* Lenguaje Generalizado Estándar para el Formato de Documentos *("*mark-up*" es un término de imprenta que significa* el conjunto de instrucciones estilísticas detalladas escritas en un manuscrito que debe ser tipografiado*). Es* un estándar internacional que permite definir lenguajes para dar formato a documentos *(*mark-up languages). Por ejemplo, el HTML, es un lenguaje de formato de documentos definido de acuerdo con SGML *(o en otras palabras* una aplicación de SGML*)* para dar formato a documentos *de* hipertexto.

El HTML es un lenguaje muy simple. El formato de los documentos se marca mediante etiquetas (*tags*) que indican el comienzo y el final de los elementos que componen el documento. Cada uno de estos elementos tiene un significado estructural diferente. Por ejemplo, el elemento p contiene un párrafo de texto. El comienzo del párrafo se marca con la etiqueta y el final del párrafo se marca (opcionalmente) con la etiqueta . El elemento h1 contiene un encabezado (por ejemplo, el título de un capítulo) y está delimitado por las etiquetas <h1> y </h1>. El elemento a indica un hipervínculo (o más concretamente el origen o el destino de un hipervínculo, según cómo se marque en el documento), etc.

En teoría, el código HTML sólo contiene por tanto información sobre la estructura de los contenidos. Si escribimos, por ejemplo, el siguiente párrafo:

Esto es un párrafo. Aunque haga un salto de línea, seguirá siendo un párrafo. se representará como un solo párrafo; porque así es como está marcado, como un párrafo. En tu navegador se representa así:

La simplicidad del HTML es un punto a su favor, como los contenidos están estructurados de manera lógica, pueden ser representados de acuerdo con esa estructura por cualquier navegador, según sus capacidades. Él mismo se encargará de escribir los títulos con un tipo más grande que el de los párrafos, de poner el espacio entre párrafos, de dibujar los marcadores de las listas, de dibujar las líneas entre las celdas de una tabla, etc., sin que nosotros tengamos que preocuparnos de esos aspectos.

Sin embargo, los documentos así obtenidos carecen de atractivo visual. Hoy es posible utilizar hojas de estilo para especificar la apariencia de los elementos, pero durante varios años ha sido necesario recurrir a trucos y a elementos inventados para ello: por ejemplo, el elemento "font" para cambiar la fuente de un elemento, la utilización de tablas para colocar los elementos en la pantalla en lugar de para contener datos tabulares, o la división de un mismo documento en marcos. Todo ello introdujo rápidamente problemas: el HTML se complicó y los documentos se hicieron menos accesibles: los navegadores más antiguos ya no eran capaces de entender la estructura de los nuevos documentos al estar ésta mezclada con los elementos de presentación. De hecho los documentos perdieron su estructura, lo cual era la base misma del HTML.

SGML y XML

Como ya se vio, desde su creación, el HTML ha ido aumentando su complejidad para responder a las demandas de los usuarios de la Web. Al principio era suficiente para los físicos nucleares para los que iba a servir, pero hoy los documentos HTML tienen gráficos, animaciones, música; cada día llega a tecnologías diferentes (dispositivos portátiles, teléfonos móviles) y algún día se convertirá en una Web realmente interactiva. El lenguaje de la Web necesita seguir evolucionando.

XML es una respuesta a esta necesidad. XML no es un nuevo lenguaje que vaya a suplantar a HTML. En realidad es, como el SGML, un lenguaje para definir lenguajes. Es una versión de SGML, más sencilla y más fácil de aplicar que el SGML, diseñada precisamente para hacer frente a los problemas de compatibilidad y adaptabilidad de las nuevas tecnologías a Internet.

XML significa "Extensible Mark-up Language" en XML no hay elementos. Cada usuario (o grupo de usuarios) puede crear su propio lenguaje para el formato de datos y documentos, su propio *vocabulario*, según sus necesidades, siguiendo las reglas de XML. Por ejemplo, si quieres crear una lista de libros, puedes definir tus propios elementos, encerrados entre las etiquetas correspondientes: <titulo>, <autor>, <precio>, <editorial>, etc. A partir de ahí, podrías definir una hoja de estilo para definir la presentación de cada tipo de elemento en un navegador visual o en una salida impresa; pero también podrías utilizar una aplicación (tal vez una de las muchas aplicaciones ya existentes para el manejo de documentos XML) para buscar libros por autor, para ordenarlos, etc. Si por ejemplo quisieras vender tus libros por Internet podrías utilizar estas aplicaciones para permitir a tus clientes realizar búsquedas y hacer pedidos.

La idea no es que cada usuario se cree su propio lenguaje, sino que haya estándares generales, y que se escojan los apropiados combinándolos entre sí si es necesario. La *extensibilidad* se refiere precisamente a eso. Hay muchos lenguajes definidos de acuerdo con las reglas de XML (aplicaciones XML). Por ejemplo, DocBook es un lenguaje para el formato de libros electrónicos. MathML es un lenguaje para el formato de ecuaciones matemáticas. En DocBook hay un elemento para párrafos (<Para>). En MathML hay definidos elementos útiles

para las fórmulas matemáticas, como sumatorios (<sum>), exponenciales (<exp>), etc., pero no hay un elemento para párrafos.

Esto permite crear lenguajes específicos para cada aplicación o para cada tecnología, lo cual finalmente puede simplificar las cosas. Por ejemplo, los teléfonos móviles tienen una capacidad de procesamiento mucho menor que la de los ordenadores personales. Como el HTML es demasiado complicado para ellos, se desarrolló un nuevo lenguaje simplificado llamado WAP específico para teléfonos móviles. Desgraciadamente, WAP no es una aplicación XML. En el futuro se espera que los teléfonos móviles soporten XML, pero esto va a suponer esperar un tiempo de adaptación que se podría haber evitado. Vemos aquí un ejemplo de cómo los estándares pueden ayudar a facilitar la adaptación de nuevas tecnologías a la Web.

XML y XHTML

Si con XML se pueden definir lenguajes para formato de documentos, ¿se podría definir HTML como aplicación de XML? Sí. Es más, ya lo hicieron, y lo llamaron XHTML 1.0. El XHTML 1.0 es exactamente igual que el HTML 4, excepto en que sigue las reglas de XML, que son más estrictas que las de SGML. Por ejemplo, en HTML el elemento p no necesita la etiqueta final . En XHTML sí: todos los elementos necesitan una etiqueta inicial y otra final. En HTML puedes escribir o <P>, no importa. En XHTML, las etiquetas tienen que ir obligatoriamente en minúsculas.

Por ahora veamos un poco más a detalle lo que es el HTML

HTML (*HyperText Markup Language*) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con *enlaces* (*hyperlinks*) que conducen a otros documentos o fuentes de información relacionadas, y con *inserciones* multimedia (gráficos, sonido...) La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado).

El HTML es una aplicación SGML conforme al Estándar Internacional ISO 8879, y es ampliamente considerado como el lenguaje estándar para la publicación en el World Wide Web.

El HTML, fue concebido originalmente, para ser un lenguaje para el intercambio de documentos científicos y técnicos. Por lo tanto tenía el objetivo de reducir el problema de la complejidad del SGML, especificando un pequeño conjunto de etiquetas estructurales y semánticas para escribir documentos relativamente simples.

En HTML es posible establecer tres tipos diferentes de documento:

1) **HTML 4.0 Strict:** o tipo de documento HTML 4.0 estricto, en el que tiene mayor prioridad la estructura del mismo sobre su presentación, es decir, que el documento debe estar conformado de acuerdo con las reglas del estándar HTML 4.0 de una forma estricta, sin desviarse de él en lo más mínimo.

Por ello, no se permite en el documento la presencia de elementos o atributos desaprobados por el estándar HTML 4.0 (etiquetas IMG sin el atributo ALT, definiciones de márgenes del BODY mediante atributos TOPMARGIN,etc.), y para lograr esto hace falta la declaración de estilos mediante Hojas de Estilos en Cascada (CSS) y una perfecta definición de todos los

atributos establecidos como necesarios para cada etiqueta. Como ventaja se tiene que con un documento de esta forma tenemos garantizada la accesibilidad y compatibilidad de nuestra página, y como defectos el que navegadores antiguos no lo soportan y que la construcción del documento no es flexible en absoluto (no se admiten errores en el código).

La declaración de este tipo de documentos se consigue mediante la etiqueta:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.O//EN" "http://www.w3.org/TR/REC-html40/strict.dtd">

También es posible prescindir en la etiqueta anterior de la última parte, en la que se da la URI del documento DTD, con lo que queda:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.O//EN">

2) **HTML 4. Transitional:** o tipo de documento HTML 4.0 transicional, es decir, que además de incluir los elementos y atributos del HTML 4.0 estricto añade elementos de presentación y elementos desaprobados por el Consorcio para este estándar. Con él damos apoyo a navegadores que soportan mal las Hojas de Estilos en Cascadas (CSS) y disponemos de más flexibilidad a la hora de escribir el código de nuestra página, a costa de no entrar de lleno en el estándar y de perder un poco de accesibilidad.

La etiqueta para establecer este tipo de documento es:

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.O Transitional//EN"
"http://www.w3.org/TR/REC-html40/loose.dtd">
donde es posible también usar la etiqueta abreviada:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.O Transitional//EN">

sin especificar la URI del documento de definición del Consorcio.

3) **HTML 4.0 Frameset:** que es una variación del HTML 4.0 Transitional para aquellos documentos que empleen frames. En estos documentos el elemento FRAMESET sustituye al BODY.

La declaración de este tipo se realiza mediante la etiqueta:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.O Frameset//EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">
cuya versión corta es:
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.O Frameset//EN">

Es todas estas etiquetas se utilizan las dobles barras (//) como elementos separadores de las palabras claves de la declaración, la palabra PUBLIC hace referencia al tipo de disponibilidad del documento (en nuestro caso público, por lo que está a disposición de toda la WWW), el signo - indica que las organización W3C, que es la que ha establecido el DTD, no es ISO certificada, y las letras EN indican que el sistema de caracteres en el que está escrito el **DTD** es el correspondiente a la lengua inglesa.

Estas etiquetas son usadas por las aplicaciones de validación de código HTML para saber qué tipo de validación deben aplicar, por lo que si no definimos una de ellas al principio de la página no sabrá cómo deben trabajar. En este caso, a veces no efectuará la validación, y otros tomará por defecto en valor de HTML 4.0 Transitional.

A veces hay que tener cuidado cuando especificamos la URI del documento .DTD en la declaración del DOCTYPE, ya que algunos navegadores en este caso toman siempre la declaración como HTML 4.0 estricto, cosa que no nos puede interesar, por lo que si usamos una declaración de tipo transaccional o frameset es mejor emplear la forma corta de la etiqueta.

Las hojas de estilo CSS

Uno de los objetivos iniciales del HTML es que la información descrita en éste lenguaje pudiera ser desplegada de manera independiente al dispositivo, sin que se perdiera su estructura original. Es decir, que independientemente de cómo el agente del usuario le presentara la información, ya sea de manera visual, auditiva, etc., el documento debería ser presentado sin perder el contenido que se deseaba transmitir.

Sin embargo con el éxito alcanzado con el navegador gráfico Mozilla a mediados de los noventas, Netscape Inc. comenzó a integrar nuevas características al lenguaje de manera que los autores de documentos HTML pudieran explotar las capacidades gráficas y multimedia del navegador.

Esto hizo que el espíritu inicial del HTML se comenzara a perder, hasta que el World Wide Web Consortium desarrolló el concepto de CSS. El mecanismo CSS es un mecanismo de definición de estilo, que permite a los autores y a los lectores de documentos HTML adjuntar un estilo a éstos; por ejemplo, tipo de letra, color, espaciado, etc.

Así los autores pueden adjuntar su hoja de estilo preferido, mientras que los lectores pueden tener su hoja de estilo personal que se ajuste a su discapacidad técnica o humana.

Otra característica de los CSS es su capacidad para definir objetos que indican comportamientos genéricos, que pueden ser aplicados a distintas estructuras del HTML/XHTML. Además estos objetos pueden ser sobrecargados y heredados, siguiendo un modelo orientado a objetos.

Lenguajes de servidor/cliente

Con un lenguaje del lado del cliente se pueden construir scripts (programas) que pueden acompañar a un documento HTML o estar embebidos directamente en él. El programa se ejecuta en la máquina del cliente cuando el documento se carga, o en algún otro momento, como cuando una liga es activada. El HTML soporta el manejo de scripts independientemente del lenguaje, algunos de estos lenguajes se describen a continuación.

Lenguajes del lado del Cliente

JavaScript: Al igual que Java, es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML. Al ser la más sencilla, es por el momento la más extendida.

JavaScript es similar a Java en cuanto a la sintaxis, es un lenguaje complementario que aporta algunas de las características de Java, ya que es un lenguaje basado en objetos, pero no orientado a objetos, puede reconocer y responder a varios eventos que son generados por el software del browser, incluyendo los clicks del ratón y movimientos, carga de páginas y entradas.

No es un lenguaje orientado a objetos, ya que, por ejemplo, no existe el concepto de clase. Es basado en objetos, de modo que se trabaja directamente con instancias de objetos.

Las dos principales características de JavaScript son, por un lado que es un lenguaje basado en objetos (es decir, el paradigma de programación es básicamente el de la programación dirigida a objetos, pero con menos restricciones), y por otro JavaScript es además un lenguaje orientado a eventos, debido por supuesto al tipo de entornos en los que se utiliza (Windows y sistemas X-Windows). Esto implica que gran parte de la programación en JavaScript se centra en describir objetos (con sus variables de instancia y métodos de "clase") y escribir funciones que respondan a movimientos del ratón, pulsación de teclas, apertura y cerrado de ventanas o carga de una página, entre otros eventos.

JavaScript permite la realización de aplicaciones de propósito general a través de la WWW y, aunque no está diseñado para el desarrollo de grandes aplicaciones, es suficiente para la implementación de aplicaciones WWW completas o interfaces WWW hacia otras más complejas. Los scripts JavaScript están embebidos dentro de los documentos Web y son ejecutados por el browser en varios puntos de la carga del documento. Un uso excelente de JavaScript es utilizado para realizar la comprobación de errores en forms en línea. Si se tienen formas que necesitan un programa "*server-side*" para la comprobación de errores, se puede realizar la comprobación de errores en el cliente con JavaScript. Ésto permite, por ejemplo, que una aplicación escrita en JavaScript e incrustada en un documento HTML, proporcione un mecanismo para la detección y tratamiento de eventos, como clicks del ratón o validación de entradas realizadas en formas.
Sin existir comunicación a través de la red, una página HTML con JavaScript incrustado, puede interpretar, y alertar al usuario con una ventana de diálogo, de que las entradas de los formularios no son válidas. O bien realizar algún tipo de acción como ejecutar un fichero de sonido, un Applet de Java. Otro uso es el de realizar cálculos en el cliente en vez de el servidor, reduciendo así el tráfico en la red y el tiempo de necesario para la transmisión de los datos.

VBScript: Este otro lenguaje script es una versión reducida de Visual Basic para aplicaciones (VBA). Aunque no ofrece la funcionalidad del VBA (no tiene características de Visual Basic, como las clases o las llamadas API) para que el lenguaje sea portable y seguro, viene provista de una herramienta potente y fácil de utilizar. En este lenguaje, si el programador o diseñador de las páginas tiene experiencia en VB o VBA, se encontrará con que trabajar con VBS es fácil, y es inmediatamente productivo. Aún si no se ha trabajado nunca con ninguna de estas dos herramientas, VBScript es muy fácil de aprender. Esta es la gran ventaja que tiene VBScript sobre JavaScript, ya que éste está basado en Java, y éste a su vez en C++, con lo cual, el tiempo de aprendizaje es más largo.

En cuanto a los parecidos entre VBScript y JavaScript, todo se reduce a que ambos "viven" dentro de los documentos HTML. Esto es lo único en lo que ambos lenguajes script son similares. La primera gran diferencia entre ambos es el lenguaje al que es más cercano cada uno, es decir, VBScript es código VB, y JavaScript es código Java (o C y C++) en sintaxis.

VBScript permite embeber código Visual Basic en los documentos. Cuando un browser web capacitado para entender VBScript alcanza el Script, el código se compila y se ejecuta.

Los scripts ofrecen a los autores una forma de extender los documentos HTML en formas altamente activas e interactivas:

- Los scripts puede modificar el contenido de documento de manera dinámica.

Pueden procesar el contenido de una forma HTML

- Pueden ser ejecutados al dispararse un evento específico tal como la carga y descarga de elementos, movimientos del ratón, etc.
- Pueden ligar controles de formas (botones por ejemplo) para producir elementos de GUI.

Netscape Inc. fue el primero en incluir un lenguaje de script en su navegador: el JavaScript. Posteriormente Microsoft sacó el JScript, su versión para el navegador Internet Explorer, la cual, aunque sintácticamente es afín con Java Script, cuenta con muchas incompatibilidades. Además Microsoft le agregó soporte a su navegador para VBScript, otro lenguaje de script basado en su VisualBasic.

Lenguajes del lado del servidor

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML y convertirlas en las páginas que son el resultado de dicha orden.

Cuando nosotros hacemos clic sobre un enlace hipertexto, en realidad lo que pasa es que establecemos una petición de un archivo HTML residente en el servidor (un ordenador que se encuentra continuamente conectado a la red) el cual es enviado e interpretado por nuestro navegador (el cliente), sin embargo, si la página que pedimos no es un archivo HTML, el navegador es incapaz de interpretarla y lo único que es capaz de hacer es salvarla en forma de archivo. Es por ello que, si queremos emplear lenguajes accesorios para realizar un sitio web, es absolutamente necesario que sea el propio servidor quien los ejecute e interprete para luego enviarlos al cliente (navegador) en forma de archivo HTML totalmente legible por él.

De modo que, cuando hacemos click sobre un enlace a una página que contiene un script en un lenguaje comprensible únicamente por el servidor, lo que ocurre en realidad es que dicho script es ejecutado por el servidor y el resultado de esa ejecución da lugar a la generación de un archivo HTML que es enviado al cliente.

Así pues, podemos hablar de lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.
Un lenguaje de lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo. Por otra parte, los scripts son almacenados en el servidor quien los ejecuta y traduce a HTML por lo que permanecen ocultos para el cliente. Este hecho puede resultar a todas luces una forma legítima de proteger el trabajo intelectual realizado.

CGI

Es el sistema más antiguo que existe para la programación de las páginas dinámicas de servidor. Actualmente se encuentra un poco desfasado por diversas razones entre las que destaca la dificultad con la que se desarrollan los programas y la pesada carga que supone para el servidor que los ejecuta.

Los CGI se escriben habitualmente en el lenguaje Perl, sin embargo, otros lenguajes como C, C++ o Visual Basic pueden ser también empleados para construirlos.

El CGI es un estándar para tener interfaces entre aplicaciones externas y servidores de información, tales como los servidores HTTP. Un documento HTML enviado por un servidor HTTP a un cliente es estático, el texto que contiene esta en un estado constante, no cambia. Por otro lado, un programa CGI se ejecuta en tiempo real, por lo que puede recibir información

del usuario, procesarla y arrojar datos formateados en HTML, texto, o algún otro formato plano o binario, en base a esta información.

El funcionamiento básico de un programa CGI:

- Se realiza una petición http, a la que pueden acompañar datos llegados o bien por un formulario o bien a través de la URL.

El servidor ejecuta los programas CGI a los que se accede y trabaja con los recursos necesarios para llevar a cabo las acciones, como por ejemplo bases de datos.

El programa CGI va escribiendo en la salida estándar el resultado de la ejecución del CGI, que incluye etiquetas HTML, ya que lo que se escribe es una página web.

Algunas desventajas de la programación en CGI son las siguientes:

- Los resultados se escriben directamente con el CGI, así que el código del programa se mezcla con el del HTML haciendo difícil su comprensión y mantenimiento.
- Cada programa CGI que se pone en marcha lo hace en un espacio de memoria propio. Así, si tres usuarios ponen en marcha un CGI a la vez se multiplicará por tres la cantidad de recursos que ocupe ese CGI. Esto significa una grave ineficiencia.

El CGI provee de una manera consistente para pasar datos a partir de la solicitud de un agente del usuario a un programa que puede ser localizado y ejecutado por el servidor de HTTP.

Diseño del CGI

Un programa CGI puede adquirir los datos de entrada por diferentes medios: Por medio de variables de entorno (insertadas por el servidor HTTP al momento de generar el nuevo proceso que ejecutará el programa); y por medio de la entrada estándar, este flujo de información puede estar formateada en un estilo MIME y debe ser analizada sintácticamente para dividirla en sus distintas variables. Cuando los datos se reciben por la entrada estándar, es por que la información se envió por un método GET, POST, PUT, DELETE, TRACE, HEAD del HTTP.

Cuando el servidor llama a un programa a petición del cliente, le envía información mediante variables de entorno y argumentos. Las variables de entorno las activará el programa CGI, mientras que los argumentos se pasan como parámetros de la llamada, es decir, en la misma línea de comandos.

Servlet

Los Servlets son módulos que extienden los servidores orientados a petición-respuesta, como los servidores web compatibles con Java. Por ejemplo, un servlet podría ser responsable de tomar los datos de un formulario de entrada de pedidos en HTML y aplicarle la lógica de negocios utilizada para actualizar la base de datos de pedidos de la compañía. Los Servlets son para los servidores lo que los applets son para los navegadores. Sin embargo, al contrario que los applets, los servlets no tienen interface gráfico de usuario.

Los servelts pueden ser incluidos en muchos servidores diferentes porque el API Servlet, el que se utiliza para escribir Servlets, no asume nada sobre el entorno o protocolo del servidor. Los

servlets se están utilizando ampliamente dentro de servidores HTTP; muchos servidores Web soportan el API Servlet.

Servlets en lugar de Scripts CGI's

Un servlet es una clase Java, embebida dentro del web server, y utilizada para extender la capacidad del servidor. La API de servlets provee clases e interfaces para responder a cualquier tipo de requerimientos; en particular, para las aplicaciones que corren en servidores web, la API define clases de servlet específicas para requerimientos HTTP.
No necesitan ser ejecutados como nuevos procesos dado que corren directamente en el web server. Viven entre sesiones y se puede decir que son persistentes: no es necesario crear un servlet por cada requerimiento realizado desde el cliente, sino que corren dentro de éste múltiples hilos.
Los servlets son programas Java que proveen la funcionalidad de generar dinámicamente contenidos Web. Pueden ser ejecutados a través de una línea de comando. A diferencia de los applets, no poseen restricciones en cuanto a seguridad. Tienen las propiedades de cualquier aplicación *Java* y pueden acceder a los archivos del servidor para escribir y leer, cargar clases, cambiar propiedades del sistema, etc. Del mismo modo que las aplicaciones de programas *Java*, los servlets están restringidos por los permisos del sistema.
Son cargados la primera vez que son usados, y permanecen en memoria para satisfacer futuros requerimientos. Tiene un método init, donde el programador puede inicializar el estado del *servlet*, y un método destroy para administrar los recursos que son mantenidos por el servlet.

Los Servlet son un reemplazo efectivo para los scripts CGI. Proporcionan una forma de generar documentos dinámicos que son fáciles de escribir y rápidos en ejecutarse. Los Servlets también solucionan el problema de hacer la programación del lado del servidor con APIs específicos de la plataforma: están desarrollados con el API Java Servlet, una extensión estándar de Java.

Por eso se utilizan los servlets para manejar peticiones de cliente HTTP. Por ejemplo, tener un servlet procesando datos enviados con el método POST sobre HTTP utilizando un formulario HTML, incluyendo datos del pedido o de la tarjeta de crédito. Un servlet como este podría ser parte de un sistema de procesamiento de pedidos, trabajando con bases de datos de productos e inventarios, y quizás un sistema de pago on-line.

Usos de los Servlets

Permitir la colaboración entre la gente. Un servlet puede manejar múltiples peticiones concurrentes, y puede sincronizarlas. Esto permite a los servlets soportar sistemas como conferencias on-line.

Los Servlets pueden reenviar peticiones a otros servidores y servlets. Con esto los servlets pueden ser utilizados para cargar balances desde varios servidores que reflejan el mismo contenido, y para particionar un único servicio lógico en varios servidores, de acuerdo con los tipos de tareas o la organización compartida.

Analicemos a continuación estos dos lenguajes, con lo descrito arriba de cada unos de estos lenguajes se puede ver que JSP y ASP sirven para hacer, más o menos, el mismo tipo de aplicaciones web. Sin embargo, en el fondo tienen bastantes diferencias.

Código incrustado en el HTML

Este modelo, incrusta código ejecutable dentro de un documento HTML, de esta manera, el servidor, al recibir una petición, ejecuta el código incrustado y produce un documento que contiene únicamente código HTML y es enviado al agente del usuario que realizó la petición.

Dicho de otra manera, en los modelos anteriores, el desarrollador de la aplicación Web escribía código en lenguaje de programación, que como salida, producía HTML. Esta tarea llega a ser laboriosa y muchas veces aburrida. Por lo cual se definió un nuevo modelo, donde el desarrollador no se fija de manera única en el código en algún lenguaje de programación, sino que se dedica a escribir HTML y únicamente donde se deben colocar elementos dinámicos, el desarrollador utiliza marcas que define el sistema del intérprete incrustado utilizado, y escribe el código.

Lenguajes:

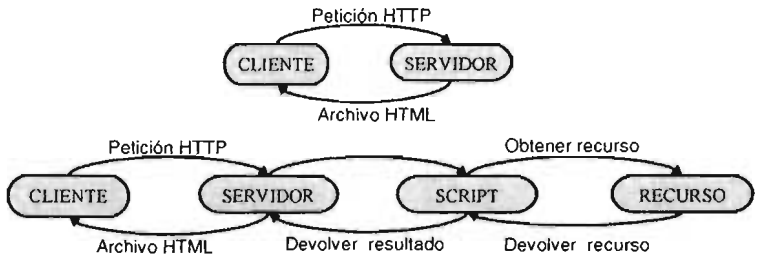


fig. 2.9

Ejemplos de estos lenguajes: VBScript (Active Server Pages –ASP- de Microsoft PHP, JSP, ASP)

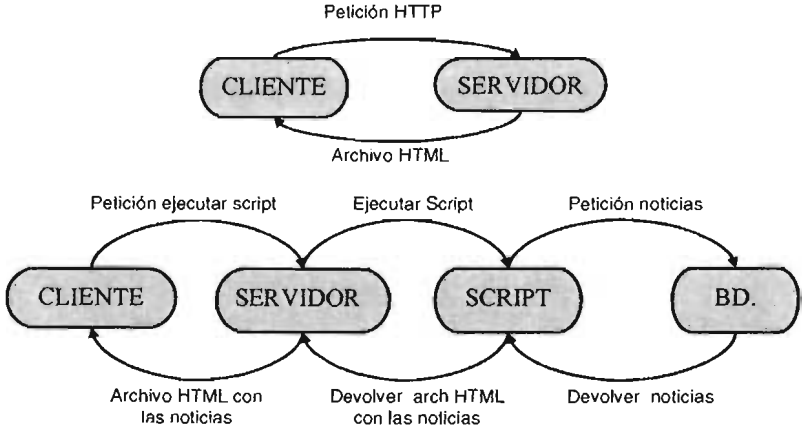


fig. 2.10

PHP

PHP es un lenguaje de programación del lado del servidor. Esto es un lenguaje que permite programar aplicaciones asociadas al servidor de Web, aumentando la funcionalidad de dicho servidor y convirtiéndolo en un sistema de desarrollo de aplicaciones cliente/servidor mucho más completo.

Ya no sólo en lo referente a acceso a bases de datos, que soporta más de 16 tipos de motores de bases de datos diferentes. Con php es muy sencillo tratar cadenas de texto o procesar ficheros, generar imágenes on-the-fly, crear documentos PDF o tratar con documentos XML y otras muchas características que lo hacen inigualable.

PHP ha tomado muchas de las mejores cualidades de otros lenguajes existentes: la versatilidad del C, los objetos de Java y la facilidad y potencia del parser de Perl. La última versión se muestra espectacularmente rápida, gracias a la incorporación del motor Zend, en algunos casos incluso de hasta un 500% más rápida que la versión anterior.

Su licencia es Open Source, corre en múltiples plataformas, Linux, AIX, SCO, casi todos los tipos de Unix, además de todas las versiones de Windows 9x, 2000 y NT

Características de PHP

Al nivel más básico, PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o mandar y recibir cookies.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

- Adabas D
- Ingres
- Oracle (OCI7 and OCI8)
- dBase
- InterBase
- PostgreSQL
- Empress
- FrontBase
- Solid
- mSQL
- Sybase
- MySQL
- Informix
- ODBC
- Unix dbm

PHP también tiene soporte para gran cantidad de estándares de red: LDAP, CORBA, FTP, IMAP, POP3, NNTP, NIS, SMNP, HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos.

Y además PHP también ofrece:

- Soporte de templates dinámicos y caché de templates.
- Multiplataforma y multiservidor web
- Soporte XML, WDDrX, XSLT, DOM
- Extensa documentación
- Soporte expresiones regulares (de perl, posix y propias)
- Generación on-the-fly de Imagenes, Flash y PDF

Con la última versión de PHP (la versión 4), también viene incorporado, además de todo lo anterior, el soporte de sesiones.

Java Server Pages (JSP)

Java es un lenguaje de programación orientado a objetos desarrollado por la compañía Sun Microsystems. Está construido a partir de lenguajes orientados a objetos anteriores, como C++, pero no pretende ser compatible con ellos sino ir mucho más lejos, añadiendo nuevas características como recolección de basura, programación multihilos y manejo de memoria a cargo del lenguaje. Java fue diseñado para que la ejecución de código a través de la red fuera segura, para lo cual fue necesario deshacerse de herramientas de C tales como los punteros. También se han eliminado aspectos que demostraron ser mejores en la teoría que en la práctica, tales como sobrecarga de operadores, que por cierto todavía está en discusión, y herencia múltiple.
La portabilidad fue otra de las claves para el desarrollo de Java, para lograr que las aplicaciones se escriban una sola vez sin la necesidad de modificarlas para que corran en diferentes plataformas. Esta independencia se alcanza tanto a nivel de código fuente (similar a C++) como a nivel de código binario. La solución adoptada fue compilar el código fuente para generar un código intermedio (*bytecodes*) igual para cualquier plataforma.
La Java Virtual Machine (**JVM**), donde reside el intérprete Java, sólo tiene que interpretarlos.
JSP provee a los desarrolladores de *web* de un entorno de desarrollo para crear contenidos dinámicos en el servidor usando plantillas HTML y XML (Lenguaje de Marcado Extensible), en código Java, encapsulando la lógica que genera el contenido de las páginas.
Cuando se ejecuta una página JSP es traducida a una clase de *Java*, la cual es compilada para obtener un servlet. Esta fase de traducción y compilación ocurre solamente cuando el archivo JSP es llamado la primera vez, o después de que ocurran cambios.

JSP es un acrónimo de Java Server Pages, que en castellano vendría a decir algo como Páginas de Servidor Java. Es, pues, una tecnología orientada a crear páginas web con programación en Java.

Con JSP podemos crear aplicaciones web que se ejecuten en variados servidores web, de múltiples plataformas, ya que Java es en esencia un lenguaje multiplataforma. Las páginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar scripts de servidor en sintaxis Java. Los ficheros JSP contienen código embebido que permite al diseñador de la página web acceder a datos desde código Java que se ejecuta en el servidor.

Con JSP se pueden crear páginas de manera parecida a como se crean con ASP o PHP (otras dos tecnologías de servidor). Se generan archivos con extensión .jsp que incluyen, dentro de la estructura de etiquetas HTML, las sentencias Java a ejecutar en el servidor. Antes de que sean funcionales los archivos, el motor JSP lleva a cabo una fase de traducción de esa página en un

servlet, implementado en un archivo class (Byte codes de Java). Esta fase de traducción se lleva a cabo habitualmente cuando se recibe la primera solicitud de la página .jsp.

JSP se implementa utilizando la tecnología Servlet. Cuando un servidor web recibe una petición de una página .jsp, la redirecciona a un proceso especial dedicado a manejar la ejecución de servlets (servlet container) llamado JSP container.

Características

Utilización de hilos Java para el manejo de las peticiones. El contenedor servlet puede ser ejecutado como parte del servidor web. Como podemos tener hilos con el mismo padre podremos compartir recursos con facilidad entre las peticiones. Soporte de componentes reutilizables, creación y utilización de JavaBeans del servidor. Los JavaBeans utilizados en páginas .jsp pueden ser utilizados en servlets, applets o aplicaciones Java. Separación entre la programación y la presentación. Los cambios realizados en el diseño de las páginas no interfieren en la lógica de la programación y viceversa.

Para poder utilizar esta tecnología es necesario un servidor web que soporte a páginas .html y código que implemente un contenedor JSP donde ejecutar las etiquetas JSP. Existen servidores web que incorporan dicha capacidad dentro de su código (Netscape Enterprise, Aplication Server 4.0, TomCat, Orion, IPlanet) así como servidores escritos íntegramente en Java (Java Web Server de Sun y Jigsaw de W3 Consortium) que dan soporte directamente a esta tecnología.

JSP y acceso a bases de datos.

El mecanismo que utiliza JSP para tener acceso a bases de datos es extendido de JAVA

- Java DataBase Connectivity (**JDBC**) es la tecnología Java que permite a las aplicaciones interactuar directamente con motores de base de datos relacionales.
- La API JDBC es una parte integral de la plataforma Java, por lo tanto no es necesario descargar ningún paquete adicional para usarla.
- JDBC provee una interface única, que independiza a las aplicaciones del motor de base de datos usado y del mecanismo de conexión.
- JDBC generaliza las funciones de acceso a datos más comunes. Los métodos están contenidos en los paquetes: **java.sql** y **java.txt**.
- Un driver JDBC es usado por la JVM para traducir las invocaciones JDBC genéricas en invocaciones que la bd propietaria entiende. Los drivers son clases Java que se cargan en ejecución.
- Existen drivers JDBC para la mayoría de los motores de base de datos más populares. Típicamente, los fabricantes de bases de datos proveen el driver JDBC para su motor, aunque también es posible encontrarlos en Internet.

Java Data Base Connectivity (JDBC)

Es una interfaz que provee comunicación con bases de datos. Consiste de un conjunto de clases e interfaces escritas en Java, que proveen una Interfaz de Programación de Aplicación (**API**) estándar para desarrolladores de herramientas de base de datos, permitiendo independizar la aplicación de la base de datos que utiliza.

La API JDBC es la interfaz natural a las abstracciones y conceptos básicos de Structured Query Language (**SQL**) permite crear conexiones, ejecutar sentencias SQL y manipular los resultados obtenidos. Conceptualmente es similar a Conectividad de Base de Datos Abierta (**ODBC**), pero ésta no es apropiada para usar directamente desde Java porque usa una interfaz en C y una traducción literal de C a Java no es deseable.
JDBC soporta dos modelos de acceso a base de datos: modelo de dos capas y modelo de tres capas. En el primer caso, la aplicación Java se comunica directamente con la base de datos mediante un controlador JDBC específico para cada Sistema de Administración de Base de Datos (**DBMS**) que se desee manipular.

En el segundo caso, los comandos son enviados a una capa intermedia de servicios, encargado de reenviar las sentencias SQL a la base de datos.
Existe un controlador, llamado puente JDBC-ODBC, que implementa las operaciones de JDBC traduciéndolas en operaciones ODBC, con lo cual se provee acceso a cualquier base de datos cuyo controlador ODBC se encuentre disponible.

Lenguaje ASP (Active Server Pages)

Con Active Server Pages (**ASP**) al igual que con JSP y PHP se pueden crear páginas Web dinámicas e interactivas. También son ficheros implementados como páginas HTML en las que se encuentra embebido código. Son ejecutados en el servidor, que realiza la tarea de interpretación del código para devolver código HTML que puede ser mostrado por cualquier navegador. Todos estos lenguajes tienen la característica de permitir el acceso a Bases de Datos.

ASP se basa en el servidor Web IIS de Microsoft, aunque es posible utilizarlo en UNIX con una herramienta específica. ASP es una característica del Microsoft Internet Information Server (**IIS**), pero dado que el script del lado del servidor solamente se dedica a construir una página HTML normal, puede utilizarla prácticamente cualquier navegador. Se puede crear un archivo ASP incluyendo un script escrito en VBScript o Jscript en un archivo HTML y luego cambiarle el nombre por la extensión ".asp". Microsoft recomienda el uso del ASP del lado del servidor más que un script del lado del cliente cuando sea posible elegir, ya que el script del lado del servidor dará como resultado una página HTML fácilmente desplegable. Los scripts del lado del cliente (por ejemplo, con JavaScript) pueden no funcionar de manera óptima en navegadores antiguos.

Una página ASP está formada por código HTML e instrucciones en un lenguaje script. El código script aparece encerrado entre las etiquetas **<%** y **%>**, o entre <SCRIPT LANGUAGE="VBScript" RUNAT="SERVER"> y </ SCRIPT>. Cuando un usuario realiza la petición de una página ASP (fichero con extensión ASP) a través de su navegador, el servidor se encarga de ejecutar el código que se encuentra entre las etiquetas de manera totalmente transparente al cliente y manteniendo igual el resto de la página. Una vez ejecutado, el servidor devuelve al cliente una página que estará formada únicamente por código HTML . ASP utiliza la tecnología ISAPI (Internet Server Aplication Program Interface). Una aplicación ISAPI está basada en librerías de enlace dinámico que se ejecutan en el mismo espacio de direcciones que el servidor web, de manera que soportan numerosas peticiones simultáneas con una sola

imagen en memoria. Cuando el servidor recibe la petición de una página ASP, se inicia la ejecución de una librería (ASP.dll) encargada, en primer lugar, de averiguar el lenguaje en que está codificada la página. Una vez determinado, pone en marcha un motor de ejecución adecuado al lenguaje que realiza la validación de las instrucciones.

Por último, se ejecutan dichas instrucciones y el cliente recibe un fichero, que únicamente contiene código HTML, generado como resultado de la unión del propio código HTML que ya contenía la página y el que resulta de la ejecución de los scripts. No siempre la ejecución de los scripts contenidos dentro de una página ASP tiene que dar lugar a código HTML, sino que pueden ser utilizados para realizar otras tareas como, por ejemplo, la actualización de registros en una base de datos.

Los objetos integrados de *ASP* son: Application, Session, Server, Request y Response. Cada uno de ellos cuenta con una serie de métodos y propiedades cuyo funcionamiento se explica brevemente a continuación.
Para que una aplicación *ASP* funcione correctamente, el servidor *Web* debe cumplir una serie de exigencias. El principal requisito es utilizar como servidor de páginas Microsoft Internet Information Server (**IIS**), que tiene instaladas las extensiones necesarias para trabajar con páginas *ASP*. La limitación de este software es que sólo funciona bajo el sistema operativo Windows NT Server, o su sustituto Windows 2000 Server. Aunque se puede instalar un servidor IIS sobre un sistema operativo *Windows* 98 o Windows NT Workstation, el servidor *Web* ve reducidas sus prestaciones y servicios. Esta opción se suele escoger sobre todo para el desarrollo de aplicaciones en un equipo local o para la realización de pruebas de ejecución. El hecho de que las páginas *ASP* no funcionen sobre un servidor que no sea ***IIS*** y que éste, a su vez, no se ejecute bajo un sistema operativo que no sea de Microsoft, les resta competitividad con respecto a otras tecnologías actuales que funcionan bajo diferentes sistemas operativos. Esta característica hace que las páginas *ASP* sean especialmente apropiadas para el desarrollo de aplicaciones *Web* para *Intranets*, donde el sistema operativo mayoritario es Microsoft Windows NT. Los requisitos necesarios para visualizar en el navegador de un equipo cliente una página *ASP* son prácticamente los mismos que para visualizar cualquier otra página *Web*, con la única exigencia de que el navegador debe estar preparado para aceptar cookies.

ASP y Bases de Datos

ASP realiza el acceso a bases de datos utilizando el modelo *ActiveX Data Objects* (***ADO***) y la tecnología de conexión con bases de datos *Open Data Base Connectivity* (***ODBC***), aunque también puede usarse ***OLEDB***. Gracias a esto, *ASP* permite trabajar fácilmente con la mayor parte de los sistemas gestores de bases de datos actuales.

ODBC proporciona una interfaz que permite a la aplicación acceder a la información contenida en las bases de datos a través de sus distintos controladores, utilizando para ello el lenguaje ***SQL*** estándar, independientemente del sistema gestor de bases de datos empleado. Los objetos ***ADO***, basados en tecnología Component Model Object (***COM***), ofrecen una serie de métodos que proporcionan un soporte adecuado para el acceso a bases de datos desde páginas *ASP*. Básicamente *ASP* utiliza siete objetos *ADO* para trabajar con bases de *datos:* Connection, RecordSet, Command, Error, Parameter, Field y Property.

La ventaja JSP sobre ASP

La tecnología JSP usa Java como lenguaje de Script mientras que ASP usa VBScript o Jscript. Java es un lenguaje más potente y escalable que los lenguajes de Script. Las páginas JSP son compilados en Servlets por lo que actúan como una puerta a todos los servicios Java de Servidor y librerías Java para aplicaciones http. Java hace el trabajo del desarrollador más fácil

p.e. ayuda a proteger el sistema contra las "caídas" mientras que las aplicaciones ASP sobre sistemas NT son más susceptibles a sufrirlas, también ayuda en el manejo de la memoria protegiendo contra fallos de memoria y el duro trabajo de buscar los fallos de perdida de punteros de memoria que pueden hacer mas lento el funcionamiento de una aplicación.
Las aplicaciones que usan JSP tienen un mantenimiento más fácil que las que usan ASP.

- Los lenguajes de Script están bien para pequeñas aplicaciones, pero no encajan bien para aplicaciones grandes. Java es más fácil de construir y mantenimientos grandes como aplicaciones modulares.
- La tecnología JSP hace mayor énfasis en los componentes que en los Scripts, esto hace que sea más fácil revisar el contenido sin que afecte a la lógica o revisar la lógica sin cambiar el contenido.
- La arquitectura EJB encapsula la lógica de p. e.: acceso a BD, seguridad, integridad transaccional y aislamiento de la aplicación.
- Debido a que la tecnología JSP es abierta y multiplataforma, los servidores web, plataformas y otros componentes pueden ser fácilmente actualizados o cambiados sin que afecte a las aplicaciones basadas en la tecnología JSP.

Las ventajas sobre utilizar la tecnología Java con respecto a la propietaria de Microsoft (ASP) son, como se ha podido ver, diversas e interesantes. Sin embargo, podemos apuntar una ventaja de la programación en ASP, pues resulta bastante más fácil de aprender que JSP, por lo menos si no se tiene una experiencia previa en programación. Esto es debido a que Java es un lenguaje muy potente, pero un poco más complicado de usar porque es orientado a objetos y la manera de escribir los programas es más rígida.

2.4 Servidores

Servidores HTTP

Un servidor Web es un programa que se ejecuta continuamente y que espera conexiones de clientes Web, solicitando información, normalmente archivos. Los servidores y los navegadores se comunican mediante el Protocolo de transferencia de hipertexto (HTTP), Los servidores Web normalmente se llaman servidores HTTP.

Un servidor Web típico necesita, un **daemon** (demonio: Un **daemon** o **demonio** es un programa que esta en ejecución constante en el "background", con poca prioridad, a la espera de una señal para enviar datos. Generalmente, uno no se da cuenta de que están funcionando, a menos que expresamente los instale o creen problemas) que entregue información bajo el protocolo HTTP, las páginas HTML. Además, si se necesitan bases de datos, es necesario otro demonio y si queremos páginas dinámicas, como se describio mas arriba en este trabajo, es necesario integrar un lenguaje de script al servidor HTTP a través de un módulo o como CGI.

Entre los principales servidores HTTP se encuentran:

- **Apache**
- **Tomcat**
- **IIS**

De los cuales se describirán a continuación algunas características.

Apache

El servidor HTTP Apache es un servidor Web poderoso y flexible. Implementa lo último en protocolos. Es altamente configurable y extensible con módulos de terceros. Puede ser personalizado escribiendo módulos a través de una API. Provee acceso completo al código fuente, y utiliza una licencia no restrictiva. Corre bajo Windows NT/9x, Netware 5.x, OS/2, y en la mayoría de las versiones de Unix, así como en varios otros sistemas operativos.

Todas estas características lo convierten en el servidor HTTP más utilizado en Internet, Apache es considerado como uno de los mejores servidores de Web utilizados en la red Internet desde hace mucho tiempo. Por lo que éste servidor puede ser pensado como uno de los mayores triunfos del software libre.

Es un servidor de Web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1)

Las siguientes pueden ser consideradas como tres de sus principales virtudes.

- Implementa los últimos protocolos, aunque se base en el HTTP / 1.1
- Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo y con la API de programación de módulos.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para solución de los mismos.

El nombre de Apache viene de "APAtCHyserver", (Un servidor lleno de remiendos). Estaba basado en alguna codificación existente y en una serie de archivos "parche".

Apache ha mostrado ser substancialmente más rápido que muchos otros servidores libres. Aunque seguro que los servidores comerciales han exigido superar la rapidez del Apache, ellos opinan que es mejor tener en su mayor parte un servidor libre rápido, que un servidor extremadamente rápido pero que cueste miles de dólares. Apache funciona en sitios que tienen millones de usos al día, y estas se ejecutan sin complicaciones.

Iplanet

El iPlanet, es un servidor Web diseñado para desplegar los sitios y aplicaciones Web de gran tamaño. La plataforma iPlanet, ofrece el rendimiento más rápido en la ejecución de las aplicaciones basadas en tecnología Java, acelera la disponibilidad de contenidos Web dinámicos. iPlanet ayuda a reducir el costo total de propiedad que supone tener un servidor Web, gracias a sus bajos costos de mantenimiento y su facilidad de gestión.

iPlanet Web Server es compatible con múltiples plataformas, bases de datos y tipos de documentos.

iPlanet soporta entornos multiproceso y multi-threaded, que incorpora un nuevo motor servlet integrado, soporta extensiones Java Servlet 2.1 y JavaVMs de forma nativa. Según las pruebas realizadas por Mindcraft, las nuevas características proporcionan un rendimiento de Java Servlet un 222% mayor que las versiones comerciales de Apache Web Server y Stronghold 2.4.2 y un 22% mejor frente a CGI scripts.

Este servidor tiene capacidad para manejar el crecimiento exponencial del tráfico en la red, a la vez que mantiene la disponibilidad las 24 horas del día los siete días de la semana y reduce las

caídas del sistema. Si una aplicación Web se bloquea, sólo se detiene uno de los procesos del servidor Web, mientras los demás se mantienen para que los clientes puedan acceder al sitio Web. Mientras tanto, los monitores de servidor reinician automáticamente el proceso que ha fallado sin ninguna intervención del administrador de red. Además, dispone de una nueva función de rotación de registro dinámica, para que el administrador pueda rotar los accesos al servidor sin apagarlo, lo cual mejora la disponibilidad del sitio Web durante más tiempo.

Este servidor Web está estandarizando sobre la plataforma de aplicaciones Java y soporta componentes de Java 2 Enterprise Edition, tales como servlets Java y JSP's para proporcionar un entorno optimizado para el desarrollo de aplicaciones Web y para un mayor retorno de las inversiones.

iPlanet Web Server soporta estándares abiertos como HTTP 1.1, SSL 3.0, LDAP v3.0 y SNMP. Funciona bajo los siguientes sistemas: Intel Windows NT Workstations, Server, 4.0 SP4, Sun Solaris 2.6, Solaris 7, Hewlett Packard HP-UX 11.0, IBM AIX, Tru 64 Unix, SGI e IRIX.

Tomcat

Tomcat es un contenedor de servlets. El contenedor es parte del servidor web o del servidor de aplicaciones y se encarga de realzar todo el trabajo de las conexiones, decodificación de mime y gestionar la vida del servlet. Un contenedor es un shell de ejecución que maneja e invoca servlets por cuenta del usuario. Podemos dividir los contenedores de Servlest en:

Contenedores de Servlets Stand-Alone (Independientes):

Estos son una parte integral del servidor Web. Este es el caso cuando se utiliza un servidor web basado en Java, por ejemplo, el contenedor de servlets es parte de JavaWebServer. En este el contenedor por default usado es tomcat. Sin embargo, la mayoría de los servidores, no están basados en java, los que nos llevan a los dos siguientes tipos de contenidotes:

Contenedores de servlets dentro de proceso:

Aquí el contenedor servlet es una combinación de un plugin para el servidor web y una implementación de contenedor java. El plugin del servidor web abre una JVM dentro del espacio de direcciones del servidor web y permite que el contenedor java se ejecute en el. Si una cierta petición debería ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor java. Un contenedor de este tipo es adecuado para servidores multi-thread de un solo proceso y proporciona un buen rendimiento pero esta limitado en escalabilidad.

Contenedores de Servlets fuera de proceso

El contenedor servlet es una combinación de un plugin para el servidor web y una implementación de contenedor java que se ejecuta en un JVM fuera del servidor web. El plugin del servidor web y el JVM del contenedor java se comunican usando algún mecanismo IPC(normalmente sockets TCP/IP). Si una cierta petición debería ejecutar un servlet, el plugin toma el control sobre la petición y lo pasa al contenedor Java (usando IPCs). El tiempo de respuesta en este tipo de contenedores no es tan bueno como el anterior, pero obtiene mejores rendimientos en otras cosas (escalabilidad, estabilidad, etc).

Tomcat puede utilizarse como un contenedor solitario (principalmente para desarrollo y depuración) o como plugin para un servidor web existente. Esto significa que siempre que despleguemos Tomcat, tendremos que decidir como usarlo, y, si seleccionamos las opciones 2 o 3, también necesitaremos instalar un adaptador de servidor web.

Servidores de Bases De Datos

La creación de aplicaciones cliente/servidor está asociada a la utilización de servidores de bases de datos relacionales SQL, y dependiendo de los requerimientos y restricciones se debe elegir entre una arquitectura dos o tres planos. Pero una arquitectura centrada en un servidor de bases de datos, cualquiera de las modalidades de planos, permite que un proceso cliente solicite datos y servicios directamente a un servidor de bases de datos. Según las distintas normas de SQL (SQL-89, SQL-92, SQL3) el servidor debe proveer un acceso compartido a los datos con los mecanismos de protección necesarios, así como proveer mecanismos para seleccionar resultados dentro de un conjunto de datos, posibilitando un ahorro en procesos de comunicación. El servidor debe también proveer mecanismos de concurrencia, seguridad y consistencia de datos, basado principalmente en el concepto de transacción en el que todo se realiza, y por lo tanto se hace permanente, o, todo falla, anulándose la transacción en tal caso.

Los servidores de bases de datos surgen con motivo de la necesidad de las empresas de manejar grandes y complejos volúmenes de datos, al tiempo que requieren compartir la información con un conjunto de clientes (que pueden ser tanto aplicaciones como usuarios) de una manera segura.

Ante este enfoque, un sistema manejador de bases de datos (**DBMS**) deberá ofrecer soluciones de forma fiable, rentable y de alto rendimiento. A estas tres características, le debemos añadir una más: debe proporcionar servicios de forma global y, en la medida de lo posible, independientemente de la plataforma. Internet se ha convertido en nuestros días en la mayor plataforma de sistemas.

Aunque parece clara la función de un DBMS, en la actualidad cada vez más filosofías y tecnologías tienden a confluir en un mismo punto. Ya se está hablando acerca de las posibilidades de los nuevos DBMS de poder almacenar contenidos multimedia, objetos, documentos complejos. La explosión de nuevos servicios ha hecho que cada vez más aplicaciones dependan de estos servidores de datos, delegando la responsabilidad de la gestión y almacenamiento de la información a aquellos que mejor están preparados para su tratamiento.

Definición de Bases De Datos

Es una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los Sistemas de Información de una empresa o negocio en particular. Las bases de datos proporcionan la infraestructura requerida para los Sistemas de Apoyo a la Toma de Decisiones y para los Sistemas de Información Estratégicos, ya que estos sistemas explotan la información contenida en las bases de datos de la organización para apoyar el proceso de toma de decisiones o para lograr ventajas competitivas.
Los Sistemas Transaccionales o los Sistemas Estratégicos (SIS) son los encargados de recolectar la información que contendrá la base de datos, por medio de las funciones de creación, bajas o modificación de la información. La forma de operar de estos sistemas puede ser batch o en línea. Los Sistemas de Bases de Datos tiene cuatro componentes principales: datos, hardware, software y usuarios.

Ventajas en el uso de Bases De Datos

La utilización de bases de datos como plataforma para el desarrollo de Sistemas de Aplicación en las Organizaciones se ha incrementado notablemente en los últimos años, se debe a las ventajas que ofrece su utilización, algunas de las cuales se comentarán a continuación:

- Globalización de la información: permite a los diferentes usuarios considerar la información como un recurso corporativo que carece de dueños específicos.

- Eliminación de información inconsistente: si existen dos o más archivos con la misma información, los cambios que se hagan a éstos las copias del archivo de facturas.
- Permite compartir información
- Permite mantener la integridad en la información: la integridad de la información es una de sus cualidades altamente deseable y tiene por objetivo que sólo se almacena la información correcta.
- Independencia de datos: el concepto de independencia de datos es quizás el que más ha ayudado a la rápida proliferación del desarrollo de Sistemas de Bases de Datos. La independencia de datos implica un divorcio entre programas y datos. Deberán hacerse a todas.

Acceso a base de datos

Gran parte de las aplicaciones Web sirven como interfaz hacia un almacén de datos, ya que permiten obtener y escribir datos de una manera intuitiva. Para aplicaciones pequeñas que no requieran de gran cantidad de procesamiento de datos, es viable almacenarlos en estructuras específicas que se pueden bloquear y desbloquear según las necesidades de la aplicación.

Sin embargo, para aplicaciones que requieren manipulación compleja de los datos, control de acceso concurrente de usuarios, mayor estabilidad, control de integridad de datos, entre otros, es imprescindible el uso de un manejador de bases de datos o DBMS con soporte para SQL.

Según la definición de Elmasri y Navathe en *Fundametals of Database Systems*, un DBMS es "*un sistema de software de propósito general que facilita el proceso de definir, construir, y manipular bases de datos para diferentes aplicaciones*".

Características generales de un DBMS

Los datos se guardan en relaciones (o tablas). Estas tablas deben estar organizadas de manera que cada una de sus columnas pueda ser identificada por su nombre, la manera en la que los renglones estén organizados no es relevante.

Las operaciones deben ser relacionales. Las operaciones que el sistema provea, deben de generar nuevas tablas a partir de otras existentes. Por ejemplo:*SELECT nombre, id FROM customer* debe de crear una nueva tabla como resultado (la nueva tabla no tiene nombre) y ésta es un subconjunto de la tabla existente

- El DDL (Data Description Language) o Lenguaje de Definición de Datos, el cual contiene órdenes para crear, modificar o eliminar estructuras de datos como tablas, bases de datos, índices; en las que se almacenan los datos.
  Las instrucciones correspondientes son:
  CREATE, ALTER, DROP, RENAME

- Por otra parte se encuentra en DCL (Data Control Language) o Lenguaje de Control de Datos, que incluye elementos para trabajar con el entorno multiusuario en el que cobra importancia la protección de los datos así como la seguridad de las tablas y el establecimiento de privilegios en el acceso. Siendo sus comandos:
  GRANT, REVOKE

- El Lenguaje de Manipulación de Datos o DML (Data Manipulation Language), el cual permite insertar, modificar y eliminar datos de las tablas de la base de datos.

  SELECT, INSERT, UPDATE, DELETE

Procedimientos Almacenados

Una de las posibilidades de implementar de mejor forma un sistema cliente/servidor en dos planos (two-tier), sin olvidar todas sus restricciones y limitaciones, es a través de procedimientos almacenados, que son funciones que agrupan un conjunto de instrucciones y lógica de procedimientos SQL, los cuales son compilados y almacenados en la misma base. Ya se hizo notar que con estos mecanismos los proveedores de bases de datos introducen la lógica de negocios dentro de su servidor propietario; lo lógico sería mantener una independencia entre datos y códigos.

El rol principal de los procedimientos almacenados es proveer la parte servidora de la lógica de una aplicación cliente/servidor, es decir vendría a reemplazar al servidor de aplicaciones en una arquitectura tres planos (three-tier). Con todo, si bien los procedimientos almacenados permiten a un servidor brindar servicios aptos para OLTP (On Line Transaction Proccesing), no se compara con los rendimientos alcanzados por una arquitectura en tres planos con TP pesado.

Desencadenantes: Son mecanismos que permiten realizar acciones automáticamente sobre los datos, las cuales están asociadas a algún evento definido. Normalmente son implementados a través de procedimientos almacenados. Los eventos a los cuales se hace referencia están asociados a las actualizaciones de tablas mediante sentencias delete, insert o update, y son llamados implícitamente al suceder cualquiera de estos eventos, a diferencia de los procedimientos almacenados que son llamados explícitamente por un proceso cliente.

Restricciones: Al igual que los desencadenantes, son acciones que se realizan asociadas a algún evento determinado y están orientadas a llevar a cabo validaciones más simples de datos. Los tipos de eventos son los mismos que para los desencadenantes.

El sistema manejador de Base De Datos (DBMS).

Es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Una de las ventajas del DBMS es que puede ser invocado desde programas de aplicación que pertenecen a Sistemas

Transaccionales escritos en algún lenguaje de alto nivel, para la creación o actualización de las bases de datos, o bien para efectos de consulta a través de lenguajes propios tienen las bases de datos o lenguajes de cuarta generación.

Tipos de Modelos de Datos: Existen fundamentalmente tres alternativas disponibles para diseñar las bases de datos: el modelo jerárquico, el modelo de red y el modelo relacional.
Modelo Jerárquico: Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos.
Modelo de Red: Este modelo permite la representación de muchos a muchos, de tal forma que cualquier registro dentro de la base de datos puede tener varias ocurrencias superiores a él. El modelo de red evita redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector.
Modelo Relacional: Los principios del modelo relacional fueron planteados por primera vez por el Dr. E. F. Codd en Junio de 1970. Una base de datos relacional usa relaciones o tablas de dos dimensiones para almacenar información.
Tabla: Dentro del enfoque relacional una tabla es conocida como una Relación: Una relación es una tabla de dos dimensiones con las siguientes propiedades:
Cada columna contiene valores relativos al mismo atributo, y cada valor de una columna de la tabla debe ser simple (un solo valor). Cada columna tiene un nombre distinto (nombre del atributo), y el orden de las columnas no es importante. Cada renglón es distinto, esto es, un renglón no puede duplicarse en otro para un grupo de columnas seleccionadas como llave. Cada atributo no llave debe depender sólo de la llave de la relación no de ninguna otra no llave.
Tupla: Conjunto de valores que componen un renglón de la relación. Es equivalente a una instancia de un registro.
Grado de una tupla: Número de atributos que tiene una tupla (n de una n-tupla)
Cardinalidad: Número de tuplas de una relación.
Dominio: Conjunto de todos los valores posibles para un atributo.
Este modelo se está empleando con más frecuencia en la práctica, debido a las ventajas que ofrece sobre los dos modelos anteriores, entre ellas, el rápido entendimiento por parte de usuarios que no tienen conocimientos profundos

Bases De Datos Distribuidas

Las bases de datos distribuidas se están utilizando cada vez más en la misma medida en que se usan las arquitecturas de cliente-servidor y Groupware. Los principales problemas que se generan por el uso de la tecnología de bases de datos distribuidas son en lo referente a duplicidad de datos y a su integridad al momento de realizar actualizaciones a los mismos. Además, el control de la información puede constituir una desventaja, debido a que se encuentra diseminada en diferentes localidades geográficas.

Tendencias Futuras

En el futuro la mayoría de las organizaciones cambiarán la forma convencional de manejo de la información a la arquitectura de base de datos a las ventajas derivadas de su uso.

El uso de las bases de datos distribuidas se incrementará de manera considerable en la medida en que la tecnología de comunicación de datos brinde más facilidades para ello. El uso de bases de datos facilitará y soportará en gran medida a los Sistemas de Información para la Toma de Decisiones.

Sybase

Sybase es un sofisticado manejador de bases de datos relacionales. Soporta casi todas las instrucciones SQL, incluyendo subconsultas, transacciones, funciones y tipos de datos definidos por el usuario.

- Los sistemas de administración de base de datos (DBMS) relacionales tradicionales soportan un modelo de datos que consisten en una colección de nombres relacionados, conteniendo atributos de un tipo específico. En los sistemas comerciales actuales, los tipos posibles incluyen números de punto flotante, enteros, cadenas de caracteres, monetario, y fechas. El modelo relacional ha reemplazado exitosamente modelos previos. Sin embargo, como se mencionó, esta simplicidad muchas veces hace que la implementación de ciertas aplicaciones sea muy difícil. Otras características que proveen poder y flexibilidad adicional:
- Restricciones
- *Triggers*
- Reglas
- Integridad de transacciones

Oracle

Es manejador de base de datos relacional que hace uso de los recursos del sistema informático en todas las arquitecturas de hardware, para garantizar su aprovechamiento al máximo en ambientes cargados de información. Es el conjunto de datos que proporciona la capacidad de almacenar y acude a estos de forma consecuente con un modelo definido como relacional. Además es una suite de productos que ofrece una gran variedad de herramientas.

Historia de Oracle

El manejador de Base de datos ORACLE, surgió a final de los años 70 y principio de los años 80. George Koch y su equipo de tropas de asalto de técnicos fue el primero en desembarcar en el terreno de Oracle en 1982, durante un proceso de evaluación de sistema de gestión de base de datos para una importante aplicación comercial que George estaba diseñando y construyendo. Cuando termino, la evaluación fue descrita en Computer World como el estudio más severo de SGBD que se había hecho nunca. El estudio fue tan riguroso con los vendedores cuyos productos había estudiado George, que la prensa hizo eco de sus palabras en lugares tan distantes como Nueva

Zelandia y en publicaciones muy alejadas del campo como el Christian Sciencia Monitor.

El poderoso modelo relacional ha evolucionado desde herramientas y los modelos de datos de redes. La mayor manera aceptada y usada de un modelo de datos es el modelo relacional. El relacional conocido en 1969 con la revisión hecha por IBM.

Un modelo relacional posee tres grandes aspectos

- Estructuras: Definición de objetos que contengan datos y que son accesibles a los usuarios.
- Operaciones: Definir acciones que manipulen datos u objetos.
- Reglas: Leyes para gobernar la información, como y quien manipular.

Una base de datos relacional simplifica y definida como un modelo de información es estrictamente visualizable por los usuarios mediante tablas. Una tabla esta compuesta por una matriz bidimencional de filas y columnas. En cualquier ocasión la información es cambiada en una base de datos relacional, cualquier información es el resultado de una consulta presentad por el usuario en el formato filas/columnas.

Oracle soporta dos tipos de almacemiento, por caracter (RAW) o por bloques (Files System), generalmente es recomendable que los sean colocados en Raw Divice.

Raw Divice: es un dispositivo de caracteres disponibles en algunos sistemas operativos el cual es asignado directamente a Oracle.

Oracle corre más rápidamente con Raw Divice que con Files System, por varias razones:

1. E I/O (Input/Output) es realizado directamente en el disco por Oracle, independientemente del sistema operativo.
2. El buffer cache del sistema del sistema operativo es dejado a un lado.
3. Los buffers del sistema operativo y de oracle son independiente entre sí.

Con la intención de evitar la contención de los discos, se debe considerar la instalación de Oracle en dispositivos separados, especialmente si se tienen varios discos, y más esencialmente, si se poseen más de una controladora de disco. La planeación debe realizarse teniendo en cuenta los sigueintes criterios:

- Los Files System y sus dispositivos asignados.
- El swapping y paginamiento en Oracle, deberán estar en los dispositivos más rápidos.
- Los tablaspace para tables e índices en dispotivos separados.
- Los Log Files en un dispositivo separado al del tablespace de RDBMS Oracle.

Estructura Física y Lógica

Las estructura física tales como los archivos del sistema operativo, son almacenados tangibles como son cintas magnéticas, discos y otros. A cada archivo le corresponde un espacio en el sistema operativo. Oracle requiere de varios archivos para su funcionamiento, los cuales conforman su estructura física. A la estructura lógica le corresponde un espacio por unidad, pero sus limitaciones son independientes de las localizaciones de espacio físico.

Mejoras de SQL en Oracle

Oracle posee igual interacción en todas la plataformas (Windows, Unix, Macintosh y Mainframes). Estos porque más del 80% de los códigos internos de Oracle son iguales a los establecidos en todas las plataformas de Sistemas Operativos.

Oracle soporta bases de datos de todos los tamaños, desde severas cantidades de bytes y gigabytes en tamaño.

Oracle provee salvar con seguridad de error lo visto en el monitor y la información de acceso y uso.

Oracle soporta un verdadero ambiente cliente servidor. Este establece un proceso entre bases de datos del servidor y el cliente para la aplicación de programas.

MS SQL Server

La forma recomendada de publicar bases de datos en Internet es utilizando un **servidor de bases de datos**, ya que es un sistema mucho más robusto y escalable que archivos individuales como las bases de datos de Access, archivos de texto plano, etc.

SQL Server es el servidor de bases de datos de Microsoft, seguro, robusto y con las más avanzadas prestaciones: transacciones, procedimientos almacenados, triggers, etc. Además se puede gestionar la bases de datos en este servidor utilizando la consola de SQL Server que tenga instalada en su ordenador de trabajo. Por ello cada **Servidor Virtual Windows** tiene asignada una base de datos SQL Server incluida en el precio, con 20 Mb de espacio en disco, de los cuales 10 Mb corresponden a datos y 10 Mb a transacciones.

Con objeto de facilitar la gestión directa de las bases de datos MS SQL Server y el espacio en disco correspondiente, el **panel de control** del **Servidor Virtual** permite controlar en todo momento el espacio ocupado por las bases de datos y realizar los aumentos de espacio en disco y las altas y bajas de las bases de datos.

Es posible acceder al servidor MS SQL Server mediante Access o la consola de su MS SQL Server desde cualquier dirección IP. Sin embargo, estos accesos deben quedar limitados a las tareas propias de mantenimiento y/o sincronización de los datos; es decir, no se permite la utilización de programas ejecutables que accedan al servidor de datos como parte de su funcionamiento habitual. Se entiende que el servidor de datos es principalmente para ser usado por las aplicaciones web del **Servidor Virtual** o de cualquiera de sus dominios adicionales (scripts CGI, ASP...).

ODBC y gestión de DSN (Data Source Name)

El sistema de acceso a datos más extendido en los sistemas operativos de Microsoft se llama **ODBC** (Open Data Base Conectivity, 'conectividad abierta para bases de datos'). ODBC permite acceder a cualquier base de datos de la misma forma, independientemente de cuál sea el fabricante de la misma. Basta con que exista un controlador ODBC para ese tipo de bases de datos.

Gracias a esto y a que existen controladores ODBC para las bases de datos MySQL y PostgreSQL, nuestros clientes de **Servidores Virtuales Windows** pueden utilizar para sus aplicaciones cualquiera de los sistemas de bases de datos que ofrecemos: Access, SQL Server, MySQL y PostgreSQL. No importa el hecho de que MySQL y PostgreSQL residan físicamente en servidores Linux. Como nuestros Servidores Windows tienen los controladores ODBC de todos estos sistemas, nuestros clientes con aplicaciones ASP pueden utilizar esas bases de datos exactamente igual que si fuesen de Windows.

Para acceder a una base de datos vía ODBC se necesita un DSN (Data Source Name, 'nombre de origen de datos'). El DSN es simplemente un nombre que identifica la base de datos y que permite acceder a ella desde aplicaciones ASP propias o CGI.

Programando sus aplicaciones de forma que accedan a bases de datos mediante ODBC conseguirá que sean independientes del sistema gestor de bases de datos empleado y así podrá cambiar, por ejemplo, de Access a SQL Server o de SQL Server a PostgreSQL prácticamente sin cambiar su código.

Desde el **panel de control** de su **Servidor Virtual** podrá crear los DSN que necesite, tanto para bases de datos de Access (archivos .mdb que cargue en el servidor mediante FTP) como para Microsoft SQL Server.

Resumen

En este capitulo se describieron las características de la arquitectura cliente/servidor tres planos, la cual, por sus ventajas y beneficios, será aplicada en el desarrollo de este trabajo.

La elección de una arquitectura cliente/servidor (dos planos) presenta ventajas de simplicidad en el ciclo de desarrollo, aunque no necesariamente en el mantenimiento, pero a costa de obligar a que los mecanismos de bases de datos hagan funciones de servidores más allá del modelo de datos relacional, a la vez que obliga a depender demasiado de un proveedor de base de datos y de lenguajes propietarios poco estándares (SQL extendido), lo que puede dificultar la exportación de las aplicaciones a otras plataformas.

La elección de la arquitectura cliente/servidor (tres planos) brinda más grados de libertad en cuanto a la disponibilidad de plataformas, pero se debe tener en cuenta que las decisiones en la elección del software, si no son correctas, pueden hacer más compleja la implementación de una solución. De todas maneras se puede apelar a la característica de flexibilidad de esta estructura como para asegurar la posibilidad de contar con alternativas suficientes para tomar buenas decisiones.

Para el presente trabajo se eligió a sybase como servidor de base de datos, además de las ventajas que se describieron arriba, por que es el recurso que fue asignado por las personas de DGSCA para el desarrollo de este sistema. Utilizar recursos ya existentes nos beneficia ya que no será necesario llevar a cabo instalaciones y migraciones de la base de datos de Agenda Cultural que ya existe.

El gran inconveniente de los CGI's es el rendimiento, ya que por cada petición se crea una nueva copia del programa en la memoria del servidor. Así, si acceden muchos usuarios de forma simultánea se produce una disminución en la eficiencia de dicho servidor. Los programas CGI pueden presentar agujeros de seguridad de manera intencional o no, pueden mostrar información sobre el sistema que puede ayudar a terceros a comprometerlo. Pueden realizar tareas dentro del sistema con la información proveída por el usuario remoto, permitiendo que con esta información le deje ejecutar comandos del sistema. Otra tecnología ampliamente utilizada hoy en día y que ha sido elegida en esta ocasión para el desarrollo de la presente tesis, son los Java Servlets y JSP's. Su funcionamiento es similar al de los CGI's, con la diferencia de que los programas que se ejecutan en el servidor están escritos en lenguaje Java. Esta tecnología soluciona el problema del rendimiento gracias a que sólo mantiene una imagen del programa en memoria, aunque existan múltiples peticiones. Otra importante ventaja es la portabilidad que ofrece utilizar *Java*. Asegura la independencia de la plataforma, al contrario de lo que ocurre con *ASP*. Además utiliza un lenguaje totalmente orientado a objetos y mucho más robusto que los lenguajes *scripts*.

Aunque en el mercado existen muchos servidores Web y entre los que se describieron en este capitulo se encuentra alguno con grandes características, como podría ser el Iplanet, su costo es grande, por tal razón se eligió al contenedor de JSP's Tomcat, el cual es de distribución libre,

además de contar con las características necesarias para soportar una aplicación como la que se va a desarrollar a lo largo de este trabajo. Además del costo, el tiempo de adquisición es mínimo ya que este software se puede obtener de Internet.

CAPÍTULO 3

Análisis y diseño de una aplicación Web

CAPITULO 3

El desarrollo de aplicaciones Web involucra decisiones no triviales de diseño e implementación que inevitablemente influyen en todo el proceso de desarrollo, afectando la división de tareas. Los problemas involucrados, como el diseño del modelo del dominio y la construcción de la interfaz de usuario, tienen requerimientos disjuntos que deben ser tratados por separado.

El alcance de la aplicación y el tipo de usuarios a los que estará dirigida son consideraciones tan importantes como las tecnologías elegidas para realizar la implementación. Así como las tecnologías pueden limitar la funcionalidad de la aplicación, decisiones de diseño equivocadas también pueden reducir su capacidad de extensión y reusabilidad. Es por ello que el uso de una metodología de diseño y de tecnologías que se adapten naturalmente a ésta, son de vital importancia para el desarrollo de aplicaciones complejas.

• 3.1 Diseño gráfico

Una vez que se cuenta con el análisis del sistema y la lista de requerimientos, el siguiente paso es la creación de la interfaz de usuario. Esta es una etapa importante en el desarrollo de aplicaciones, la interfaz de usuario es la que va a determinar que tan utilizable va a ser un sistema.

En ocasiones los programadores subestiman la importancia de la interfaz de usuario y concentran sus esfuerzos en la funcionalidad del sistema y en optimizar su desempeño. Sin embargo, si la interfaz de usuario es pobre, el usuario no se sentirá cómodo al usar la aplicación o, peor aún, dejará de usarla.

Una buena interfaz de usuario debe de ser diseñada según ciertos lineamientos:

Consistencia Todas las pantallas de la aplicación Web deben de tener una distribución consistente de imágenes, texto y controles gráficos. Ante acciones de usuario comunes (elegir un elemento de una lista desplegarle, dejar un campo en blanco donde no es permitido) el sistema debe de mostrar mensajes con la misma estructura sin importar en que parte del sistema ocurra el evento.

La única manera de lograr una interfaz de usuario consistente es la implementación de estándares de interfaz. Con ello, se sabrán como acomodar los controles gráficos, dónde poner las etiquetas, que lineamientos de justificación de texto seguir, entre otras cosas.

- **Dar soporte a diferentes niveles de usuarios.** Una aplicación debe poder ser usada tanto por usuarios neófitos como por expertos, por lo que, la aplicación debe de ser configurable para mostrar toda la gama de opciones que un usuario experto puede requerir o solo mostrar las indispensables para que un usuario neófito pueda hacer su trabajo sin verse agobiado por una gran cantidad de opciones.
- **Flujo de pantallas.** El paso de una pantalla a otra debe de ser coherente con el trabajo que intente realizar el usuario. De esta manera, no es nada conveniente hacer que el usuario tenga que pasar por una pantalla de configuración de colores en pantalla para llegar a la pantalla de configuración de márgenes de impresión.
- **No sobrepoblar las pantallas.** Al presentar gran cantidad de controles gráficos en una misma pantalla dificultan la comprensión de la misma. Si para realizar una tarea determinada se requiere de la obtención de mucha información por parte del usuario, es muy recomendable dividir en diferentes pantallas sucesivas la captura de información y/o establecimiento de opciones. A esto último se le conoce como asistente o ``wizard".

- **Agrupar elementos relacionados.** Una buena práctica en la creación de interfaces de usuario es la agrupación de elementos que estén relacionados entre sí. Por ejemplo, podemos delimitar por un rectángulo o algún otro tipo de imagen los datos generales de un empleado tales como nombre, dirección.

Existen lineamientos que son específicos al desarrollo de aplicaciones Web, una buena referencia es "*Los siete pecados mortales de un sitio Web (y por qué deben de evitarse a toda costa)*" de Jesse Berst . Los grandes pecados o errores que se pueden cometer en el diseño de un sitio Web son:

- **Navegación inconsistente.** A veces se hace click en una barra de menú lateral, otras es un menú despegable. La ubicación de los títulos, gráficos, vínculos, etc. debe de ser consistente en todas las páginas del sitio.
- **Vínculos rotos.**
- **Sitios que requieren de un navegador específico.** Es muy molesto toparse con sitios que están diseñados para un navegador específico, así como los que requieren que se baje un *plug-in* determinado sin ofrecer una versión del documento que no requiera de dicho plug-in.
- **No poner información de contacto.** Muchos diseñadores de sitios de empresas importantes olvidan poner la dirección de la empresa, números telefónicos, números de atención a clientes, etc.
- **Uso de marcos.** También conocidos como *frames*, traen consigo problemas con el uso de los botones Anterior y Siguiente de los navegadores, muchas veces son creados de un tamaño fijo y la información contenida en ellos no es visible completamente, etc.
- **Sitios que abren otras ventanas de navegación.** Es molesto para el navegador ingresar a un sitio y que éste abra nuevas ventanas de navegación con vínculos a otros sitios.
- **Símbolos de "En construcción".** La mayoría de los sitios en el Web están en constante evolución, Es poco profesional indicar que el sitio está en construcción

• 3.2 Diseño del sitio.

La estructura de un sitio web se va a referir a la disposición entre los enlaces de las diferentes páginas que lo forman, es decir, al esquema general de disposición de las páginas entre sí y a la forma de acceso entre ellas.

Estructura general de una aplicación Web

En esta sección o se hablará acerca de los problemas o situaciones con los que se tiene que tratar al desarrollar una aplicación Web. Pero antes es necesario conocer cómo es que funciona una aplicación Web. La estructura general de una aplicación Web, en primera instancia, lo que ve un usuario al ingresar a un sitio que alberga una aplicación Web es la página de contenido inicial, ésta es la página de bienvenida.

Las operaciones que se realizan en una aplicación Web requieren de un medio de almacenamiento persistente, comúnmente esto es realizado por medio de un manejador de bases de datos (DBMS por sus siglas en inglés); la aplicación Web debe de tener un medio para comunicarse con el DBMS con el fin de leer y escribir la información en la base de datos.

Un aspecto importante en el diseño de una aplicación Web es el uso de plantillas. Las plantillas son una serie de archivos que definen la apariencia de una aplicación Web, de esta manera, si

se desea cambiar la presentación de la aplicación, sólo se tiene que modificar las plantillas sin necesidad de modificar el código que le da funcionalidad a la aplicación.

Los principales tipos de estructura son

1) **Estructura jerárquica**: que parte de una página principal mediante la que se puede acceder a diferentes páginas secundarias, a partir de las cuales podemos acceder a las páginas terciarias, y así sucesivamente. La disposición de un sitio de este tipo sigue el esquema general expresado en el siguiente gráfico:

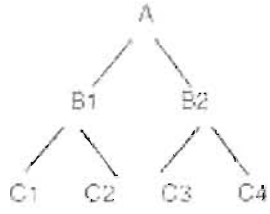


Fig. 3.1

Si usamos una estructura de tipo jerárquica podemos crear un menú general en la página principal, mediante el que daremos acceso a las diferentes páginas de entrada a las secciones, y en cada una de estas deberemos establecer otro menú desde el que el usuario pueda ir a cualquiera de las páginas que la componen. En cada una de las páginas individuales tendremos que implementar unos enlaces a las diferentes secciones principales y a la página de inicio.

2) **Estructura lineal**: en la que partiendo de una página inicial se van recorriendo las demás del sitio web secuencialmente, una detrás de otra. Es análoga en su disposición a la estructura de las páginas de un libro. Cada página posee un enlace a su anterior en la secuencia y otro a su siguiente. La representación gráfica es la siguiente:

El sistema de menús característico de este tipo de estructura sería el acceso a una página de entrada, desde la que podemos acceder únicamente a la página que le sigue en la secuencia establecida, y en esta encontraremos un pequeño menú, generalmente situado en la parte inferior o superior de la misma, desde el que podemos acceder tanto a la página anterior como a la siguiente en la secuencia, y así sucesivamente, hasta llegar a la última página, en la que sólo figurará un enlace a la página anterior. Debido a sus repercusiones de diseño y navegación, este tipo de estructuras en su forma pura es raramente usado.

3) **Estructura lineal-jerárquica o mixta**: que como su propio nombre indica es una mezcla de las dos anteriores, en la que partiendo de una página principal o de inicio se accede a diferentes páginas de entrada a secciones, a partir de las cuales la navegación es lineal.

Su representación gráfica es la siguiente:

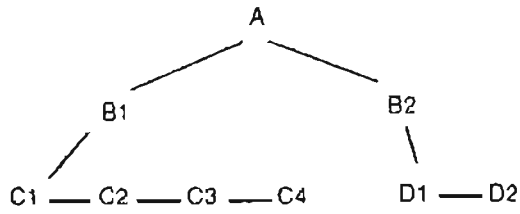


Fig. 3.2

En el caso de usar este tipo de jerarquía deberemos situar un menú en la página de inicio desde el que podamos acceder a las páginas de entrada a cada uno de los temas a tratar, y en cada una de las páginas que forman la secuencia del tema tendremos que establecer un link a la página anterior y otro a la siguiente. Como complemento podemos habilitar en cada una de ellas una liga a la página que abre la secuencia, y en cada una de estas otras a la página de inicio

4) **Estructura de frames**: que es la típica de una interfaz a base de frames y en la que el usuario dispone de un menú siempre presente desde el que puede acceder en todo momento a las páginas de entrada a las diferentes secciones del sitio Web, a partir de las que puede navegar bien de forma jerárquica, bien de forma lineal, bien de forma mixta. Su representación gráfica es del tipo:

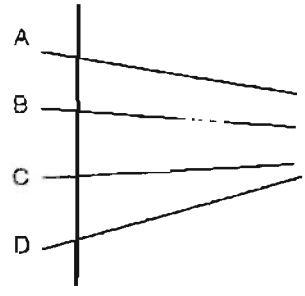


Fig. 3.4

Este tipo de estructura se suele combinar con otro jerárquico o mixto que nos ayude a navegar dentro de los subsistemas principales, a los que accedemos a través del frame lateral.

El sistema de menús consiste generalmente en un menú situado en un frame lateral, que nunca cambia, desde el que se accede a las diferentes secciones principales del sitio, cuyas páginas iniciales se cargan en el frame central, que suele ser el de mayor tamaño. Para acceder a las diferentes subsecciones y páginas se suelen establecer los enlaces adecuados bien como submenús en el frame lateral, bien como menús individuales dentro de cada página de entrada a las secciones, dentro del frame principal.

Estructura web: en la que podemos estructurar las diferentes páginas con libertad total. Es la que da más facilidades a los diseñadores, pero puede resultar a veces demasiado confusa para los usuarios, ya que le permite visitar nuestro sitio sin un rumbo fijo, pudiendo desde cualquier página acceder a los contenidos de un conjunto indeterminado de otras. No es aconsejable su uso, ya que suele resultar caótica. Su representación gráfica puede ser del tipo:

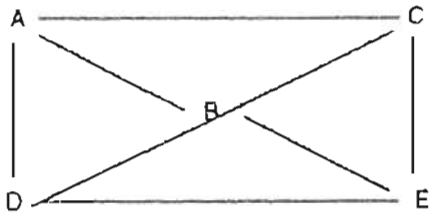


Fig 3.5

Estos son los tipos principales de estructura de un sitio Web. Generalmente se suele usar un tipo mixto o un tipo frames, dependiendo siempre de la naturaleza propia del sitio.

En el caso del Sitio de Difusión Cultural UNAM el diseño estará basado en una estructura mixta, como se puede observar en el siguiente diagrama general:

En el que se puede observar la existencia de un a página principal que contiene ligas a cada uno de las secciones con las que cuenta, a partir de las cuales se puede seguir navegando en forma lineal. Por lo que cumple con las características de un una estructura mixta. Además la página de registro de usuarios esta diseñada con la estructura de frames.

• 3.3 Análisis del sistema

Metodologías

Modelo de Cascada

Para diseñar un buen sistema de software, diferentes métodos han sido propuestos para describir ya sea un proyecto para desarrollar una primera versión de un producto o para dar una vista global del ciclo de vida de un sistema completo. La definición de un "buen" sistema de software varía dependiendo de las aplicaciones. Por ejemplo, en algunos casos es el rendimiento lo que importa, mientras que en otros es más importante la interfaz gráfica con el usuario. También depende de la estructura del sistema, por ejemplo, si esta será distribuida o centralizada. Lo que es común es que todos los sistemas necesitan ser modificados en algún momento.

Tradicionalmente, el trabajo de análisis y diseño es estructurado y descrito usando diferentes tipos de modelos de cascada, ver figura 3.6. Estos modelos describen el proceso de desarrollo del sistema donde la idea inicial es que cada fase deberá completarse antes de iniciar la siguiente. Sin embargo, esto último fue rápidamente abolido y en la actualidad es permitido iniciar una fase antes de completar totalmente la anterior. El modelo de cascada ha tenido un tremendo impacto sobre los métodos de ingeniería de software, aunque éste no se sigue de manera rígida cuando es aplicado.

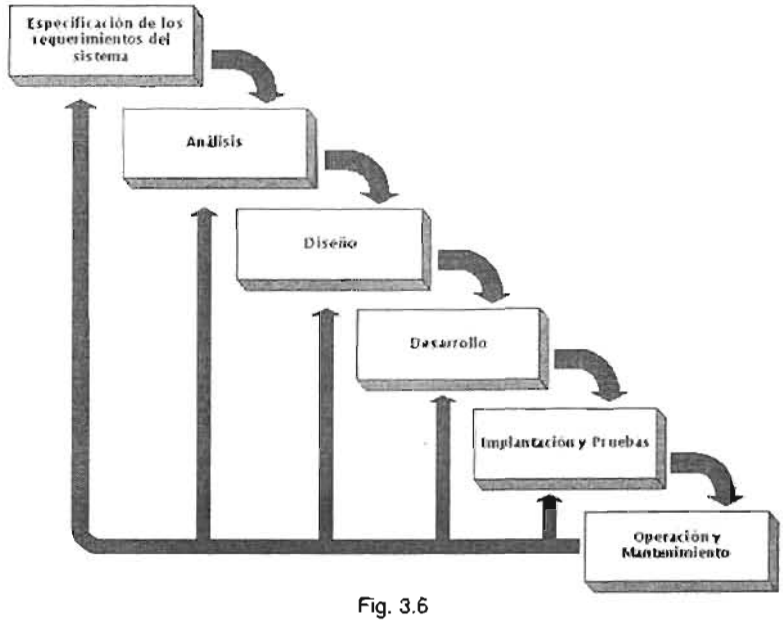


Fig. 3.6

Este método tiene como etapas la de:

- Especificación de los requerimientos del sistema
- Análisis
- Diseño
- Desarrollo
- Implantación y pruebas
- Operación y mantenimiento

En la primera etapa que es la de Especificación de los requerimientos del sistema se llevó acabo el estudio de que es lo que se pretendía que el sistema hiciera, así como obtener un entendimiento conceptual del problema a resolver y que tipo de herramientas, modelos y estructuras se iban a emplear.

En la etapa de análisis se hizo la definición del problema para así poder plantear los objetivos y posibles mejoras al sistema actual.

En la etapa de diseño una vez conocido el problema, se identifican los distintos procesos computacionales y se define una arquitectura (basada en los requisitos y arquitectura tecnológica) y una implantación. La especificación, está muy relacionada con el software básico específico que se utiliza en la construcción que en este caso corresponde a las herramientas de desarrollo.

En esta etapa de desarrollo se lleva a cabo la creación de los módulos, su programación así como la creación de la base de datos.

En la etapa de implantación y pruebas se hace la integración del sistema para así poder hacer pruebas pilotos.

En la etapa de operación y mantenimiento se da cuando el sistema está concluido, aprobado por el usuario final y listo para su operación.

• 3.4 Modelado de la aplicación

La situación actual del desarrollo de software requiere de grandes esfuerzos en la etapa diseño y análisis debido a la complejidad de las aplicaciones que diversos tipos de usuarios requieren hoy en dia. Estas aplicaciones deben basarse en modelos del mundo real; y un método que aproxima de una forma evidente a dicho mundo es la metodología orientada a objetos.
El modelado de objetos permite abstraer los sistemas en clases y métodos que son manipulados mediante aplicaciones y lenguajes orientados a objetos, con el objeto de facilitar el control en el diseño e implantación de sistemas de gran magnitud, los cuales resulten imposibles, en muchos casos, de ser comprensibles para una persona.

Otro aspecto importante en la metodología orientada a objetos es la relación existente entre el análisis de requisitos, que es la relación entre la asignación de software al nivel del sistema y el diseño del software, para lo cual han surgido diversas metodologías como son la Object Oriented Software Engineering (**OOSE**), Object Modeling Technique (**OMT**) y el método Booch.

UML

El Lenguaje de Modelamiento Unificado (UML -Unified Modeling Language). UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

La versión 1.0 de UML fue lanzada en Enero de 1997, ha sido utilizado con éxito en todo tipo de industria a todo nivel. El modelado tiene como objetivo principal el de describir la conducta de una aplicación basada en los requerimientos de funcionalidad, desempeño y confiabilidad.

- Mejores tiempos totales de desarrollo (reducidos en un 50 %).
- 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 maquinas.
- Mejor soporte a la plantación y al control de proyectos.
- Alta reutilización y minimización de costos
- Básicamente UML describe al modelo mediante una colección de vistas, diagramas, símbolos, y reglas que constituyen un lenguaje de modelado.
- Cada uno de estos elementos describen un aspecto específico del producto o sistema en construcción.

Etapas de desarrollo
UML utiliza diversos elementos del lenguaje de modelado para diversas etapas de desarrollo de sistemas, estas etapas son:

- Análisis de Requerimientos
- Análisis
- Diseño
- Programación
- Pruebas

3.5 El Proceso Unificado de Desarrollo

Una metodología de desarrollo es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software, esta basado en componentes y utiliza el Lenguaje de Modelado (UML).

- Dirigido por casos de uso
- Centrado en la arquitectura
- Iterativo e incremental

Casos de Uso

Es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante (Requisitos funcionales), todos los casos de uso constituyen el modelo de casos de uso.

Un caso de uso es un escenario típico de interacción entre el usuario y la aplicación. Esta interacción puede parecer de repente demasiado sencilla u obvia, por lo cual es usualmente omitida en un esquema de documentación, y por consiguiente, no es considerada como parte de una planeación plenamente constituida.

Estas interacciones o casos de uso tienen como caracteristicas las siguientes:

- El caso de uso capta alguna función visible para el usuario.
- El caso de uso puede ser pequeño o grande.
- El caso de uso logra un objetivo discreto para el usuario.

Para lograr esto es necesario analizar la funcionalidad deseada en el sistema directamente con los usuarios, para entonces definir las interacciones requeridas.

Diagramas de casos de usos

Las interacciones están representadas en un diagrama de casos de usos mediante actores que son la representación de los usuarios, y los casos de uso que son los elementos a los que tiene acceso el actor en el sistema.

Es posible obtener a los actores de un diagrama de casos de uso a través de las siguientes preguntas:

- ¿Quién utilizará la funcionalidad principal del sistema (actores primarios)?
- ¿Quién necesitará soporte del sistema para realizar sus actividades diarias?
- ¿Quién necesitará mantener, administrar y trabajar el sistema (actores secundarios)?
- ¿Qué dispositivos de hardware necesitará manejar el sistema?

- ¿Con qué otros sistemas necesitará interactuar el sistema a desarrollar?
- ¿Quién o qué tiene interés en los resultados (los valores) que el sistema producirá?

Se recomienda que solo se muestren aquellos actores del sistema que necesiten de un caso de uso, para así facilitar la comprensión y el diseño de los requerimientos. Un actor además de ser un ser humano, puede ser un sistema externo que necesite de cierta información, o requiera de ciertas acciones del sistema actual.

Interacción de los casos de uso

Un caso de uso representa la funcionalidad completa tal y como la percibe un actor. Un caso de uso en UML es definido como un conjunto de secuencias de acciones que un sistema ejecuta y que permite un resultado observable de valores para un actor en particular. Gráficamente se representan con una elipse y tiene las siguientes caracteristicas:

- Un caso de uso siempre es iniciado por un actor.
- Un caso de uso provee valores a un actor.
- Un caso de uso es completo.

Encontrando casos de uso

El proceso para encontrar casos de uso inicia encontrando al actor o actores previamente definidos. Por cada actor identificado, hay que realizar las siguientes preguntas:

- ¿Qué funciones del sistema requiere el actor? ¿Qué necesita hacer el actor?
- ¿El actor necesita leer, crear, destruir, modificar o almacenar algún tipo de información en el sistema?
- ¿El actor debe ser notificado de eventos en el sistema o viceversa? ¿Qué representan esos eventos en términos de funcionalidad?
- ¿El trabajo diario del actor podría ser simplificado o hecho más eficientemente a través de nuevas funciones en el sistema? (Comúnmente, acciones actuales del actor que no estén automatizadas)
- Otras preguntas que nos ayudan a encontrar casos de uso pero que no involucran actores son:
- ¿Qué entradas/salidas necesita el sistema? ¿De dónde vienen esas entradas o hacia dónde van las salidas?
- ¿Cuáles son los mayores problemas de la implementación actual del sistema?

Ejemplo práctico del diagrama de casos de usos

A continuación se presenta el diagrama Fig.3.7 de casos de uso propuesto para el sistema de encuestas.

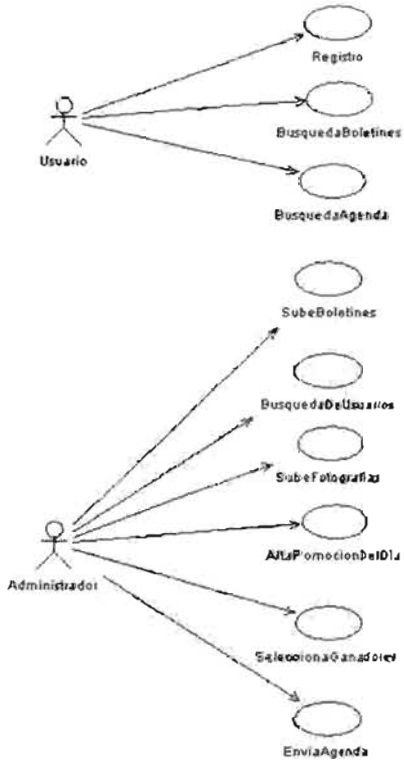


Figura 3.7 Casos de uso.

Este diagrama propone la existencia de 2 actores, que son el Usuario, y el administrador. Estos interactúan con diversas partes o módulos del sistema, los cuales conforman los casos de uso.

Escenarios

Un escenario en UML consiste en una ruta específica dentro de un diagrama de casos de uso en UML, mostrando una particular combinación de circunstancias o condiciones que requieran ser indicadas como importantes y críticas para posteriores etapas en el desarrollo del modelo.

Realizaciones

Tanto los diagrama de casos, como la concepción del modelo en general tiene u una diversidad de conceptos y abstracciones. Esto varía dependiendo del estilo y experiencia del analista. A cada una de las propuestas de modelado de UML se le conoce comúnmente como realización, y usualmente son creadas varias realizaciones desde diversos puntos de vista, para así confrontarlos y llegar a un modelo mucho más representativo.

Arquitectura

La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado.

1.- Crear esquemas de la arquitectura
2.-Trabajar con un conjunto de casos de uso, se reparte en subsistemas, clases, y componentes.
3.- Al madurar los casos de uso se desarrolla mas la arquitectura esto lleva a madurar mas casos de uso continúa hasta que la arquitectura es estable.

Iterativo e incremental

Se divide todo el trabajo en mini proyectos, cada mini proyecto es una iteración (flujo de trabajo) que resulta en un incremento (crece el producto). Cada iteración tiene una serie de flujos de trabajo: requisitos, análisis, diseño, implementación y prueba..

Iteración

Esfuerzo de trabajo en un proyecto que corre varias etapas de desarrollo (no necesariamente todas), al final del cual se ha incrementado el material disponible sobre el sistema.

Incremento

Un avance significativo en el grado de especificación, diseño, implementación, o prueba del sistema que tenga lugar durante una iteración

Vida de un sistema

La vida de un sistema es una serie de ciclos nacimiento + ciclos intermedios + muerte, cada ciclo tiene varias fases, una fase es un intervalo de tiempo entre dos hitos importantes del proceso, cuando se cumplen un conjunto de objetivos bien definidos, se completan los artefactos y se toman las decisiones sobre si se pasa a la siguiente fase.

Planificación

- Fase de un ciclo

- Inicio
- Elaboración
- Construcción
- Transición

Fase de elaboración

- Se definen la visión del proyecto y su arquitectura
- Se expresan con claridad los requisitos del sistema, son priorizados y son utilizados para crear una sólida base arquitectónica
- Se planifican las actividades y los recursos necesarios

Fases de construcción

- Se construye el producto mediante una serie de iteraciones incrementales
- Se lleva el software desde una base arquitectónica ejecutable hasta su disponibilidad de usuarios

Fase de un ciclo

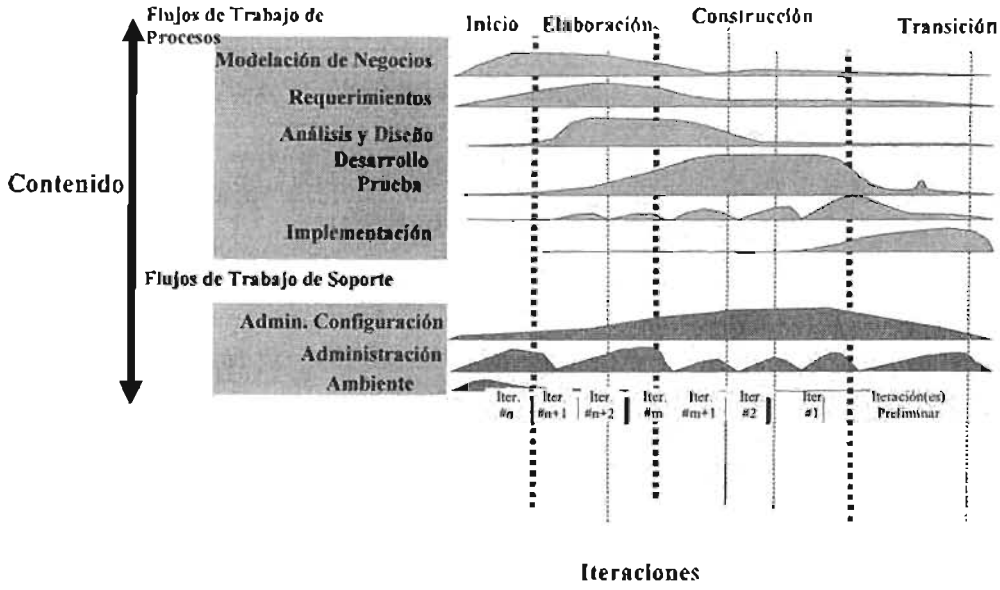


Fig 3.8 Fases de un ciclo de desarrollo con RUP

Ciclo de un sistema

- Cuando se han recorrido las cuatro fases, se dice que el sistema ha sufrido un ciclo

- Cada ciclo produce una versión del sistema
- Cada versión es un producto preparado para su entrega

Resumen

La estructura de diseño que ha sido elegida para el desarrollo de este sistema cuenta con una combinación de todas las descritas en la primer sección de este capitulo. Estructura lineal-jerárquica o mixta, ya que existen secciones del sitio que funcionaran mucho mejor si el flujo es lineal, y otras que necesitaran una estructura de jerarquía, los frames también son un elemento esencial en el diseño de este portal, ya que la presentación de información, ligas y secciones será mejor distribuida a través de este mecanismo.

Actualmente las tendencias del desarrollo de software se direccionan hacia el campo de Orientado a Objetos, lenguajes como JAVA de Sun, .NET de Microsoft, compiten por el liderazgo, ambos orientados a objetos. Ante esta situación la elección de una metodología de desarrollo se vuelve necesariamente hacia una del mismo tipo, en este caso, RUP, descrita en este capítulo. Aunque existen otras metodologías OO, RUP, por sus características ha mostrado ser de las más practicas en cuanto a proceso de desarrollo, por esta razón es la aplicada en el siguiente capitulo para el análisis y desarrollo de este trabajo.

CAPÍTULO 4

Desarrollo del Sistema

4.1 Definición del problema

4.2 Sistema propuesto

4.3 Método empleado para la solución

4.4 Etapas de Desarrollo de la aplicación

4.5 Diseño y construcción de la Base de Datos

4.6 Diseño grafico

CAPITULO 4

En este capitulo se lleva a cabo, paso a paso el desarrollo de un sitio Web para Internet. Siguiendo el procedimiento de RUP

- **4.1 Definición del Problema**

Difusión Cultural UNAM es el principal órgano de difusión de eventos culturales dentro de la Universidad, para ello, hace uso de diversos medios de comunicación, en el caso que nos ocupa de Internet, por lo que se vuelve indispensable la creación de un Portal con diversas funciones para la mejor difusión de los eventos.

Actualmente la pagina que se encuentra en línea es una pagina estática, la cual solo tiene ligas a otras paginas. Esto aunque ha sido de gran utilidad no satisface completamente las necesidades de difusión. La página actual no cuenta con un apartado de administración que facilite el mantenimiento de dicha pagina, por ejemplo, el proceso de actualización de los boletines es obsoleto y se requiere de algún con conocimientos de HTML para poder llevar a cabo esta actualización.

- **4.2 Sistema propuesto**

Con el presente trabajo de tesis se propone un Portal para difusión cultural que cuente con las funciones requeridas para imprimir el dinamismo que requiere.

Objetivos:

Diseñar e implementar un Portal para Difusión Cultural UNAM que facilite y optimice las tareas de difusión y publicación de los eventos culturales de la universidad.

- **4.3 Método empleado para la solución**

El sistema será diseñado bajo una arquitectura cliente/servidor, tres capas, en un ambiente de red, y con prestaciones Web, teniendo una base de datos relacional que almacene toda la información recabada en los registros.

El sistema correrá en cualquier plataforma capaz de soportar a TomCat como servidor Web, el manejador de base de datos a utilizar será Sybase, se desarrollara con Tomcat como contenedor de servlets y JSP's. Para su instalación en el servidor de DGSCA el sistema será entregado en un War para ser depositado en el servidor de DGSCA.

La base de datos será diseñada tomando en consideración todas las reglas de las bases de datos relacionales en cuanto al acceso, seguridad e integridad de los datos, bajo un modelo de entidad - relación.

La aplicación será diseñada siguiendo los lineamientos de RUP, Unified Modeling Language (**UML**) y será programada utilizando el lenguaje Java Sever Pages (**JSP**), se eligieron estas tecnologías debido a las razones descritas en los resumenes de los capítulos anteriores.

Adicionalmente a JSP que es un lenguaje embebido en documentos HTML, se requerirá utilizar otros recursos como Structured Query Language (**SQL**), JavaScript, y FLASH para el diseño gráfico.

Siguiendo la metodología de RUP comenzamos describiendo a continuación los requerimientos del sistema.

- **4.4 Etapas de Desarrollo de la aplicación**

Comenzaremos describiendo las etapas de desarrollo de la aplicación basándonos en las disciplinas establecidas por RUP, las cuales se describen a continuación, mencionando la solución propuesta:

Requerimientos

Difusión cultural requiere de un sistema que cumpla con las siguientes características:

- Realizar registro de usuarios del Portal
- Publicación de Boletines
- Publicación de promociones
- Buscador en la BD de la agenda cultural
- Acceso a información de los usuarios

Además de la descripción, se hicieron algunos diagramas para detallar el flujo de la aplicación

Diagramas genéricos de la aplicación

Cada diagrama muestra en forma general el flujo de la información en los distintos escenarios que tendrá el sistema, estos diagramas se realizan con el fin de lograr mas detalle en lo que se refiere a los requerimientos.

Cada uno de los diagramas mostrados a continuación, en casos de uso y en diagramas de secuencia y de clases que describirán paso a paso la interacción entre cada una de las clases participantes en cada caso de uso.

Actores

1.- Usuario
2.- Administrador

Las siguientes imágenes muestran los diagramas de casos de uso para cada uno de los actores.

Casos de Uso

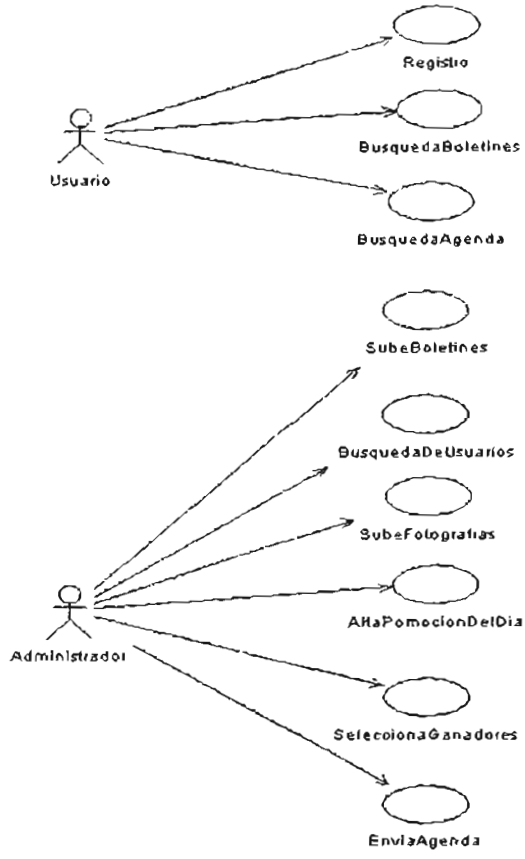


Fig. 4.1

Como se aprecia en la figura anterior, las actividades realizadas por el usuario será, su registro dentro del portal, participación en las promociones del día y el uso de los buscadores.

Descripción de cada caso de uso

Registro

1.- El usuario elige la opción de registro en la página principal.
2.- El sistema muestra un formulario para que el usuario capture sus datos.
3.- El usuario elige enviar información.
4.- El sistema muestra una vista preliminar de la información capturada por el usuario, preguntándole a este si esta de acuerdo con la información mostrada y dando la alternativa de corregir datos.
5.- El usuario elige una de las dos opciones

5.1 Si elige guardar información, el sistema guarda los datos del usuario en la base de datos y automáticamente envía un correo de notificación al usuario con su número de registro.
5.2 Si elige corregir información, el sistema regresa al formulario de captura para que el usuario corrija sus datos y sigue el procedimiento del punto 4.

6.- El caso de uso termina cuando el sistema muestra al usuario un mensaje de envió exitoso.

Búsqueda de Boletines

1.- El usuario elige en la página principal la opción de sala de prensa.
2.- El sistema muestra una pantalla para que el usuario capture su número de registro, y una liga para que en caso de no tener número de registro pueda registrarse en ese momento.

2.1 El usuario tiene su número de registro y lo captura en el formulario mostrado.

a) El sistema verifica que el número de usuario exista si existe el registro el sistema lo direcciona a la sala de prensa, en caso de no existir el número el sistema envía al usuario a la página de registro.
b) En la pantalla que se muestra hay una lista de los boletines publicados para el mes actual
c) El usuario selecciona la liga del boletín que desea consultar
d) El sistema abre una nueva ventana con el documento del boletín y las fotografías asociadas a este.
e) En el momento en que se abre el boletín se registra en la base de datos el número de usuario y el documento que se ha consultado para llevar un histórico de este.

2.2 El usuario no cuenta con un número de registro.

a) El sistema redirecciona a la pantalla de registro para que el usuario haga su registro en ese momento.

Búsqueda en agenda cultural

1.- El usuario elige Buscador en la página principal
2.- El sistema muestra un formulario en el que el usuario deberá elegir la categoría del evento a buscar.
3.- El sistema muestra una página con el resultado de la búsqueda.

Sube Boletines

1.- El administrador dentro del ambiente de mantenimiento, elige la opción de subir boletines.
2.- El sistema muestra un formulario para que el administrador capture la información del boletín a subir.
3.- El sistema presenta una vista preliminar de la informaron capturada y solicita el archivo del boletín a subir.
4.- El sistema muestra un mensaje de los resultados del proceso y las opciones de subir otro boletín y agregar fotografías al boletín.
5.- Si se elige agregar fotografías al boletín se muestra un formulario para capturar los datos de la fotografía así como el archivo de esta.
6.- El sistema envía un mensaje del resultado del proceso.

Sube Fotografías

1.- El administrador dentro del proceso de sube boletines selecciona la opción subir fotografía. Continúa en el punto 5 del caso de uso Sube Boletines.
2.- El caso de uso termina cuando el usuario elige la opción de salir.

Alta promoción del día

1.- El administrador dentro del ambiente de mantenimiento, elige la opción de alta de promoción del día.
2.- El sistema muestra un formulario para capturar la información del la promoción, el administrado elige enviar información.
3.- El sistema almacena la información del la promoción del día.
4.- El caso de uso termina cuando el usuario elige la opción de salir.

Selecciona ganadores

1.- El administrador dentro del ambiente de mantenimiento, elige la opción de seleccionar ganadores de la promoción del día.
2.- El sistema muestra una lista con los primeros 10 registros del dia.
3.- El caso de uso termina cuando el usuario elige la opción de salir.

Análisis y Diseño

La siguiente disciplina se refiere a identificar los casos y uso y actores para el sistema. En este caso, se identificaron dos actores principales.

A continuación se muestran los diagramas de secuencia para cada Caso de Uso

Registra Usuario

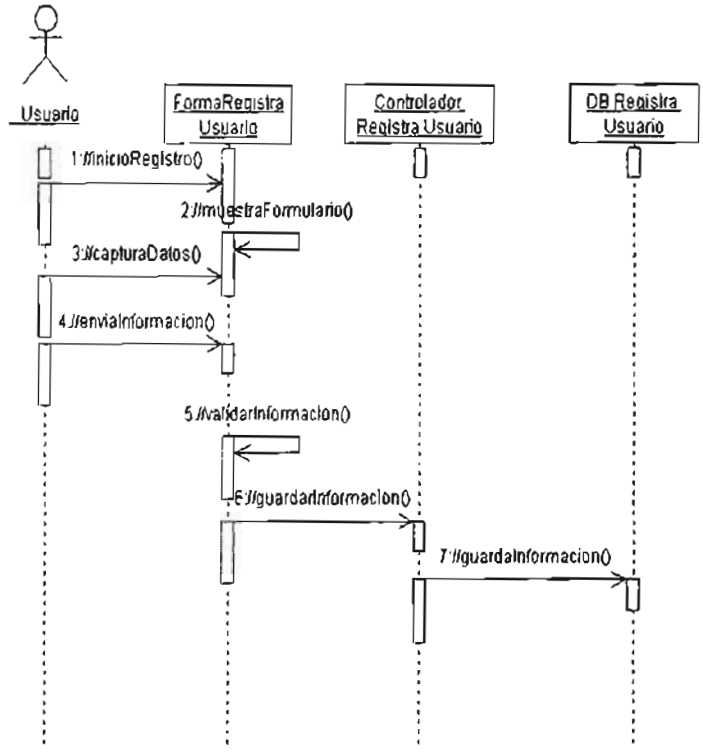


Fig 4.2 Diagrama de secuencia para el Caso de Uso de registro

Buscador

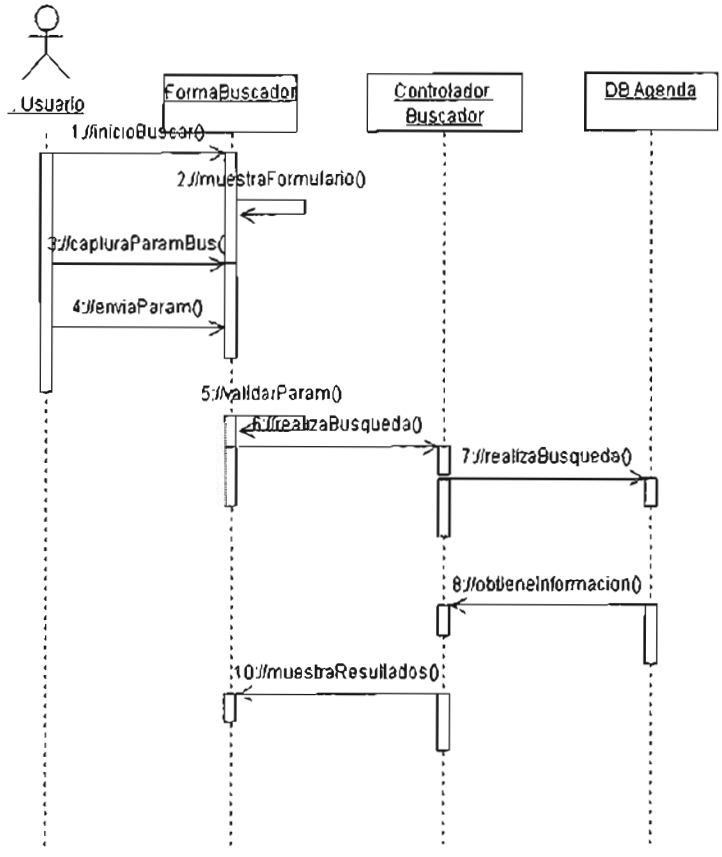


Fig 4.3 Diagrama de secuencia para el Caso de Uso de registro

Alta Boletines

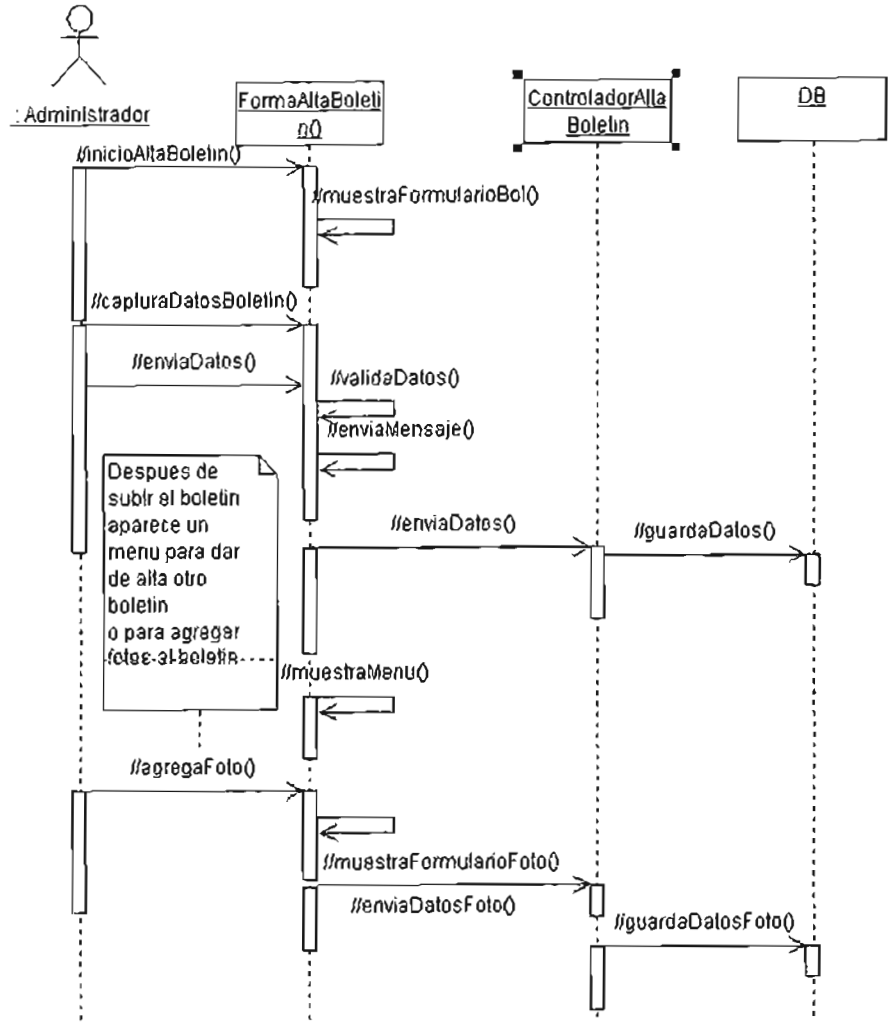
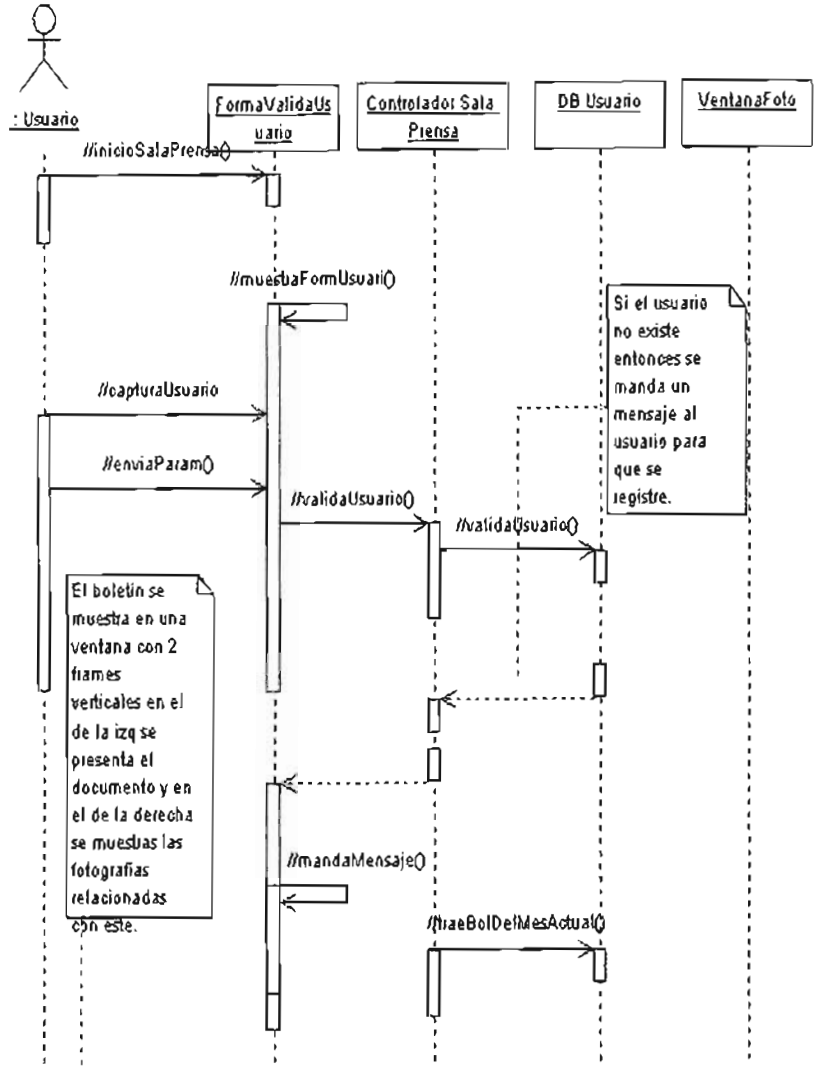


Fig 4.4 Diagrama de secuencia para el Caso de Uso alta boletines

Sala de Prensa



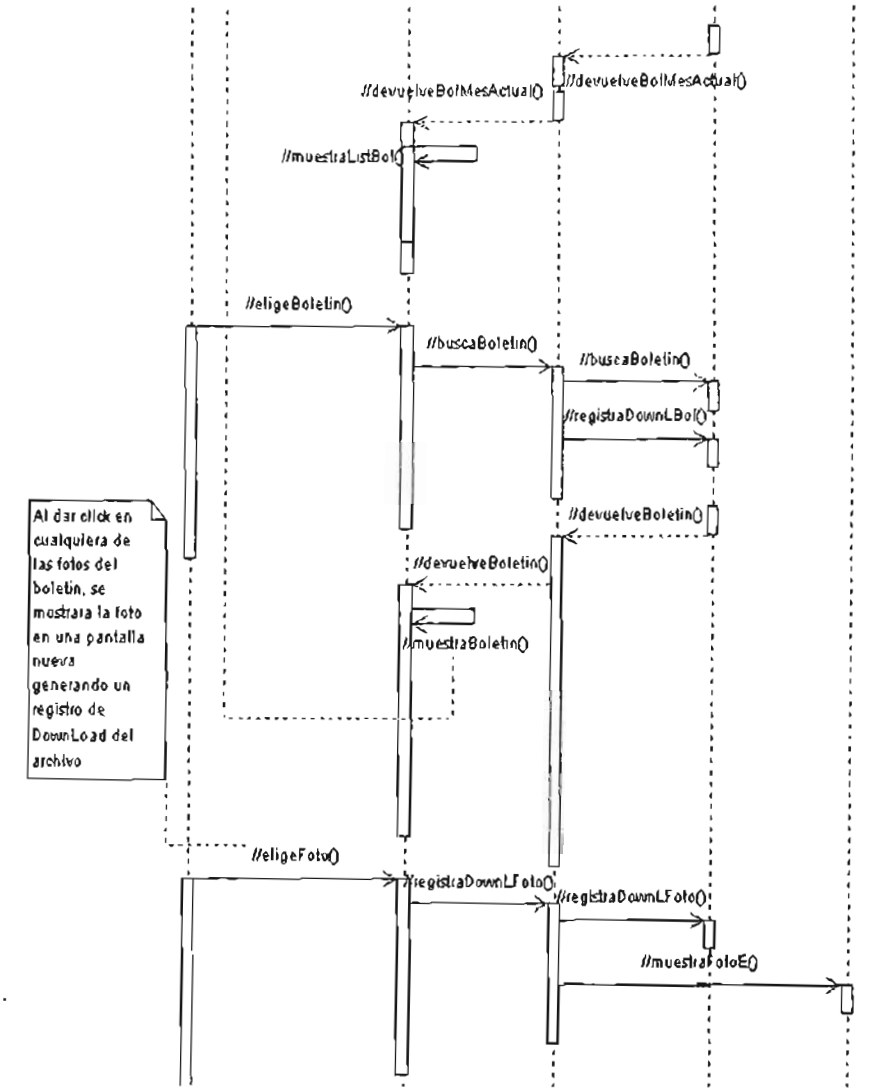


Fig 4.5 Diagrama de secuencia para el Caso de Uso consultar sala de prensa

En cada diagrama se muestra representa un formulario el cual funciona como interfaz entre el usuario y el sistema, todas las formas de ingreso de información se convierten durante el diseño en JSPs, mientras que el controlador se convierten en una clase o servlet, como se muestra en los siguientes diagramas de clase.

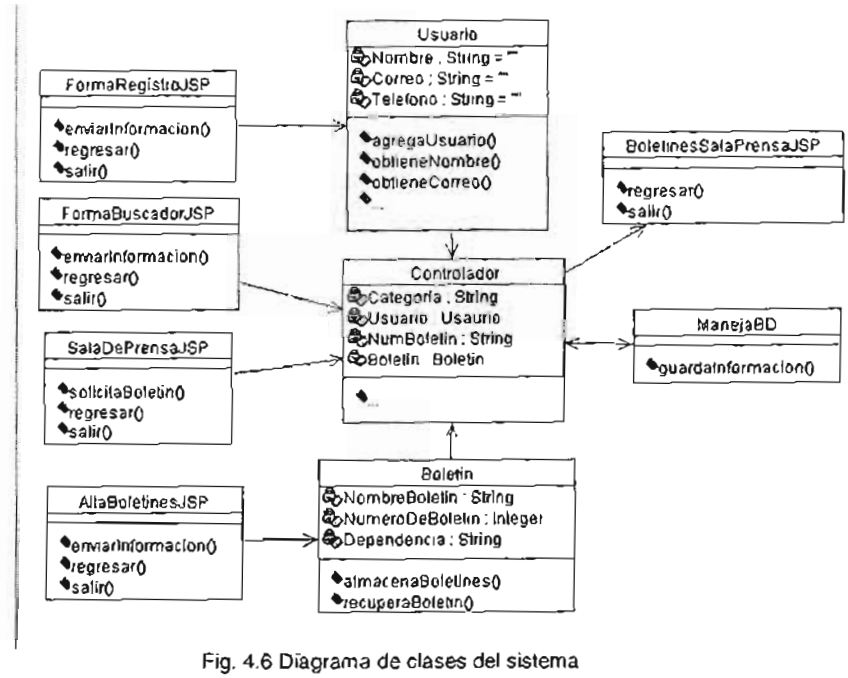


Fig. 4.6 Diagrama de clases del sistema

En la figura 4.7 se muestra el diagrama de componentes que conforman el sistema para DCUNAM. El diagrama de componentes representa a cada conjunto de clases y/o servlet's que conjuntamente llevan a cabo una funcion común.

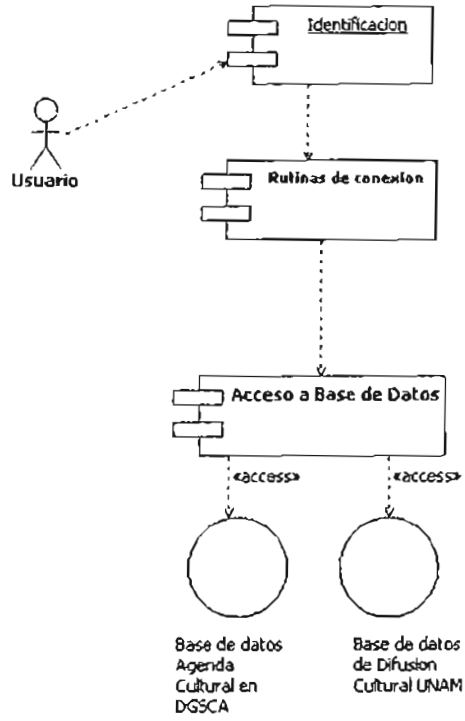


Fig. 4.7 Diagrama de componentes del sistema

Implementación

Servidor de Base de datos y Base de datos (Back - End tier)

Primeramente describiremos la tercera capa conocida como back end, en la cual se encuentra el servidor de BD y la base de datos. Se empleo como servidor de BD a Sybase.

Diseño y construcción de la base de datos

En el desarrollo de este trabajo se utilizo una base de datos ya existente, la base de datos de la agenda cultural la cual tiene el siguiente diseño.

EVENTO	Contenido	
EVEN_clave	Int	Primary key
EVEN_claveclavegrupo	Int	
EVENT_titulo	Varchar(150)	
EVEN_clave	Int	
EVEN_titular1	Varchar(120)	
ENEN_titutal2	Varchar(120)	
EVEN_diain	Int	
EVEN_mesin	Varchar(10)	
LUGAR_clave	Int	
EVEN_informes	Varchar(150)	
EVEN_comentarios	Varchar(150)	
EVEN_url	Varchar(100)	
EVEN_costo	Varchar(60)	
EVEN_anio	Int	
EVEN_sinopsis	Varchar(255)	
EVEN_institucion	Varchar(10)	
EVEN_masinf	Varchar(100)	

Tabla. 4.1

LUGAR	Contenido
LUGAR_clave	Int Primary key
LUGAR_nombre	Varchar(30)

Tabla. 4.2

CATEGORIA	Contenido
CATE_clave	Varchar(30)
CATE_nombre	

Tabla. 4.3

RECINTO	Contenido
RECI_clave	Int Primary key
RECI_nombre	Varchar(100)
RECI_dirección	Varchar(100)

Tabla. 4.4

Creación de la Base de Datos

Además de la BD de la Agenda Cultural, se diseño una Base de Datos en sybase en el servidor Kaos.fi-b.unam.mx, a continuación se muestra el modelo de las tablas agregadas.

Tablas de la Base de Datos

Nombre de la Tabla	Contenido
Usuario	Datos del Usuario
Boletín	Información del Boletín
Boletín_Foto	Contiene los Ids de fotos y boletines relacionados. Esta tabla se utiliza para romper la relacion muchos a muchos entre las tablas Boletín y FotoBoletín
FotoBoletin	Información de las fotografías asociadas al boletín

Promociones	Información sobre los usuarios que se registraron para participar en la promoción del día
DownLoad	Esta tablas contiene los datos referentes a los usuarios y a los archivos que han bajado
PromocionDelDia	Datos de la Promoción del día

Tabla. 4.5

Descripción de las tablas

Usuario		
Id_Usuario	Numeric(9) Identity	Not null
Nombre	Varchar(30)	Not null
Apellido	Varchar(30)	Not null
Edad	Varchar(5)	Not null
Mail	Varchar(30)	null
Escuela	Varchar(30)	null
numCuenta	Varchar(30)	null
Ocupación	Varchar(40)	null
Dependencia	Varchat(30)	null

Tabla. 4.6

Boletin		
Id_Boletin	Numeric(9) Identity	Not null
NombreBol	Varchar(100)	Not null
TituloObra	Varchar(100)	Not null
Autor	Varchar(100)	Not null
Id_Dependencia	Varchar(80)	Not null
Dia	Varchar(5)	
Mes	Varchar(5)	
Anio	Varchar(10)	
Numbol	Varchar(10)	
Liga	Varchar(100)	

Tabla. 4.7

PromociónDelDia		
Id_PromoDia	Numeric Identity	Not null
Titulo	Varchar(100)	Not null
Dia	Varcha(10)	Not null
Mes	Varchar(10)	Not null
Anio	Varchar(10)	Not null
Hora	Varchar(10)	Not null
Id_Dependencia	Varchar(10)	
Comentarios	Varchar(100)	Not null

Tabla. 4.8

Boletin_Foto		
Id_Boletin	Numeric	Not null
Id_Foto	Numeric	Not null

Tabla. 4.9

FotoBoletin		
Id_Foto	Numeric Identity	Not null
NumeroBol	Varchar(10)	Not null
Liga	Varchar(30)	Not null
NombreFoto	Varchar(50)	Not null
Dependencia	Varchat(50)	Not null

Tabla. 4.10

Download		
Id_Usuario	Numeric	Not null
nombbreArchivo	Varchar(100)	Not null
TipoArchivo	Varcha(50)	Not null
LigaArchiva	Varchar(150)	Not null
FechaD	Varchar(15)	Not null
Dependencia	Varchar(25)	Not null

Tabla. 4.11

omoción		
Id_PromoDia	Numeric Identity	Not null
Titulo	Varchar(100)	Not null
Dia	Varcha(10)	Not null
Mes	Varchar(10)	Not null
Anio	Varchar(10)	Not null
Hora	Varchar(10)	Not null
Comentarios	Varchar(100)	Not null

Tabla. 4.12

Modelo Entidad Relación

Base de Datos

Para la creación de las tablas se echó mano de sentencias de SQL, dichas sentencias se clasifican según su finalidad y uso, en tres sublenguajes que son:

- El DDL (Data Description Language) o Lenguaje de Definición de Datos.
- DCL (Data Control Language) o Lenguaje de Control de Datos.
- El Lenguaje de Manipulación de Datos o DML (Data Manipulation Language).

Tenemos entonces que algunos de los queries usados para la creación de las tablas de la base de datos usada en el sistema y que aparecen en la figura.

```
create table Usuairo(
Id_Usuario Numeric(9) Identity Not null,
Nombre Varchar(30) Not null,
Apellido Varchar(30) Not null,
Edad Varchar(5) Not null,
Mail Varchar(30) null,
Escuela Varchar(30)
NumCuenta Varchar(30) null,
Ocupación Varchar(40) null,
Dependencia Varchar(30) null
)

create table Boletin(
Id_Boletin Numeric(9) Identity Not null,
NombreBol Varchar(100) Not null,
TituloObra Varchar(100) Not null,
Autor Varchar(100) Not Null,
Id_Dependencia Varchar(80) Not null,
Dia Varchar(5)
)

create table FotoBoletin(
Id_Foto Numeric Identity Not null,
NumeroBol Varchar(10) Not null,
Liga Varchar(30) Not null,
NombreFoto Varchar(50) not null,
```

```
Id_Dependencia  Varchar(50)  Not null
)

create table BoletinFoto (
Id_Boletín  Numeric  Not null,
Id_Foto  Numeric  Not null
)

create table PromocionDelDia(
Id_PromoDia  Numeric  Identity  Not null,
Titulo  Varchar(100)  Not null,
Dia  Varcha(10)  Not null,
Mes  Varchar(10)  Not null,
Anio  Varchar(10)  Not null,
Hora  Varchar(10)  Not null,
Id_Dependencia  Varchar(10)  not null
)

create table Promociones(
Id_Mail  Numeric  Identity  Not null,
Nombre  Varchar(100)  Not null,
Telefono  Varcha(10)  Not null,
FechaPromo  Varchar(10)  Not null
)

create table DownLoad(
Id_Usuario  Numeric  Not null,
NombreArchivo  Varchar(100)  Not null,
TipoArchivo  Varcha(50)  Not null,
LigaArchiva  Varchar(150)  Not null,
FechaD  Varchar(15)  Not null,
Dependencia  Varchar(25)
)
```

Resultando en la creación de la base de datos que se muestra a continuación

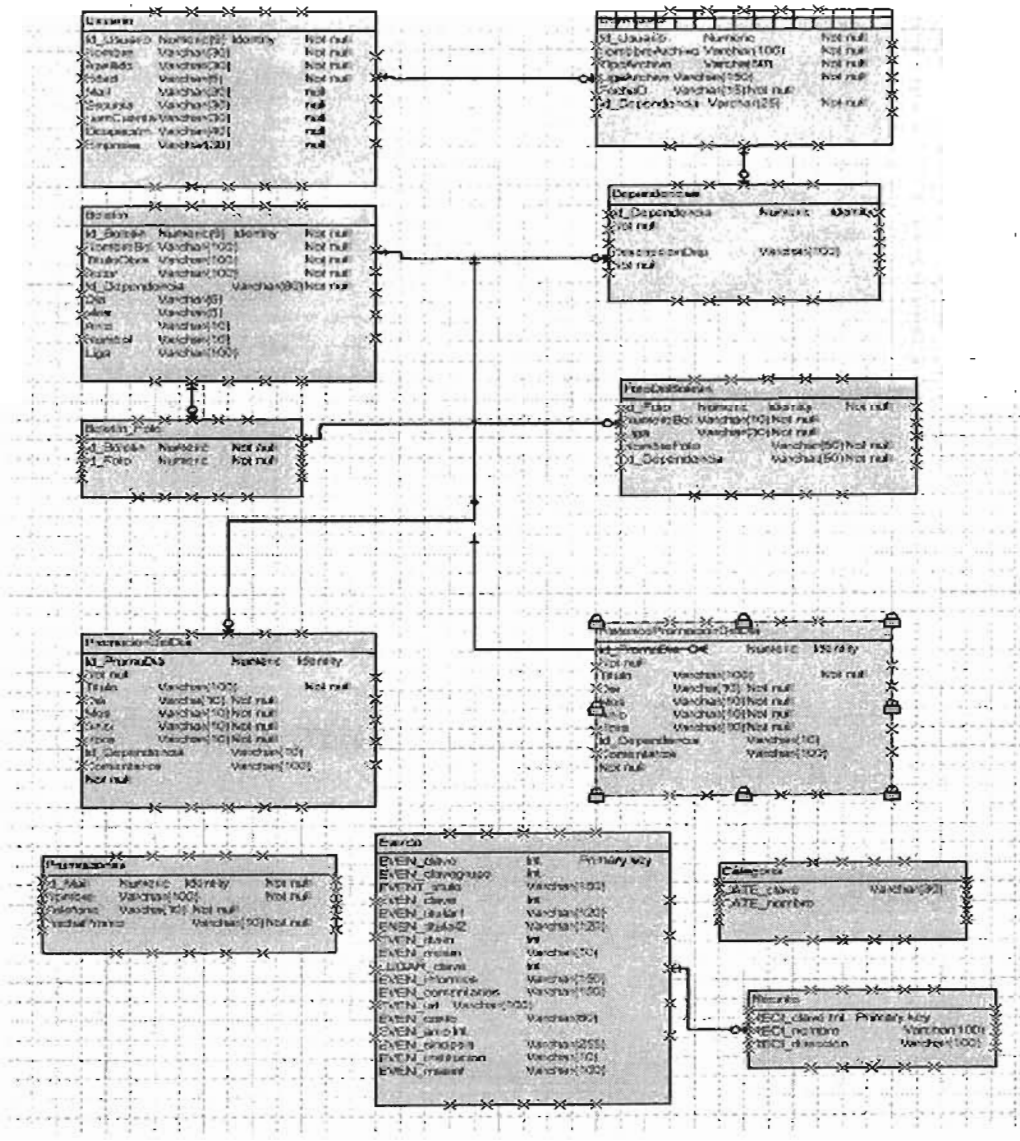


Figura 4.7. Base de Datos y sus tablas.

- **4.5 Diseño Gráfico**

En las siguientes imágenes se muestra el diseño de las pantallas para cada caso de uso, así como su flujo.

Pagina principal

Esta es la pantalla principal de la aplicación. Como se muestra en la imagen, a partir de esta pantalla se inicia la navegación dentro del portal de Difusión cultural
UNAM.

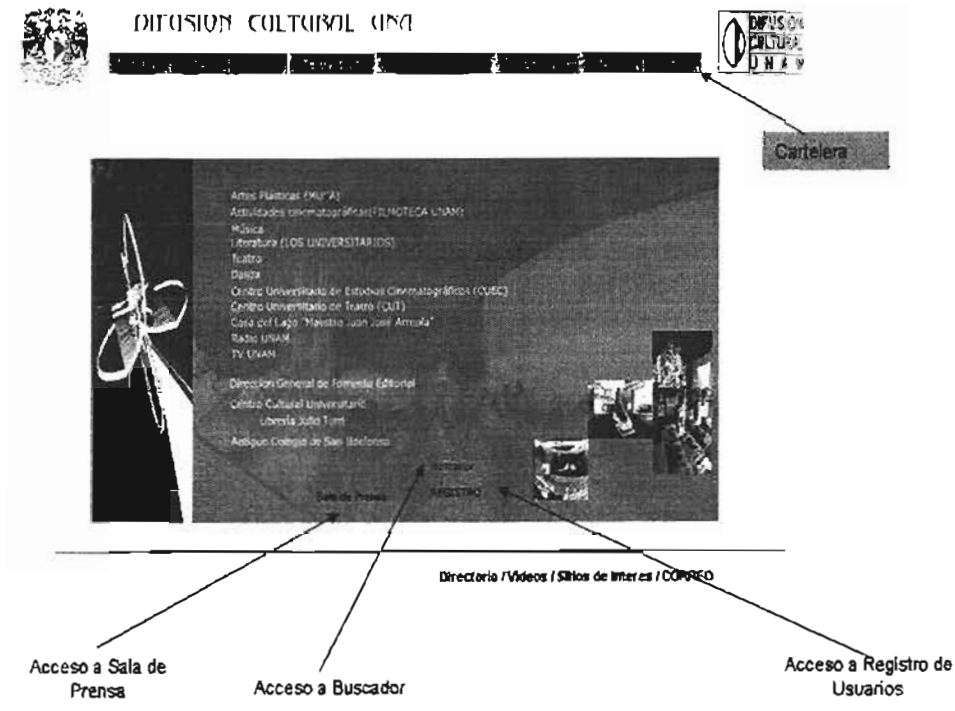


Fig 4.8 Pantalla principal del sito web.

Sala de Prensa

En esta pantalla se muestra en el lado izquierdo una lista con ligas a los meses del año, al ser seleccionada alguna en la parte derecha se despliega una lista con los boletines de ese mes. Para poder el acceso a cada boletín basta seleccionar la liga del titulo del boletín.

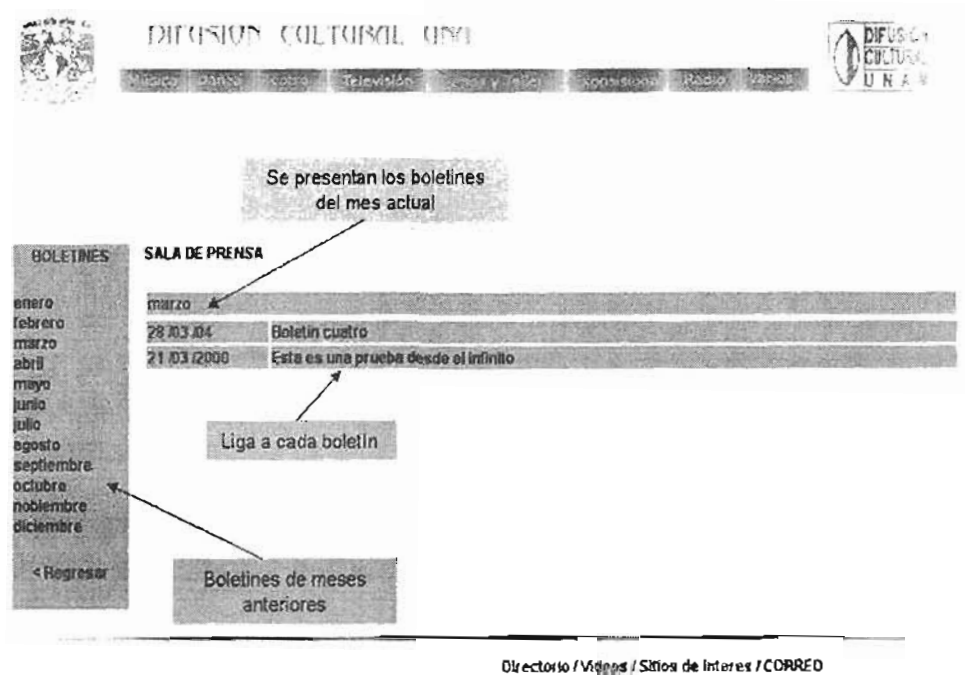


Fig 4.9 Pantalla principal de sala de prensa.

Registro Usuario

La pantalla de registro de usuario muestra una serie de campos de textos en los que se solicita cierta información del usuario. El sistema verifica que los datos tengan infamación valida y muestra una vista previa de los datos.

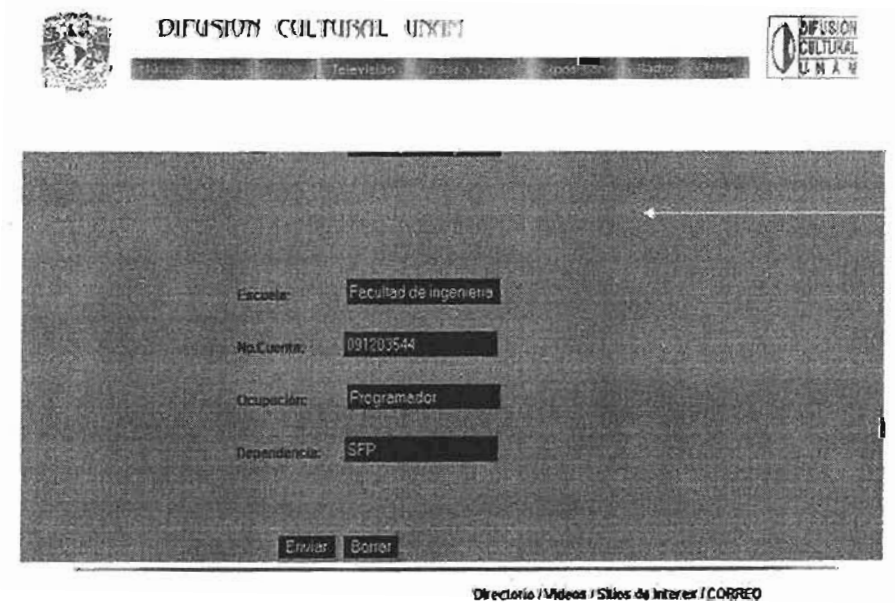


Fig 4.10 Pantalla para el registro de usuario.

Pantalla de previsualización de datos

En esta pantalla el sistema muestra los datos capturados por el usuario, preguntando si es correcta la información para ser enviada, o si se corrige.

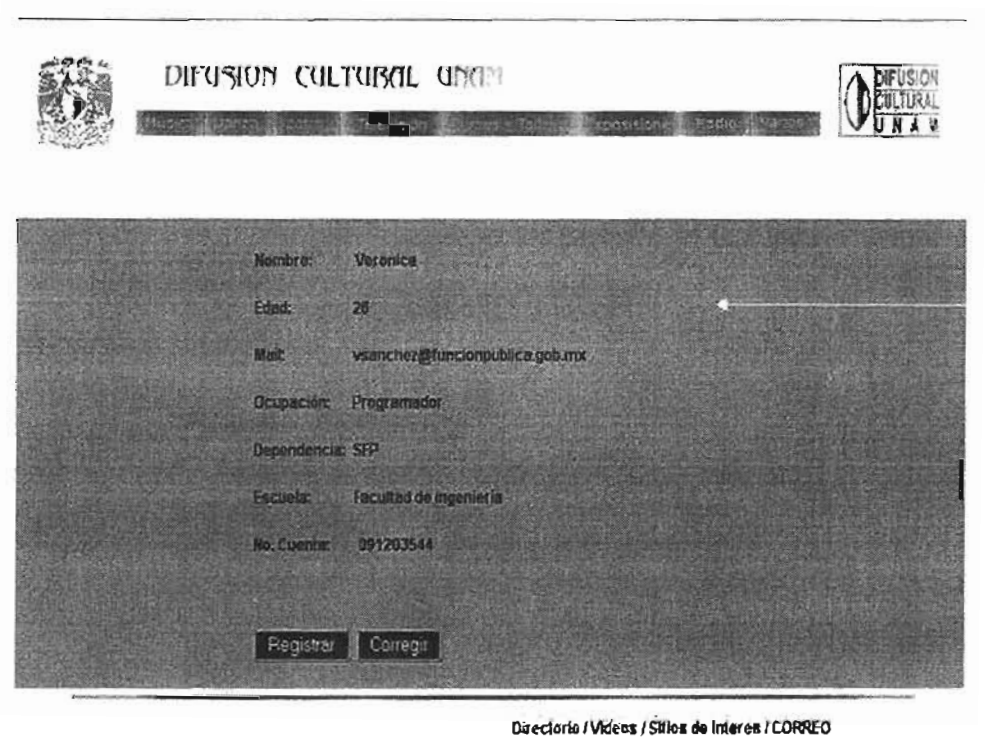


Fig 4.11 Pantalla de previsualizacion de datos para el registro de usuario.

Buscador

Aquí se muestra una lista con las categorías de eventos culturales que se pueden seleccionar para realizar la búsqueda.

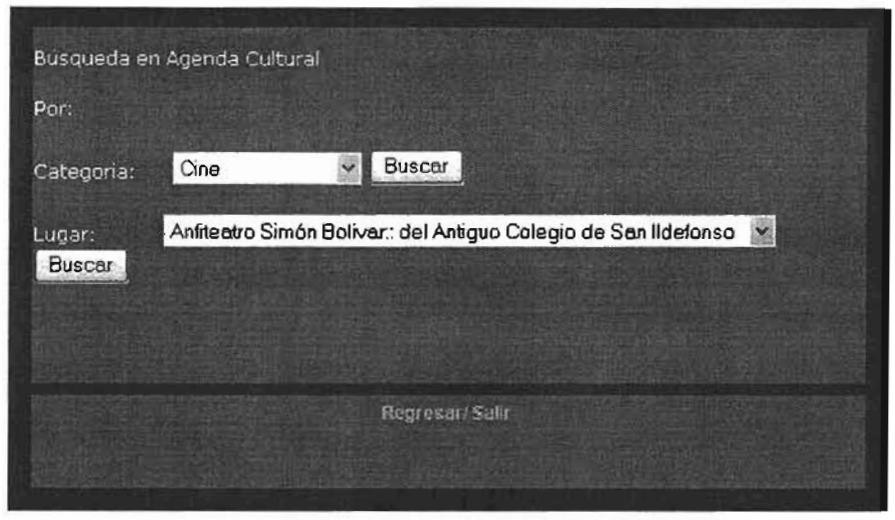


Fig 4.12 Pantalla para el buscador.

Centro Cultural Universitario

Esta es la primera pantalla de la visita guiada por el Espacio Escultórico. Aquí se muestran animaciones en flash de los diferentes recintos con los que cuenta el CCU.

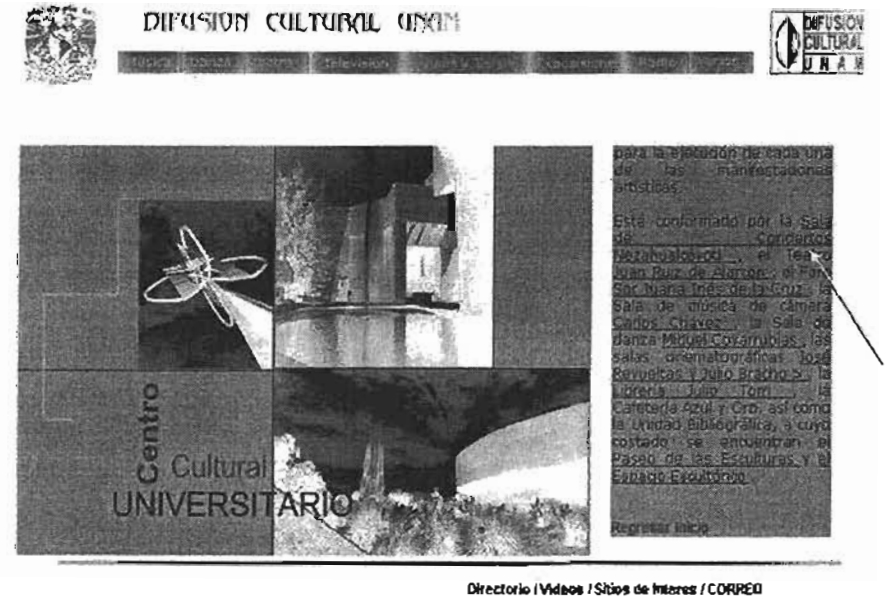


Fig 4.13 Pantalla del centro cultural universitario.

Visita Centro Cultural

En la siguiente imagen se muestran tres diferentes pantallas de la pagina del CCU, Sala Carlos Chávez, Sala Miguel Covarrubias y Espacio Escultórico.

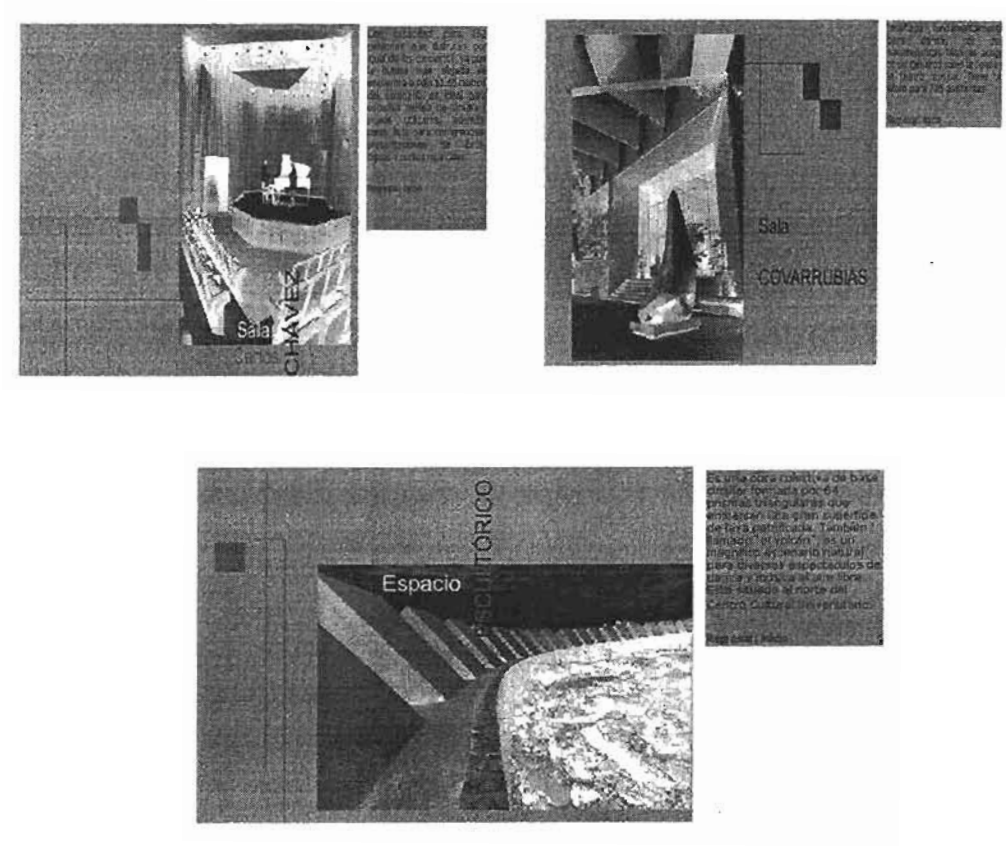


Fig 4.14 Pantallas de Sala Carlos Chávez, Miguel Covarrubias, Espacio Escultórico.

En las siguientes imágenes se muestran otras pantallas de los recintos que conforman el CCU.

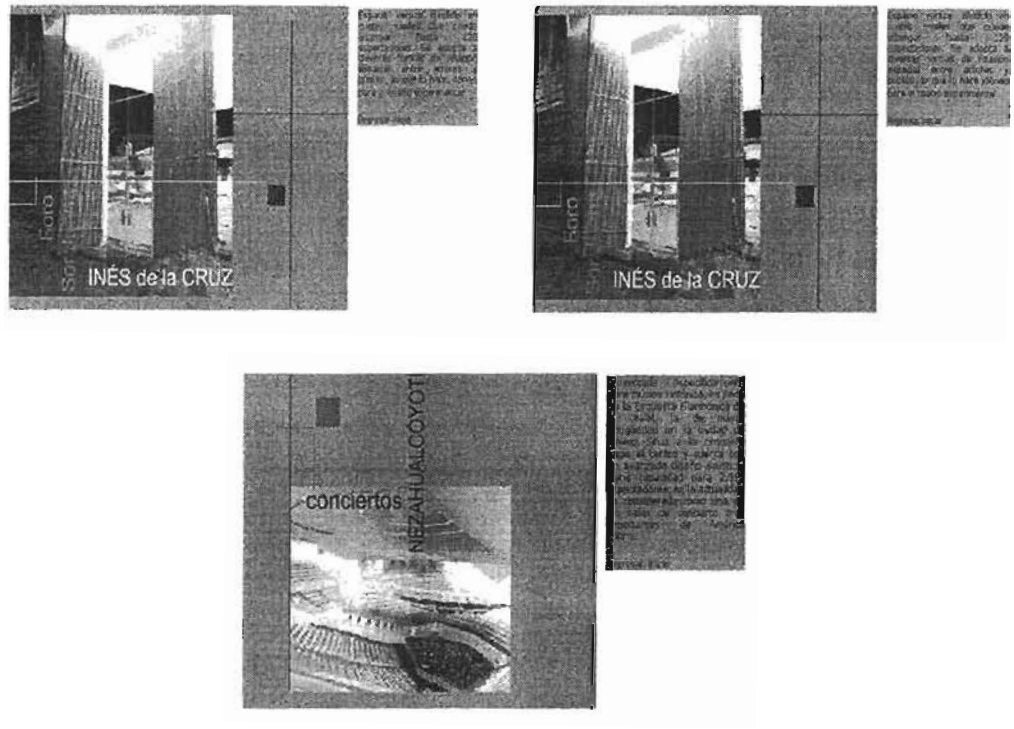


Fig 4.15 Pantallas de Foro Sor Juana Inés de la Cruz, Sala de Conciertos Nezahualcoyotl

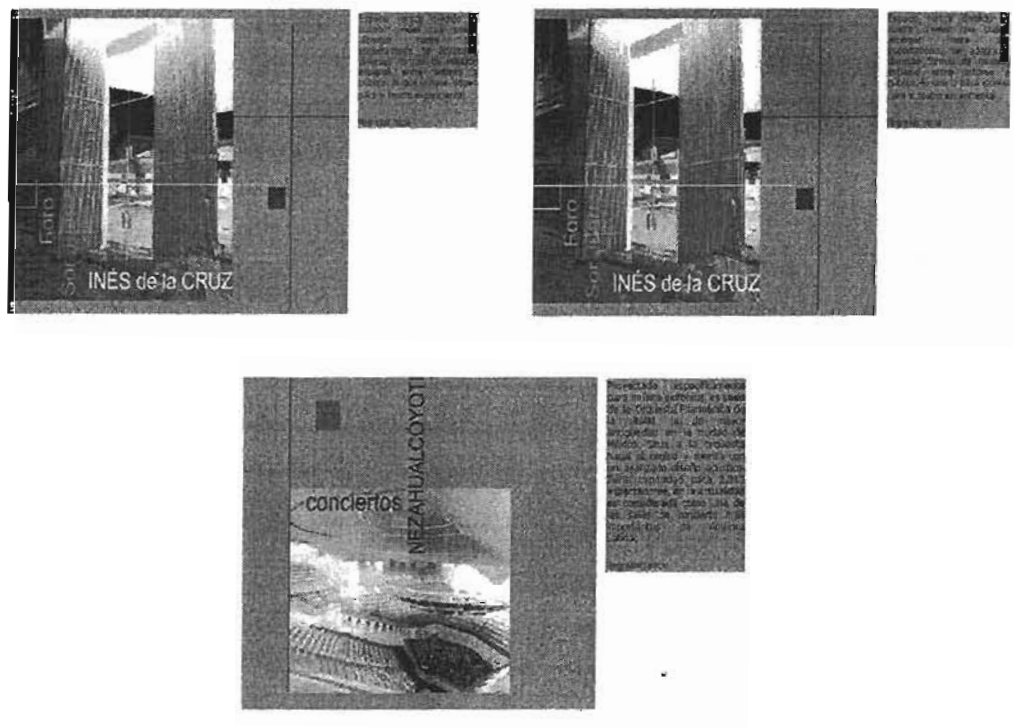


Fig 4.16 Pantallas de Foro Sor Juana Inés de la Cruz, Sala Nezahualcoyotl

La siguiente pantalla muestra una fotografía del interior de la Librería Julio Torri, ubicada en el CCU.

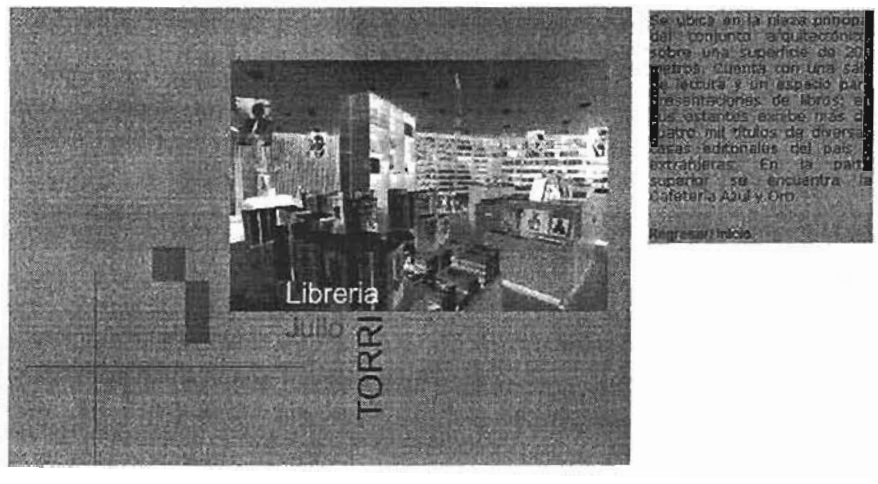


Fig 4.17 Pantallas Librería Julio Torri

Pruebas

Se realizaron pruebas de funcionalidad bajo la supervisión del personal de DCUNAM que será encargado de dar mantenimiento al portal. La documentación generada bajo RUP para llevar a cabo estas pruebas se presenta en el Apéndice C.

Distribución

Para llevar a cabo la implementación de este sistema se generó un archivo WAR para ser entregado a los administradores del servidor de DGSCA. Véase documentación RUP en el Apéndice C.

- ## Conclusiones

Al inicio de este trabajo se planteó la necesidad por parte de DCUNAM de contar con un sistema para llevar a cabo la difusión de eventos culturales, registrar usuarios, así como optimizar el mantenimiento del sitio.

El objetivo principal de esta tesis fue el de la construcción de un sitio Web para agilizar las diversas actividades de DCUNAM. Como objetivo alterno se estableció la utilización de tecnologías de libre distribución como Tomcat, y aunque el Java Development Kit no cumple completamente con los lineamientos de GNU, es de distribución gratuita por lo que se desarrollo con esta tecnología.

En los tres primeros capítulos de este trabajo se desarrollaron temas teóricos en los que se describen características de las principales tecnologías que podrían ser utilizadas para el desarrollo de sistemas para Internet, tales como; arquitectura cliente / servidor, lenguajes de programación, servidores Web y de bases de datos, metodologías de desarrollo de sistemas, así como estructuras para el diseño grafico de las pantallas y su flujo.

Para lograr lo anterior era necesario contar con un sitio Web de publicación de boletines de prensa, registro de usuarios para disponer de una base de datos de correos a través de los cuales difundir los eventos y una pagina de mantenimiento para el sitio. Dadas estas necesidades se realizó en este trabajo una investigación sobre las posibles opciones para desarrollar dicho sistema; encontrando lo siguiente.

La evolución de las tecnologías para el desarrollo de sistemas informáticos ha ido en aumento, el número de opciones en el mercado es muy grande, por lo que se vuelve necesario un amplio conocimiento de estas para poder elegir alguna que nos resulte óptima para las necesidades de nuestro sistema.

De las diversas arquitecturas mostradas se encontró que la arquitectura cliente/servidor tres capas, brinda más grados de libertad en cuanto a la disponibilidad de plataformas, esto es flexibilidad en cuanto al uso de diversas opciones de sistemas operativos, bases de datos, aplicaciones cliente, etcétera.

Se recomienda tener cuidado en cuanto a la elección de la arquitectura del sistema, ya que de no realizar la correcta, puede hacer mas compleja la implementación de la solución. Aunque podemos apoyarnos en las características de flexibilidad de esta estructura para asegurar la posibilidad de contar con alternativas suficientes para tomar decisiones acertadas. Tal vez de las mas importantes ventajas de esta arquitectura es que una aplicación cliente/servidor, idealmente es independiente del hardware y de sistemas operativos; mezclando e igualando estas plataformas.

En este trabajo se llegó a la implementación de esta arquitectura teniendo como configuración la siguiente:

Un cliente: Cuya plataforma (Windows, Unix, Linux) y software (Netscape, Internet Explorer) puede ser variable.

Un servidor Web: Que será el encargado de administrar las peticiones de los clientes, ya sea para generar pagina HTML de respuesta o realizar un proceso de acceso a datos. En nuestro trabajo se utilizó Tomcat para realizar estas tareas.

Un servidor de Base de Datos: Cuya función es procesar el almacenamiento y administrar la información. Debido a los requerimientos de DCUNAM, se utilizó el manejador de base de datos Sybase.

En el área de bases de datos también se observó que existe gran variedad de manejadores de Bases de Datos compitiendo en el mercado y aunque todos cuentan con características similares, cambian sólo algunos detalles como la sintaxis en el lenguaje para realizar consultas, capacidad de almacenamiento, velocidad de respuesta. Oracle, por ejemplo cuenta con PL / SQL un lenguaje para desarrollo Web, pero en general todos prestan los servicios necesarios para la administración de información.

Por consiguiente, los criterios de elección de manejador de bases de datos son diversos, pueden ser económicos, de convenios, técnicos, de requerimientos, de integración con otras tecnologías, etcétera. En este caso se eligió a Sybase entre las opciones presentadas, como servidor de base de datos, además de las características presentadas en este trabajo, tales como:

- Restricciones.
- *Triggers.*
- Reglas.
- Integridad de transacciones.

Que proveen poder y flexibilidad, siendo el recurso que fue asignado por las personas de DGSCA para el desarrollo de este sistema, utilizar recursos ya existentes nos beneficia en el sentido de que ya no será necesario llevar a cabo instalaciones y migraciones de la Base de Datos de Agenda Cultural que ya existe, además de que uno de los requerimientos era trabajar con la BD ya existente, por lo que solo se crearon las tablas necesarias para implementar la nueva funcionalidad del portal.

Los lenguajes para programar sistemas de Internet son numerosos sin embargo nos encontramos con alguna razones para no usar varios de ellos, empecemos con los CGI's, el gran inconveniente de estos es el rendimiento, ya que por cada petición se crea una nueva copia del programa en la memoria del servidor. Así, si accede un gran número de usuarios de forma simultánea, se produce una disminución en la eficiencia de dicho servidor. Los programas CGI's pueden presentar agujeros de seguridad de manera intencional o no, pueden mostrar información sobre el sistema que puede ayudar a terceros a comprometerlo.

La tecnología ASP de Microsoft es ampliamente dependiente de la plataforma, PHP aunque es un lenguaje bondadoso no cuenta con características como independencia de plataforma, o capacidad para programación de Servlets, ni programación orientada a objetos en un sentido amplio tal como lo hace por ejemplo JAVA, características que si posee el lenguaje elegido en esta ocasión, una tecnología ampliamente utilizada hoy en día para el desarrollo de sistemas, son los Java Servlets y JSP's, basados en el lenguaje JAVA, cuya utilidad no solo se reduce a sistemas de PC's, sino a sistemas para teléfonos celulares, electrodomésticos digitales, etc.

Su funcionamiento de la tecnología web basada en JSP's y Servlets, es similar al de los CGI's, con la diferencia de que los programas que se ejecutan en el servidor están escritos en lenguaje

Java. Esta tecnología soluciona el problema del rendimiento gracias a que sólo mantiene una imagen del programa en memoria, aunque existan múltiples peticiones. Otra importante ventaja es la portabilidad que ofrece utilizar *Java*. Asegura la independencia de la plataforma, al contrario de lo que ocurre con ASP. Además utiliza un lenguaje totalmente orientado a objetos y mucho más robusto que los lenguajes scripts como PHP. Además este lenguaje tiene soporte para la mayoría de las Bases de Datos existentes mediante el uso de un driver específico para cada Base de datos.

Aunque en el mercado existen muchos servidores Web y entre los que se describieron en este trabajo se encuentra alguno con importantes características, como podría ser el Iplanet, su costo es alto, el IIS de Microsoft es dependiente de la plataforma, APACHE aunque se trata de un buen servidor Web, no cuenta con soporte para JSP's, por tal razón se eligió al contenedor de JSP's Tomcat, el cual es de distribución libre, además de contar con las características necesarias para soportar una aplicación como la que se desarrollo a lo largo de este trabajo. Además del costo, el tiempo de adquisición es mínimo ya que este software se puede obtener de Internet. Como principales características de tomcat que lo hacen ideal para este tipo de desarrollos se encontraron las siguientes:

- Es un contenedor de servlets de acceso libre y gratuito.
- Es parte del servidor web o del servidor de aplicaciones, y puede trabajar stand alone.
- Se encarga de realizar todo el trabajo de las conexiones, descodificación de mime y gestionar la vida del servlet.

Como se vió a lo largo del trabajo, las metodologías de desarrollo de sistemas, de la misma manera que las arquitecturas, lenguajes y servidores, han evolucionado a lo largo del tiempo. Desde las usadas para desarrollar sistemas con lenguajes de programación estructurados, hasta las metodologías orientadas al desarrollo de sistemas con lenguajes orientados a objetos.

De la misma manera que las otras tecnologías, elegir una adecuada para el desarrollo depende de varios factores, por lo que es vuelve necesario tener conocimiento por lo menos básico de varias de ellas y elegir la que se adecue a las necesidades y circunstancias del sistema a desarrollar, algunos puntos a tomar en cuenta a la hora de elegir podrían ser:

- El tiempo de desarrollo con el que se cuenta para el desarrollo.
- Tipo de sistema a desarrollar.
- Herramientas con las que se cuenta para el modelado del sistema.

Una vez considerados estos puntos, se puede llevar a cabo una mejor elección de la metodología. Cabe señalar que aún cuando se haya elegido una específica, esta puede ser adaptada a las necesidades del desarrollo del sistema, por ejemplo, en nuestro caso cuya elección ha sido RUP, se omitió la realización de algunos documentos por la sencilla razón de que eran redundantes o la información de estos estaba contenida en otros documentos.

Se pueden encontrar otras metodologías, como SunTone, propia de Sun Microsystems, la cual ha tomado algunas características de RUP para adaptarlas a su propia metodología, orientándose más a los servicios.

Actualmente las tendencias del desarrollo de software se direccionan hacia el campo de Orientado a Objetos, tecnologías como JAVA de Sun, y .NET de Microsoft, compiten por el liderazgo, ambos orientados a objetos. Ante esta situación para el trabajo actual ya elegidos un lenguaje OO como java y un servidor como TomCat para soportar esta tecnología, lo que resta es la elección de una metodología también OO, en este caso, fue elegida la de RUP, descrita en el capitulo tres. Aunque existen otras metodologías OO, RUP, por sus características ha demostrado ser la más práctica y funcional hasta el momento, en cuanto a proceso de

desarrollo, por esta razón es la aplicada en el desarrollo de este trabajo. Entre las principales características de este tipo de metodologías se encuentran:

- Son iterativas e incrementales.
- Fácil de dividir el sistema en varios subsistemas independientes.
- Se fomenta la reutilización de componente.
- Se disminuye tiempos de desarrollo.

En cuanto al diseño grafico nos encontramos de igual manera con una gran variedad de opciones: lineal, jerárquica, mixta, etc. La estructura de diseño que fue elegida en nuestro caso cuenta con una combinación de todas las descritas en la primer sección del capitulo 3, en el que se habla del diseño de un sitio Web, se opto por la combinación de todas ya que existen secciones del sitio que funcionaran mucho mejor si el flujo es lineal, y otras que necesitaran una estructura de jerarquía, los frames también son un elemento esencial en el diseño de este portal, ya que la presentación de información, ligas y secciones será mejor distribuida a través de este mecanismo.

Para la creación de las animaciones se utilizo Flash y DreamWeaver así como PhotoShop para la edición de imágenes y fotografías.

Aunque hasta ahora, se ha mencionado que existe una gran cantidad de tecnologías existentes y que una correcta elección de estas depende de varios factores (económicos, infraestructura existente, tipo de sistema a desarrollar, tiempo de desarrollo, herramientas con las que se cuenta, etc.), es importante mencionar que las tendencias hoy en día son los desarrollos de sistemas utilizando lenguajes y metodologías orientado a objetos, las cuales presenta entre sus ventajas las siguientes:

Como los objetos son portables (teóricamente) mientras que la herencia permite la reusabilidad del código orientado a objetos, es más sencillo modificar código existente porque los objetos no interaccionan excepto a través de mensajes; en consecuencia un cambio en la codificación de un objeto no afectará la operación con otro objeto siempre que los métodos respectivos permanezcan intactos.

La introducción de tecnología de objetos como una herramienta conceptual para analizar, diseñar e implementar aplicaciones permite obtener aplicaciones más modificables, fácilmente extendibles y a partir de componentes reusables. Esta reusabilidad del código disminuye el tiempo que se utiliza en el desarrollo y hace que el desarrollo del software sea mas intuitivo porque la gente piensa naturalmente en términos de objetos más que en términos de algoritmos de software.

Desarrollo

Durante el desarrollo del sistema nos encontramos con dificultades en cuanto a la coordinación del trabajo con el personal de DGSCA. Debido a esto no fue posible llevar a cabo en su totalidad cada una de las iteraciones que establece el RUP, por esta razón el sistema se desarrollo en una iteración y las pruebas fueron realizadas hasta el término de la programación de los módulos, obteniéndose a partir de estas pruebas algunas observaciones y modificaciones en el sistema. Aunque los cambios no fueron de gran impacto, lo ideal hubiera sido realizar las pruebas por cada uno de los modulo e ir corrigiendo por cada una de las iteraciones. Sin embargo, se aplico en su mayor parte el proceso de desarrollo que establece

RUP, se generaron especificaciones y diagramas de casos de uso, diagramas de secuencia y de clases, así como el diagrama de componentes del sistema.

Implementación

La implementación será realizada ¿?

Después de la implementación de este sistema, se espera una mejora en los siguientes puntos:

Facilidad en el mantenimiento del sitio.

Reducción del trabajo y tiempo administrativo.

Mejora del proceso publicación de boletines y envió de cartelera vía e-mail.

Las ventajas de la tecnología con la que fue desarrollado este sistema, es que es multiplataforma, es decir, la funcionalidad no cambia independientemente de la plataforma del sistema operativo con la que se vaya a utilizar. Algunas de sus principales características se muestran a continuación.

- Alto desempeño y escalabilidad
- Arquitectura Cliente/Servidor

APENDICE A

Manuales de Instalación

INSTALACION DE TOMCAT 3.2.2
La base para utilizar cualquier producto que implementa "java" es el JDK (Java Development Kit) de la plataforma correspondiente. En este caso se hará sobre la plataforma LINUX, aunque los pasos de instalación después de la instalación del "JDK" no distan mucho de las diversas plataformas.
1.- Se debe descargar el JDK del sitio de Sun en: *http://java.sun.com/j2se/1.4*
2.- Basta con ejecutar el archivo, esto extraerá el JDK en el directorio actual generando un directorio llamado, por ejemplo, jdk1.4. Se recomienda cambiar el nombre de este directorio a jdk y moverlo al directorio /usr/local/
3.- Posteriormente se debe definir una variable de ambiente muy importante la cual le indica al sistema la ubicación del JDK, esta variable es JAVA_HOME la cual debe ser agregada a /etc/bashrc
Los pasos anteriores son los necesarios para el JDK, en lo que se refiere a la instalación de Tomcat.
Tomcat
1.- Descargar la versión binaria de Tomcat en: *http://jakarta.apache.org/tomcat*.
2.- Descomprimir el archivo Tar de Tomcat en /usr/local, esto genera un directorio llamado, por ejemplo, jakarta-tomcat-3.2.2, dependiendo del número de versión. Se recomienda cambiar el nombre de este directorio a simplemente "tomcat".
3.- Posteriormente se debe definir una variable de ambiente la cual indicara al sistema la ubicación del Tomcat, esta variable se llama TOMCAT_HOME, la cual debe ser agregada a etc/bashrc

Configuración Local

Ejecución y Pruebas
Una vez efectuados los pasos anteriores es posible realizar pruebas iniciales sobre Tomcat, para esto se debe ejecutar lo siguiente:
1.- Hay descender al directorio bin de Tomcat (/usr/local/Tomcat/bin) y ejecutar startup.sh

```
2004-05-13 01:25:33 - ContextManager: Adding context Ctx( )
2004-05-13 01:25:33 - ContextManager: Adding context Ctx( /test )
2004-05-13 01:25:33 - ContextManager: Adding context Ctx( /difusion )
2004-05-13 01:25:40 - PoolTcpConnector: Starting HttpConnectionHandler on 8888
2004-05-13 01:25:40 - PoolTcpConnector: Starting Ajp12ConnectionHandler on 8107
```

Se debe observar algo similar al desplegado anterior. Cabe mencionar que la pantalla permanecerá inhabilitada, este congelamiento en pantalla se debe a que aquí será enviado todo error detectado por Tomcat, así queda la configuración "default" de Tomcat.

Para detener la ejecución de Tomcat es necesario ejecutar el comando shutdown.sh que se encuentra en el mismo directorio que startup.sh.

Estructura de Directorios de Tomcat

Nombre de	Descripción

Directorio	
Bin	Contiene los scripts de arrancar/parar
Conf	Contiene varios ficheros de configuración incluyendo server.xml (el fichero de configuración principal de Tomcat) y web.xml que configura los valores por defecto para las distintas aplicaciones desplegadas en Tomcat.
Doc	Contiene varia documentación sobre Tomcat (Este manual, en Inglés).
Lib	Contiene varios ficheros jar que son utilizados por Tomcat. Sobre UNIX, cualquier fichero de este directorio se añade al classpath de Tomcat.
Logs	Aquí es donde Tomcat sitúa los ficheros de diario.
Src	Los ficheros fuentes del API Servlet. ¡No te excites, todavía! Estoa son sólo los interfaces vacios y las clases abstractas que debería implementar cualquier contenedor de servlets.
webapps	Contiene aplicaciones Web de Ejemplo.

Adicionalmente podemos, o Tomcat creará, los siguientes directorios:

Nombre de Directorio	Descripción
Work	Generado automáticamente por Tomcat, este es el sitio donde Tomcat sitúa los ficheros intermedios (como las páginas JSP compiladas) durante su trabajo. Si borramos este directorio mientras se está ejecutando Tomcat no podremos ejecutar páginas JSP.
classes	Podemos crear este directorio para añadir clases adicionales al classpath. Cualquier clase que añadamos a este directorio encontrará un lugar en el classpath de Tomcat.

Ficheros de Configuración de Tomcat

La configuración de Tomcat se basa en dos ficheros:

1. server.xml - El fichero de configuración golbal de Tomcat.
2. web.xml - Configura los distintos contextos en Tomcat.

Elemento	Descripción
Server	El elemento superior del fichero server.xml. Server define un servidor Tomcat. Generalmente no deberíamos tocarlo demasiado. Un elemento Server puede contener elementos Logger y ContextManager.
Logger	Este elemento define un objeto logger. Cada objeto de este tipo tiene un nombre que lo identifica, así como un

	path para el fichero log que contiene la salida y un verbosityLevel (que especifica el nivel de log). Actualmente hay loggeres para los servlets (donde va el ServletContext.log()), los ficheros JSP y el sistema de ejecución tomcat.
ContextManager	Un ContextManager especifica la configuración y la estructura para un conjunto de ContextInterceptors, RequestInterceptors, Contexts y sus Connectors. El ContextManager tiene unos pocos atributos que le proporcionamos con: 1. Nivel de depuración usado para marcar los mensajes de depuración 2. La localización base para webapps/, conf/, logs/ y todos los contextos definidos. Se usa para arrancar Tomcat desde un directorio distinto a TOMCAT_HOME. 3. El nombre del directorio de trabajo. 4. Se incluye una bandera para controlar el seguimiento de pila y otra información de depurado en las respuestas por defecto.
ContextInterceptor & RequestInterceptor	Estos interceptores escuchan ciertos eventos que sucenden en el ContextManager. Por ejemplo, el ContextInterceptor escucha los eventos de arrancada y parada de Tomcat, y RequestInterceptor mira las distintas fases por las que las peticiones de usuario necesitan pasar durante su servicio. El administrador de Tomcat no necesita conocer mucho sobre los interceptores; por otro lado, un desarrollador debería conocer que éste es un tipo **global** de operaciones que pueden implementarse en Tomcat (por ejemplo, loggin de seguridad por petición).
Connector	El Connector representa una conexión al usuario, a través de un servidor Web o directamente al navegador del usuario (en una configuración independiente). El objeto Connector es el responsable del control de los threads en Tomcat y de leer/escribir las peticiones/respuestas desde los sockets conectados a los

	distintos clientes. La configuración de los conectores incluye información como: 1. La clase handler. 2. El puerto TCP/IP donde escucha el controlador. 3. el backlog TCP/IP para el server socket del controlador.
GlobalNamingResources, Resource y ResourceParams	Anidado dentro de los elementos <GlobalNamingResources> es posible definir recursos JNDI ("Java Naming Directory Interface") que es una especificación que permite localizar información en distintos directorios distribuidos, para ser utilizados globalmente en Tomcat. Lo anterior evita que estos recursos tengan que ser declarados a nivel de WAR de manera individual. A través de <Resource> es como se define el tipo de recurso JNDI que será utilizado y mediante <ResourceParams> se especifican los parámetros específicos que tomará el recurso en dicha instancia de Tomcat.
Service	Este parámetro permite configurar Tomcat para diferentes modalidades de ejecución, en el archivo server.xml "Default" se definen dos modalidades a través del atributo name, la definición asignada name="Tomcat" es empleada para ejecutar Tomcat individualmente.

Context

Cada Context representa un path en el árbol de tomcat donde situarnos nuestra aplicación web. Un Context Tomcat tiene la siguiente configuración:

1. El path donde se localiza el contexto. Este puede ser un path completo o relativo al home del ContextManager.
2. Nivel de depuración usado para los mensaje de depuración.
3. Una bandera reloadable. Cuando se desarrolla un servlet es muy conveniente tener que recargar el cambio en Tomcat, esto nos permite corregir errores y hacer que Tomcat pruebe el nuevo código sin tener que parar y arrancar. Para volver a recargar el servlet seleccionamos la bandera reloadable a true. Sin embargo, detectar los cambios consume tiempo; además, como el nuevo servlet se está cargando en un nuevo objeto **class-loader** hay algunos casos en los que esto lanza errores de forzado (cast). Para evitar estos problemas, podemos seleccionar la bandera reloadable a false, esto desactivará esta característica.

Instalación Sybase

DRIVER JDBC PARA SYBASE

El driver utilizado en el desarrollo de este trabajo es **jconecctor45** el cual se puede descargar de la siguiente dirección:
http://www.sybase.com/

Para utllizar este driver es necesario guardar el archivo .zip en la carpeta lib de la instalación de tomcat, y se agrega la ruta de esta carpeta al classpath.

A continuación de muestra un bloque de código que implementa el jconnector45 para la conexión a la BD.

```
public boolean conecta() throws SQLException, Exception{

   private        String url = "jdbc:sybase:Tds:kaos.fi-b.unam.mx:7101";
   private        String userid = "usuario";
   private        String pwd = "password";
   private Connection conexion;
   private Statement sentencia;
        try{
                Driver driver = (Driver)
Class.forName("com.sybase.jdbc.SybDriver").newInstance();
                DriverManager.registerDriver(driver);
                conexion = DriverManager.getConnection( url, userid, pwd );

      } catch(SQLException e) {
        System.out.println("Error al intentar la conexion:");
        System.out.println("SQLState: "+e.getSQLState());
        System.out.println("Mensaje:"+e.getMessage());
        System.out.println("Codigo de error:"+e.getErrorCode());
        System.out.println("***");
        e.printStackTrace();
        while(e!=null) {
        System.out.println("SQLState: "+e.getSQLState());
        System.out.println("Mensaje:"+e.getMessage());
        System.out.println("Codigo de error:"+e.getErrorCode());
        e=e.getNextException();
        System.out.println("***");
          } // Fin while
          return false;
      } // Fin del try-catch

        return true;
  }//End of connecta()
```

Configuración JDK

APENDICE B

Manual Técnico de la Aplicación

En este documento tiene como propósito, describir las características necesarias para realizar la instalación del Sistema para difusión cultural UNAM, es decir, los requerimientos técnicos indispensables, así como la estructura del árbol de directorios.

Requisitos hardware y software

El Sistema para DCUNAM ha sido desarrollado utilizando tecnología Java, por ejemplo:

- Java Server Pages, JSPs, hojas de estilo (.css)
- Java Scripts
- Clases

Aunque una de las ventajas de la tecnología con la que fue desarrollado este sistema es que es multiplataforma, es decir, la funcionalidad no cambia, independientemente de la plataforma del sistema operativo con la que se vaya a utilizar, se recomienda contar con un servidor Linux.

Se recomienda obtener la versión 3.0 de Tomcat como contenedor de las clases y JSP's de este sistema por ser uno de los servidores Web más utilizados ya que cuenta con las siguientes características:

- Alto desempeño y escalabilidad
- Arquitectura de multiproceso y multitarea

Bases De Datos

Se debe contar con una base de datos en la última de Sybase versión llamada difusión; para mayor referencia ver el documento de modelado de esta: difusion.mdl. Se deberán crear en esta base de datos las siguientes tablas, Usuario, Boletin_Foto, FotoBoletin, Promociones, DownLoad, PromocionDelDia, para información mas especifica referirse al capitulo referente a base de datos.

Organización de directorios

Una vez que se cuenta con los requerimientos técnicos de hardware y software mencionados con anterioridad, se procede a colocar los JSPs, JScripts, Hojas de Estilo, Clases y Código fuente dentro de los niveles y carpetas correspondientes.

Archivos en path /webapps/difusion/cultural/

La siguiente imagen muestra de manera gráfica la conexión vía SSH al Web Server instalado, el path que se debe considerar para colocar los archivo de tipo JSP, imágenes, SWF, y HTML.

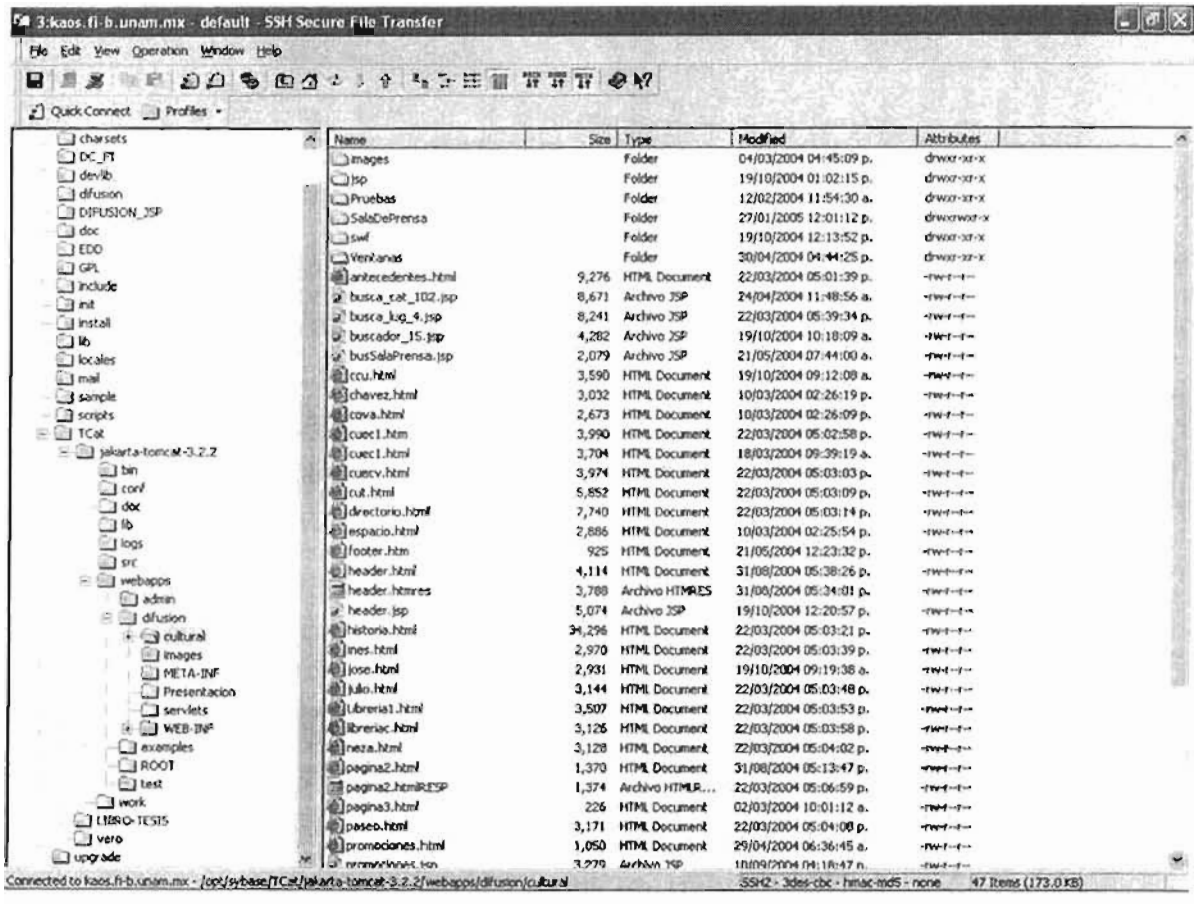


Imagen 1 Contenido del directorio /wepapps/difusion/cultural/

El directorio cultural deberá tener las carpetas:

1. images
2. jsp
3. SalaDePrensa
5. Swf

Images

Dentro de esta carpeta, las imágenes necesarias para mostrar en los diversos archivos JSP's y HTML del sistema.

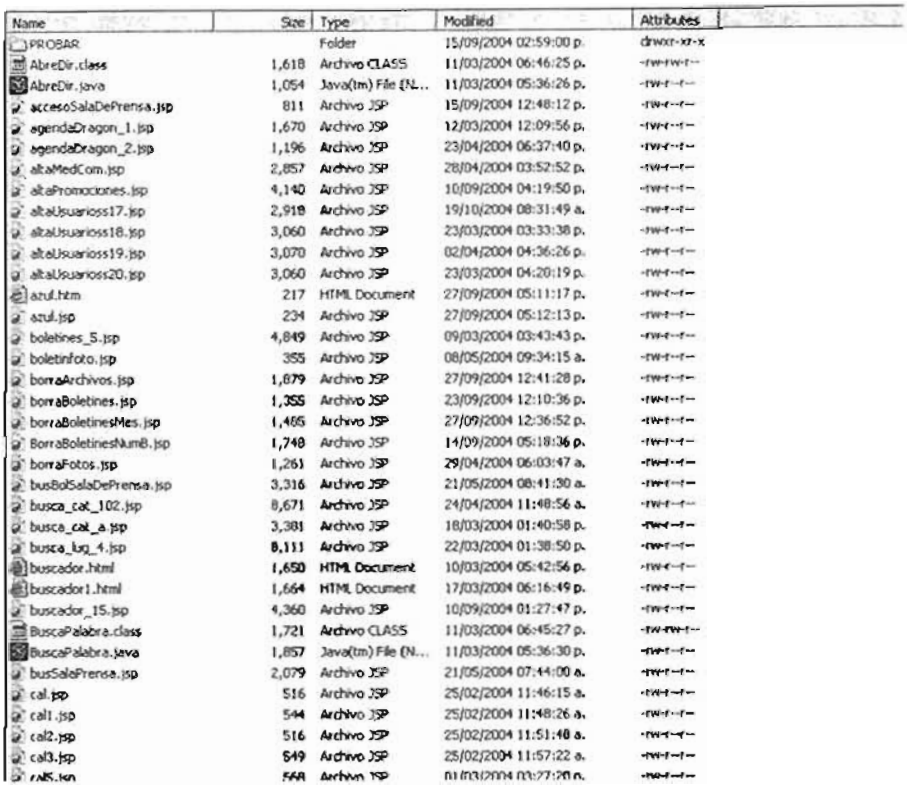
Name	Size	Type	Modified	Attributes
Pruebas		Folder	27/01/2005 12:37:44 p.	drwxr-xr-x
cine1.jpg	9,937	Archivo JPG	03/03/2004 09:51:55 a.	-rw-r--r--
cine1_enc.jpg	10,311	Archivo JPG	03/03/2004 09:52:32 a.	-rw-r--r--
cursos1.jpg	11,258	Archivo JPG	03/03/2004 03:14:21 p.	-rw-r--r--
cursos1_enc.jpg	12,310	Archivo JPG	03/03/2004 10:05:56 a.	-rw-r--r--
danza1.jpg	9,901	Archivo JPG	03/03/2004 09:53:16 a.	-rw-r--r--
danza1_enc.jpg	10,588	Archivo JPG	03/03/2004 09:53:46 a.	-rw-r--r--
difazul1.jpg	16,429	Archivo JPG	03/03/2004 04:14:17 p.	-rw-r--r--
expo1.jpg	11,475	Archivo JPG	03/03/2004 10:01:02 a.	-rw-r--r--
expo1_enc.jpg	12,278	Archivo JPG	03/03/2004 10:01:55 a.	-rw-r--r--
lin2.jpg	9,374	Archivo JPG	03/03/2004 10:14:37 a.	-rw-r--r--
lin3.jpg	9,993	Archivo JPG	03/03/2004 10:15:49 a.	-rw-r--r--
midifusionoro.jpg	20,190	Archivo JPG	03/03/2004 04:00:37 p.	-rw-r--r--
miunamoro.jpg	21,907	Archivo JPG	03/03/2004 03:54:46 p.	-rw-r--r--
musica1.jpg	10,054	Archivo JPG	03/03/2004 09:49:59 a.	-rw-r--r--
musica1_enc.jpg	10,270	Archivo JPG	03/03/2004 09:50:54 a.	-rw-r--r--
radio1.jpg	9,937	Archivo JPG	03/03/2004 03:18:52 p.	-rw-r--r--
radio1_enc.jpg	10,324	Archivo JPG	03/03/2004 09:59:27 a.	-rw-r--r--
Style.css	4,799	Documento de ...	03/03/2004 12:47:40 p.	-rw-r--r--
teatro1.jpg	9,809	Archivo JPG	03/03/2004 02:58:46 p.	-rw-r--r--
teatro1_enc.jpg	10,315	Archivo JPG	03/03/2004 09:55:02 a.	-rw-r--r--
television1.jpg	11,040	Archivo JPG	03/03/2004 03:21:00 p.	-rw-r--r--
television1_enc.jpg	11,822	Archivo JPG	03/03/2004 09:57:47 a.	-rw-r--r--
varios1.jpg	10,239	Archivo JPG	03/03/2004 10:00:00 a.	-rw-r--r--
varios1_enc.jpg	10,658	Archivo JPG	03/03/2004 10:00:21 a.	-rw-r--r--

arta-tomcat-3.2.2/webapps/difusion/cultural/images SSH2 - 3des-cbc - hmac-md5 - none 25 Items (275.2 KE

Imagen 2 Contenido del directorio /wepapps/difusion/cultural/images

JSP

Aquí se almacenan los archivos JSP de la aplicación.



Name	Size	Type	Modified	Attributes
PROBAR		Folder	15/09/2004 02:59:00 p.	drwxr-xr-x
AbreDir.class	1,618	Archivo CLASS	11/03/2004 06:46:25 p.	-rw-rw-r--
AbreDir.java	1,054	Java(tm) File (N...	11/03/2004 05:36:26 p.	-rw-r--r--
accesoSalaDePrensa.jsp	811	Archivo JSP	15/09/2004 12:48:12 p.	-rw-r--r--
agendaDragon_1.jsp	1,670	Archivo JSP	12/03/2004 12:09:56 p.	-rw-r--r--
agendaDragon_2.jsp	1,196	Archivo JSP	23/04/2004 06:37:40 p.	-rw-r--r--
altaMedCom.jsp	2,857	Archivo JSP	28/04/2004 03:52:52 p.	-rw-r--r--
altaPromociones.jsp	4,140	Archivo JSP	10/09/2004 04:19:50 p.	-rw-r--r--
altaUsuarioss17.jsp	2,918	Archivo JSP	19/10/2004 08:31:49 a.	-rw-r--r--
altaUsuarioss18.jsp	3,060	Archivo JSP	23/03/2004 03:33:38 p.	-rw-r--r--
altaUsuarioss19.jsp	3,070	Archivo JSP	02/04/2004 04:36:26 p.	-rw-r--r--
altaUsuarioss20.jsp	3,060	Archivo JSP	23/03/2004 04:20:19 p.	-rw-r--r--
azul.htm	217	HTML Document	27/09/2004 05:11:17 p.	-rw-r--r--
azul.jsp	234	Archivo JSP	27/09/2004 05:12:13 p.	-rw-r--r--
boletines_5.jsp	4,849	Archivo JSP	09/03/2004 03:43:43 p.	-rw-r--r--
boletinfoto.jsp	355	Archivo JSP	08/05/2004 09:34:15 a.	-rw-r--r--
borraArchivos.jsp	1,879	Archivo JSP	27/09/2004 12:41:28 p.	-rw-r--r--
borraBoletines.jsp	1,355	Archivo JSP	23/09/2004 12:10:36 p.	-rw-r--r--
borraBoletinesMes.jsp	1,465	Archivo JSP	27/09/2004 12:36:52 p.	-rw-r--r--
BorraBoletinesNumB.jsp	1,748	Archivo JSP	14/09/2004 05:18:36 p.	-rw-r--r--
borraFotos.jsp	1,261	Archivo JSP	29/04/2004 06:03:47 a.	-rw-r--r--
busBolSalaDePrensa.jsp	3,316	Archivo JSP	21/05/2004 08:41:30 a.	-rw-r--r--
busca_cat_102.jsp	8,671	Archivo JSP	24/04/2004 11:48:56 a.	-rw-r--r--
busca_cat_a.jsp	3,381	Archivo JSP	18/03/2004 01:40:58 p.	-rw-r--r--
busca_lug_4.jsp	8,111	Archivo JSP	22/03/2004 01:38:50 p.	-rw-r--r--
buscador.html	1,650	HTML Document	10/03/2004 05:42:56 p.	-rw-r--r--
buscador1.html	1,664	HTML Document	17/03/2004 06:16:49 p.	-rw-r--r--
buscador_15.jsp	4,360	Archivo JSP	10/09/2004 01:27:47 p.	-rw-r--r--
BuscaPalabra.class	1,721	Archivo CLASS	11/03/2004 06:45:27 p.	-rw-rw-r--
BuscaPalabra.java	1,857	Java(tm) File (N...	11/03/2004 05:36:30 p.	-rw-r--r--
busSalaPrensa.jsp	2,079	Archivo JSP	21/05/2004 07:44:00 a.	-rw-r--r--
cal.jsp	516	Archivo JSP	25/02/2004 11:46:15 a.	-rw-r--r--
cal1.jsp	544	Archivo JSP	25/02/2004 11:48:26 a.	-rw-r--r--
cal2.jsp	516	Archivo JSP	25/02/2004 11:51:48 a.	-rw-r--r--
cal3.jsp	549	Archivo JSP	25/02/2004 11:57:22 a.	-rw-r--r--
cal5.jsp	568	Archivo JSP	01/03/2004 03:27:20 p.	-rw-r--r--

Imagen 3 Contenido del directorio /wepapps/difusion/cultural/JSP

SalaDePrensa

En esta carpeta se guardan archivos JSP, HTML y de formato de imagen para el modulo de Sala de Prensa de la aplicación.

Name	Size	Type	Modified	Attributes
FILMOTECA		Folder	06/05/2004 10:00:56 a.	drwxr-xr-x
FOMENTOEDIT		Folder	06/05/2004 10:01:56 a.	drwxr-xr-x
LAGO		Folder	06/05/2004 10:01:39 a.	drwxr-xr-x
LITERATURA		Folder	21/05/2004 12:00:59 p.	drwxr-xr-x
MUCA		Folder	27/01/2005 11:59:06 a.	drwxr-xr-x
MUSICA		Folder	27/01/2005 12:01:12 p.	drwxr-xr-x
RADIO		Folder	24/05/2004 03:34:00 p.	drwxr-xr-x
SANILDEFONSO		Folder	06/05/2004 10:02:24 a.	drwxr-xr-x
TEATRO		Folder	06/09/2004 01:34:57 p.	drwxr-xr-x
TORRI		Folder	06/05/2004 10:02:15 a.	drwxr-xr-x
TV		Folder	06/05/2004 10:01:48 a.	drwxr-xr-x
-13-02-04.doc	2,794	Documento de ...	19/10/2004 11:40:14 a.	-rw-rw-r--
1478	0	Archivo	14/09/2004 11:42:43 a.	-rw-rw-r--
AbreDir.java	1,254	Java(tm) File (N...	27/02/2004 01:22:56 p.	-rw-r--r--
boletines.jsp	4,639	Archivo JSP	09/03/2004 01:51:32 p.	-rw-r--r--
boletines_1.jsp	4,663	Archivo JSP	09/03/2004 01:58:09 p.	-rw-r--r--
boletines_10.jsp	5,587	Archivo JSP	22/09/2004 12:21:35 p.	-rw-r--r--
boletines_2.jsp	4,869	Archivo JSP	09/03/2004 02:07:09 p.	-rw-r--r--
boletines_3.jsp	4,869	Archivo JSP	09/03/2004 02:08:40 p.	-rw-r--r--
boletines_4.jsp	5,013	Archivo JSP	09/03/2004 03:40:54 p.	-rw-r--r--
boletines_5.jsp	4,849	Archivo JSP	09/03/2004 03:54:22 p.	-rw-r--r--
boletines_6.jsp	4,849	Archivo JSP	09/03/2004 03:55:01 p.	-rw-r--r--
boletines_7.jsp	4,997	Archivo JSP	17/03/2004 10:22:51 a.	-rw-r--r--
boletines_7.jspRES	4,996	Archivo JSPRES	09/03/2004 04:06:29 p.	-rw-r--r--
boletines_8.jsp	4,997	Archivo JSP	17/03/2004 10:40:39 a.	-rw-r--r--
boletines_9.jsp	4,995	Archivo JSP	17/03/2004 10:45:32 a.	-rw-r--r--
boletinesP.jsp	4,639	Archivo JSP	09/03/2004 01:53:50 p.	-rw-r--r--
borraBoletines.jsp	2,077	Archivo JSP	08/05/2004 08:52:16 a.	-rw-r--r--
BorraBoletinesNumB.jsp	1,595	Archivo JSP	14/09/2004 04:58:11 p.	-rw-r--r--
busBolSalaDePrensa.jsp	3,289	Archivo JSP	06/05/2004 03:14:55 p.	-rw-r--r--
buscador_15.jsp	4,360	Archivo JSP	10/09/2004 01:27:47 p.	-rw-r--r--
BuscaPalabra.java	1,873	Java(tm) File (N...	11/03/2004 03:07:35 p.	-rw-r--r--
busSalaPrensa.jsp	2,029	Archivo JSP	06/05/2004 02:41:05 p.	-rw-r--r--
cargaBoletines.jsp	1,036	Archivo JSP	31/08/2004 05:05:16 p.	-rw-r--r--
cargaBoletines1.jsp	1,024	Archivo JSP	17/03/2004 11:20:10 a.	-rw-r--r--
fotoBol.jsp	1,970	Archivo JSP	21/09/2004 05:54:27 p.	-rw-r--r--

Imagen 4 Contenido del directorio /wepapps/difusion/cultural/SalaDePrensa

SWF

Dentro de esta carpeta, se encuentran los archivos SWF de las animaciones que forman parte del diseño de este sistema

Name	Size	Type	Modified	Attributes
ccu.swf	52,301	Shockwave Flas...	09/03/2004 06:09:59 p.	-rw-r--r--
chavez.swf	19,623	Shockwave Flas...	09/03/2004 06:10:01 p.	-rw-r--r--
coiva.swf	15,046	Shockwave Flas...	09/03/2004 06:10:03 p.	-rw-r--r--
espacio.swf	23,868	Shockwave Flas...	09/03/2004 06:10:06 p.	-rw-r--r--
ines.swf	22,531	Shockwave Flas...	09/03/2004 06:10:08 p.	-rw-r--r--
jose.swf	20,286	Shockwave Flas...	09/03/2004 06:11:08 p.	-rw-r--r--
julio.swf	60,032	Shockwave Flas...	09/03/2004 06:11:09 p.	-rw-r--r--
libreria.swf	239,070	Shockwave Flas...	09/03/2004 06:11:13 p.	-rw-r--r--
librerloc.swf	30,077	Shockwave Flas...	09/03/2004 06:11:14 p.	-rw-r--r--
MEZA.swf	15,775	Shockwave Flas...	09/03/2004 06:09:57 p.	-rw-r--r--
paseo.swf	24,786	Shockwave Flas...	09/03/2004 06:11:15 p.	-rw-r--r--
Principal.swf	127,512	Shockwave Flas...	19/10/2004 12:05:24 p.	-rw-r--r--
principal_1.swf	104,080	Shockwave Flas...	22/03/2004 03:53:53 p.	-rw-r--r--
ruiz.swf	10,032	Shockwave Flas...	09/03/2004 06:11:17 p.	-rw-r--r--
serpiente.swf	24,285	Shockwave Flas...	09/03/2004 06:11:18 p.	-rw-r--r--

Imagen 5 Contenido del directorio /wepapps/difusion/cultural/SWF

Archivos en path /webapps/difusion/cultural/

En la imagen que se muestra a continuación se muestra la carpeta /wepapps/difusión/WEBINF/classes en esta carpeta deberan colocarse todos los archivos fuente y clases del sistema, para el actual diseño de la aplicación las clases y los fuentes se encuentras dentro de la carpeta classes/cal

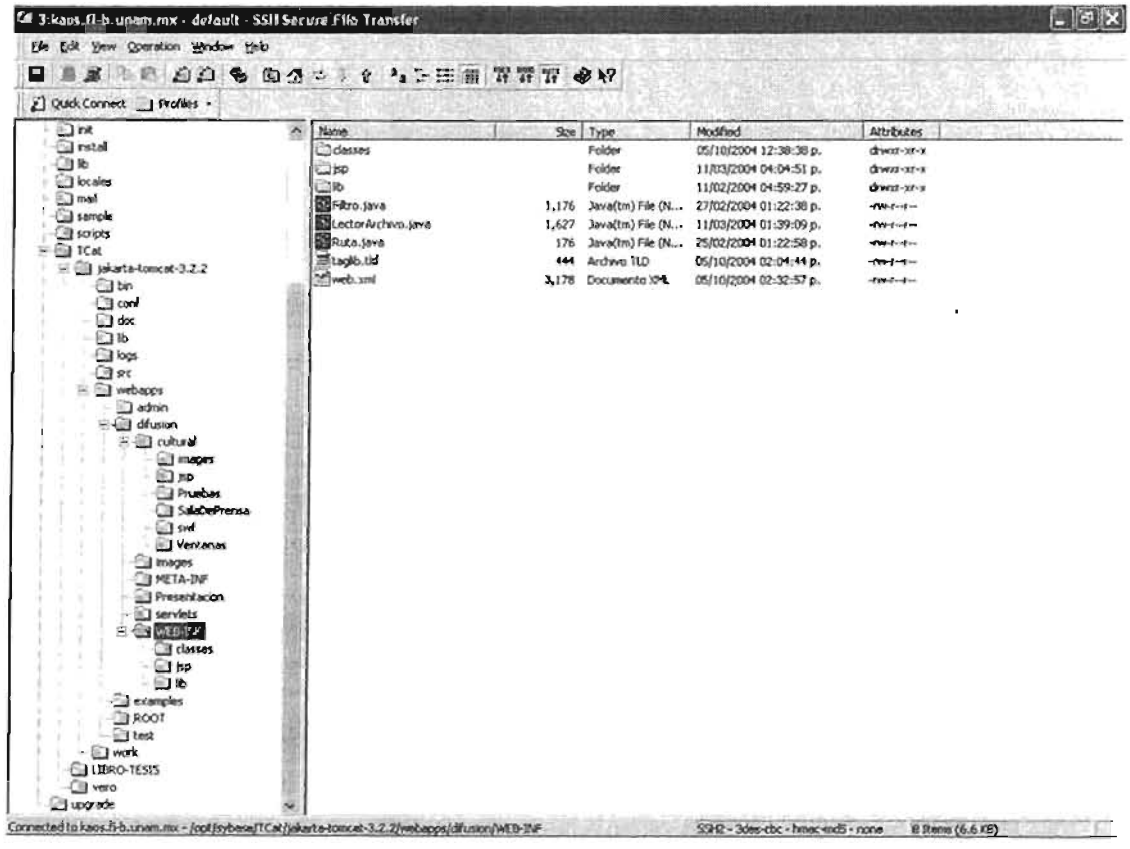


Imagen 6. Contenido del directorio /wepapps/WEB-INF

Glosario de términos

APACHE: Servidor web de libre distribución creado bajo una licencia Open Source.

ANSI: American National Standards Institute - Instituto Americano de Estándares Nacionales.

API: Application Program Interface -La interfaz de programas de aplicación describe a detalle los métodos de un sistema operativo o aplicación que pueden ser usados por un programador y realizar llamadas a ellos.

ASP: Aplication Service Provider - Proveedor de Servicios de Aplicación. Giro comercial donde una empresa ofrece una aplicación a un conjunto de clientes y su uso es a través de Internet.

B2B: Business to Businness - Empresa a Empresa. Tipo de aplicación utilizada entre empresas por medio de Internet.

B2C: Business to Consumer - Empresa a Consumidor. Tipo de aplicación donde interactúan los clientes finales con la empresa a través de Internet.

CERN: Organisation Européene pour la Recherche Nucléaire - Organización Europea para la Investigación Nuclear.

CA: Certificate Authority - Autoridad certificadora, es una entidad en una red que emite y maneja credenciales de seguridad y una llave pública para cifrado de mensajes.

CGI: Common Gateway Interface - Interfaz Común de Puerta de Enlace. Estándar la para transmisión de información entre el servidor y otras aplicaciones independientes del servidor.

COOKIE: Pedazo de información enviada por el servidor HTTP al agente del usuario para almacenar de manera presistente cualquier información.

CORBA: Common Object Request Broker Architecture - Arquitectura de Corredor Común para la Solicitud de Objetos. Es una arquitectura y especificaicón para crear, distribuir y administrar objetos de programa distribuidos en una red.

CPAN: Comprehensive Perl Archive Network - Red Global de Archivos Perl.

CSS: Cascade Style Sheet - Hoja de Estilo en Cascada. Es un conjunto de especificaciones que definen o redefinen el compartamiento de una marca en un documento HTML.

DARPA: Defense Advanced Research Projects Agency - Agencia de la Defensa para la Investigación de Proyectos Avanzados (EUA).

DBMS: DataBase Management System - Sistema de Manejo de Bases de Datos, es un programa de computadora que permite a uno o mas usuarios crear y acceder datos en una base de datos.

DNS: Domain Name Server - Servidor de Nombres de Domino. Protocolo tanto para la resolución de nombre de dominio como su viceversa.

DOM: Document Object Model - Modelo de Documento Objeto.

EXTRANET: Red privada de una empresa o institución que utiliza los protocolos de Internet y un sistema de telecomunicaciones público para compartir, de manera segura, parte de la información de negocios con clientes, proveedores, socios, u otros negocios.

GPL: General Public Licence. Es la licencia que utiliza la Free Software Foundation para proteger el software distribuido por el proyecto GNU, así como otros programas que no forman parte del proyecto GNU, pero que sus desarrolladores han adoptado.

GUI: Graphic User Interface - Interfaz Gráfica del Usuario.

GNU: GNU es un acrónimo recursivo que significa "GNU No es Unix". Stallman sugiere que se pronuncie, en inglés, como "guh-noo" (se puede observar que el logo es un ñu) para evitar confusión con "new" (nuevo). En español, GNU se pronuncia fonéticamente.

HIPERTEXTO: Es la organización de unidades de información en asociaciones conectadas que un usuario puede escoger. Una instancia de esa asociación es llamada vínculo o hipervínculo.

http: HyperText Transfer Protocol - El Protocolo de Transferencia de Hipertexto es un conjunto de reglas para intercambiar archivos (texto, gráficos, imágenes, sonido, video y otros archivos multimedia) en el World Wide Web.

IETF: Internet Engineering Task Force - Es la entidad encargada de definir el protocolo de operación estándar de Internet, el TCP/IP.

INTERNET: Es una red de redes de alcance mundial que utiliza los protocolos TCP/IP, también se le conoce como ``La Red". Es cualquier red de redes que esta limitada a cierto número de redes o a ciertas localidades geográficas. Es una red privada que utiliza los protocolos de Internet pero reside en una empresa u organización.

ISO: International Standards Organization - Organización Internacional para la Estandarización.

ISO 8879: SGML (*Standard Generalized Markup Language*) es la norma ISO 8879:1986 para la definición de lenguajes que describen estructuras de documentos. Un lenguaje SGML especifica la estructura lógica del documento (las partes que lo componen, por ejemplo), pero no su aspecto en la pantalla o en el papel (márgenes o tipos de letra, por ejemplo).

KERBEROS: Es un método seguro para autenticar una solicitud a un servicio en una red de computadoras.

LDAP: Lightweight Directory Access Protocol - Protocolo ligero de acceso a directorios, es un protocolo que permite ubicar organizaciones, individuos y otros recursos tales como archivos y dispositivos en una red, ya sea Internet o una intranet corporativa.

LLAVE PRIVADA: Es la llave utilizada para descifrar los datos cifrados con la llave pública correspondiente.

LLAVE PÚBLICA : Es un valor proveído por una autoridad certificadora que puede usarse para cifrar mensajes.

MIME: Multipurpose Internet Mail Extensions - Extensiones Multipropósito para el Correo de Internet. Extensión al protocolo original del correo electrónico para la transformación de cadenas ASCII-8 a ASCII-7. Tipos MIME representa los tipos de archivos reconocidos por su navegador. La opción Tipos MIME en el panel de control le permite especificar que tipos de archivos puede abrir o manejar su navegador (text/html, image/gif, aplication/x-word).

OPEN SOURCE: Código abierto. Es una modalidad en el desarrollo de software en la cual los desarrolladores hacen disponible el código fuente del mismo a terceras personas.

ORB: Object Request Broker - Corredor de Peticiones de Objetos. Componente de software que se encarga de transportar mensajes entre objetos que utilizan el estándar CORBA.

PAM: Pluggable Authentication Modules - Módulos Enchufables de Autenticación.

PERL: Practical Extraction and Report Language - lenguaje práctico de extracción y reportes.

PL/SQL: Lenguaje de programación Simple de Oracle.

POSIX: Portable Operating System Interface - Interfaz portable de sistemas operativos es un conjunto de estándares de interfaces de sistemas operativos basados en el sistema operativo UNIX.

RFC: Request for Comments - Solicitud de Comentarios, es un documento formal de la IETF que es el resultado de la definición de un anteproyecto acerca de un tópico determinado y está sujeto a revisión por entidades interesadas.

RSA: Sistema de cifrado y autenticación para Internet que utiliza el algoritmo creado en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman.

SGML: Standard Generalized Markup Language - Lenguaje Estándar de Marcado Generalizado. Estándar para definir un lenguaje de marcado y su conjunto de marcas.

SQL: Structured Query Language - Lenguaje Estructurado de Consultas. Lenguaje estándar interactivo para la consulta y actualización de bases de datos.

SSL: Secure Socket Layer - Capa de Sockets Seguros. Protocolo posterior al TCP/IP que cifra la infromación transmitida.

SQL: Structured Query Language - El lenguaje estructurado de consulta es un lenguaje de programación que permit e obtener y actuallzar una base de datos.

TCP/IP: Transmission Control Protocol / Internet Protocol - Protocolo para el Control de Transmisiones / Protocolo de Internet. Identifica a un conjunto de protocolos que operan en Internet.

UNIX: Sistema operativo desarrollado en los Laboratorios Bell en 1969 como un sistema interactivo de tiempo compartido.

URI: Uniform Resource Identifier - Identificador Uniforme de Recursos. Cadena que especifica al recurso que se desea solicitar.

VINCULO: Es una conexión entre una palabra, imagen u objeto de información a otro documento.

WORLD WIDE WEB: La telaraña de cobertura mundial, también conocido como WWW o Web, es el conjunto de recursos y usuarios en la Internet que hacen uso del protocolo de transferencia de hipertéxto (HTTP). Según la W3C: ``El World Wide Web es el universo de información accesible por red, es una representación del conocimiento humano".

W3C: Es un consorcio industrial que busca promover estándares para la evolución del Web y la interoperabilidad entre productos para el WWW al producir especificaciones y software de referencia.

XML: Extensible Markup Language - Lenguaje de Marcado Extensible. Es lenguaje para definir el formato de la información a compartir a través de una red.

- ## Bibliografía

Libros

Servlets and java server pages 2/ed/vol1
Autor Brown Larry
Editorial Pearson

Diseño de sitios web manual de referencia
Autor Powel Tomas
Editorial Mc Graw Hill

Java 2 Manual de Referencia
Autor Schildt Herbert
Editorial Mc Graw Hill

JSP manual de referencia
editorial Mc Graw Hill

Macromedia Flash MX
Autor:Teresa Velasco, Sixto Cantilla
Editorial: Prentice Hall

Java Guia de Desarrollo
Autor:Jaime Jawoeski
Editorial: Prentice Hall

Object-Oriented Analysis and Desing whit UML Student Manual
Version 2003.06.00
Volumen 1,2,3.

Internet

http://www.monografias.com/trabajos/objetos/objetos.shtml
http://www.htmlweb.net/manual/formularios/formularios_9.html
http://www.x-extrainternet.com/glosary.html
http://www-306.ibm.com/software/rational/
http://www.programacion.com/java/tutorial/jar/
http://www.geocities.com/SiliconValley/Bay/8259/parte1.html
http://javabasico.osmosislatina.com/curso/objetos.jsp#car
http://www.monografias.com/trabajos6/resof/resof.shtml
http://www.monografias.com/trabajos11/basda/basda.shtml
http://www.desarrolloweb.com/manuales/9/