42
29.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"ARAGON"

INGENIERIA EN COMPUTACION

ARQUITECTURA CLIENTE/SERVIDOR EN SYBASE A TRAVES DEL WEB

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

RUBEN EFREN MONTIEL LEONG

ASESOR DE TESIS: ING. JIMENEZ VAZQUEZ DONACIANO.

OCTUBRE, 1997

UNIVERSIDAD NACIONAL AVPNMA DE MEXICO

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
CAMPUS ARAGON
CARRERA DE INGENIERIA EN COMPUTACION

ING. DONACIANO JIMENEZ VAZQUEZ
ING. MANUEL MARTINEZ ORTIZ
ING. SILVIA VEGA MUYTOY
ING. JUAN GASTALDI PEREZ
ING. ERNESTO PEÑALOZA ROMERO

Informamos a ustedes de la autorización que se le concede al alumno ***RUBEN EFREN MONTIEL LEONG*** para que pueda desarrollar el trabajo de tesis titulado **"*ARQUITECTURA CLIENTE - SERVIDOR EN SITBASE A TRAVES DEL WEB*"**, dirigida por el *Ing. Donaciano Jiménez Vázquez*, solicitando a ustedes sean tan amables de revisar el avance del mismo y hacer las observaciones que consideren pertinentes, o en su caso, indicar al alumno si dicha revisión se hará a la conclusión del trabajo de tesis.

Sin otro particular, aprovecho la ocasión para enviarles un cordial saludo.

A T E N T A M E N T E
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón Edo. de Méx., Octubre 14 de 1996
LA JEFA DE CARRERA

ING. SILVIA VEGA MUYTOY

INGENIERIA EN COMPUTACION
U. N. A. M.

SVM/rgg.

Doy gracias a todas las personas que hicieron posible este trabajo, ya que sin su ayuda no hubiera sido posible. Pero quiero dar un agradecimiento especial y afectuoso a:

A mi mamá Martha, a mi abuela Paulina y mi papá Efrén por todo lo que han hecho por mí.

A mis hermanos Martha, Elizabeth, Francisco, Blanca, Virginia y Gregory por su comprensión y confianza.

A mis cuñados y sobrinos por su apoyo

A mis tíos y primos por sus consejos.

A Adriana por su confianza, comprensión y por estar a mi lado en todo momento.

A mis amigos por su apoyo y consejos.

A mis maestros por todo lo que me enseñaron.

Al ingeniero Donaciano por su ayuda en la elaboración de este trabajo.

Tesis:

Arquitectura Cliente/servidor con Sybase a través del Web.

Arquitectura Cliente/Servidor con Sybase a través del Web

Introducción

INTRODUCCION

Debido a las ventajas que ofrece la arquitectura cliente/servidor, el auge que tienen los servicios del Web en Internet, el manejo de información y seguridad que ofrece el manejador de Base de Datos Sybase, así como la programación de CGI's. El presente trabajo tiene como objetivo ser una guía para implementar servicios cliente/servidor para trabajar con Sybase a través del Web y así poder mostrar la información que se tiene a "todo el mundo".

Aunque se tienen productos comerciales que realizan algo similar, este trabajo muestra que es la solución ya que el costo de su implantación es bajo, y las ventajas que ofrecen los lenguajes como C, Perl y HTML, son aprovechados para que la programación sea fácil y de manera rápida.

El ejemplo que se da es una forma muy sencilla de llevar un control de datos de alumnos inscritos en las carreras impartidas en la ENEP Aragón.

Para llegar a este fin este trabajo se conforma de siete capítulos:

Capítulo I	Arquitectura cliente/servidor.
Capítulo II	Servicios de Internet para el Web.
Capítulo III	Conectividad de Sybase con servicios del Web.
Capítulo IV	Configuración del Servidor del Web.
Capítulo V	Seguridad y control de accesos para el Web.
Capítulo VI	Gateways y formas.
Capítulo VII	Implantación de un servicio.

El capítulo I tiene como objetivo el mostrar las ventajas que tiene el trabajar con la arquitectura cliente/servidor, así como establecer los conceptos básicos para comprender la terminología utilizada en el presente trabajo.

El capítulo II muestra y nos permite comprender que es el Web, como trabaja el World Wide Web, sus ventajas, así como también conocer algunos conceptos relacionados a este servicio.

En el capítulo III se habla de la importancia que tiene los manejadores de bases de datos para organizar y mantener presente la información de una manera segura y rápida. Así como las características de Sybase, y por último de como se puede poner la información que maneja en el Web.

En el capítulo IV se explica como configurar e instalar un servidor del Web, en caso específico del servidor del httpd de la NCSA. Para que este trabaje con la configuración básica. Además se explica que archivos están involucrados y para que sirve cada uno, y se explica las principales directivas de estos archivos.

El capítulo V trata de la seguridad que conlleva el tener un servicio a disposición del público. Se aprovecha y se utiliza la seguridad que brinda el servidor del Web por sí mismo.

En el capítulo VI se habla de los CGI's y las formas, se da la definición de CGI; utilidad de estos, los lenguajes involucrados en este trabajo y características de estos desde el punto de vista de utilización en CGI's.

En el capítulo VII se presenta el desarrollo de un servicio y la interacción de todos los componentes antes mencionados. Este servicio trata de mostrar como introducir datos a Sybase y como poder realizar consultas a estos datos.

Capítulo I

Arquitectura Cliente/Servidor

1.1 Definición de la arquitectura cliente/servidor.

El termino *Cliente/servidor* se refiere a la relación entre 2 sistemas o procesos. El cliente[1] es un sistema que hace peticiones de trabajo para que sea realizado por un sistema servidor[2]. En algunas situaciones, quien es el cliente y cual el servidor es determinado por la relación de peticiones para el servidor.

Hay que tomar en cuenta que un servidor puede en un momento ser cliente cuando envía una petición hacia otro servidor. Por eso es conveniente referirnos a un servidor como un sistema o proceso que recibe peticiones y a un cliente como un sistema o proceso que envía peticiones.

La figura 1.1 ilustra la simple relación en el cliente/servidor. El cliente envía una petición de servicio hacia el servidor, y este envía la respuesta apropiada.

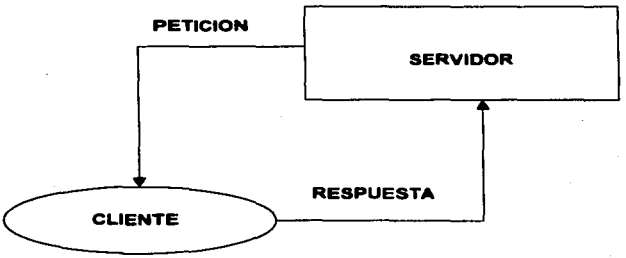

FIGURA 1.1 Relación Cliente/servidor.

[1] Al cliente se le conoce también como front-end.

[2] Al servidor se le conoce como back-end.

Con la arquitectura cliente/servidor se pueden integrar diferentes tipos de equipos y sistemas operativos en un ambiente único de procesamiento. Bajo este modelo, las operaciones de procesamiento se jerarquizan y la red determina las necesidades del usuario y solo transmite la información útil. La implantación de está arquitectura promueve el uso de sistemas abiertos dado que tanto clientes como servidores corren en diferentes plataformas de hardware y software.

1.2 Introducción a la tecnología cliente/servidor.

La tecnología *cliente/servidor* es la más grande tendencia en el desarrollo de sistemas de Bases de Datos y tecnología de redes (Los servicios de Internet se basan en está tecnología). Hoy en día el compartir datos y la manera de accesar a la información de manera fácil, rápida y seguro; es lo que se trata de obtener en los sistemas computacionales.

La comunicación entre los servidores y clientes se realizará a través de algún tipo de red. *cliente/servidor* aparece, básicamente, como respuesta a dos situaciones paralelas y complementarias:

La primera se produce en la década de los '80, cuando la tendencia en la configuración computacional de las empresas fue adquirir computadoras centrales con terminales conectados en línea (por ejemplo: HP-1000 y terminales VT100). La generación de nuevas necesidades y nuevos usuarios, que, al correr del tiempo, provocaban que la computadora se viera superada en sus capacidades. Como solución se reemplazaba la computadora central por uno más grande, el cual nuevamente se veía superado. Además, estos equipos no cuentan con una interfaz amistosa con el usuario final.

La segunda situación se produce por el rápido auge de las computadoras personales, que a bajo costo y con cada vez mayores capacidades (rapidez, memoria, bajo costo, etc.) inundaron las empresas. La multiplicidad de herramientas disponibles en este tipo de computadoras que los usuarios utilizan con gran comodidad, además de sus capacidades

gráficas, hacen deseable que el acceso a la información pueda ser realizada en forma eficiente desde estos equipos.

Esta tecnología produce un gran número de ventaja sobre otras tecnologías (como lo son la tecnología centralizada y la maestro/esclavo). La tecnología cliente/servidor optimiza el uso de recursos de hardware y software tanto en el servidor como en el cliente.

Con la arquitectura cliente/servidor, uno puede combinar la característica de usar interfaces "amigables" y la flexibilidad de correr aplicaciones sobre una PC, con accesos a tecnologías de Bases de Datos relaciónales provistas por servidores de Bases de Datos. También el desarrollo de la estandarización para accesar hacia el servidor de bases de datos SQL, hace posible el mejor desarrollo para aplicaciones portables y el acceso de datos desde las fuentes más diversas o distribuidas.

La configuración cliente/servidor intenta hacer un mejor uso de los recursos de hardware y software separando las funciones entre las 2 partes: el cliente, en donde corren las aplicaciones sobre las computadoras clientes o estaciones de trabajo; y el servidor que almacena y maneja las peticiones de datos.

En nuestro caso Sybase es nuestro servidor de base de datos y es quien ejecuta los comandos SQL mandados por el cliente, tiene los datos disponibles para cualquier petición y proporciona la seguridad e integridad de los datos. Otro servidor que tenemos es el servidor de *http* que quien nos da el acceso a la Internet a través del Web.

Cliente/servidor procura rescatar lo mejor de estos ambientes. La integridad y seguridad de los datos que proveen en general los mainframes, junto a la facilidad de uso y economía de las computadoras personales.

Los primeros trabajos conocidos para la arquitectura *cliente/servidor* los realizó Sybase, que se fundo en 1984 pensando en lanzar al mercado únicamente productos para esta arquitectura.

1.3 Ventajas y desventajas de la arquitectura cliente/servidor.

Algunas de las ventajas que se tienen con está arquitectura son los siguientes:

1. El trabajo y las funciones están distribuidas.
2. Disminuye la carga de la red.
3. La arquitectura cliente/servidor es independiente del Sistema Operativo y de la plataforma en que se está corriendo, tanto el cliente como el servidor.
4. Permite el procesamiento distribuido.

Sus desventajas son:

1. Aumenta el trabajo de administración.
2. Aumenta el trabajo de mantenimiento.
3. Aumenta el soporte y programación de los clientes.

El servidor se encarga de procesar la petición de datos y envía la respuesta hacia la aplicación del usuario, interactuar con el usuario, generar peticiones de datos, proveer el mantenimiento de la integridad de los datos, recuperación de errores, control de seguridad, así como perfecciona a los usuarios la concurrencia de accesos y la actualización de datos. El cliente en cambio toma ventaja de su interfaz gráfica disponible y se encarga de la presentación.

Aunque la aplicación cliente y el servidor pueden correr en la misma máquina, puede ser más eficiente y efectivo cuando ambos se localizan en diferentes máquinas conectadas a través de la red.

Los beneficios que aporta la arquitectura cliente/servidor se citan a continuación:

Las aplicaciones clientes no se responsabilizan de realizar ningún tipo de proceso de información. Estas se limitan a tareas como petición de entrada de datos al usuario, presentar y analizar la información utilizando las capacidades de desplegar está en la workstation o en la terminal.

Las aplicaciones clientes se diseñan de forma independiente a la localización física de los datos. Así, si la información se mueve o distribuye a otros servidores de la Base de Datos, la aplicación continua funcionando con pocas o ninguna modificación. Este es la ventaja que nos ofrece en nuestro caso Sybase.

1.4 Variantes de la arquitectura cliente/servidor.

Como se ve, los problemas potenciales de una arquitectura cliente/servidor no son triviales, y exige un estudio profundo de cada situación. Las implicaciones no están solo relacionadas con la 'performance', sino que existen otras consideraciones relativas a la seguridad, gestión de resguardo y recuperación de datos, actualización de versiones de programas, etc. Solo tocaremos la problemática desde el punto de vista de la planificación para el rendimiento.

El desarrollo de aplicaciones *cliente/servidor* es inevitable por un conjunto de razones:

1. En muchas situaciones es más eficiente que el procesamiento centralizado, dado que este experimenta una "deseconomía" de escala cuando aumenta mucho la cantidad de usuarios.

2. Existen servidores razonablemente eficientes y confiables, como es el caso de Sybase.

3. Se han establecido un estándar de hecho para una interface Cliente/Servidor: el ODBC[1] SQL, adoptado por todos los fabricantes importantes de servidores.

Es imprescindible, para apoyar con información a la creciente cantidad de ejecutivos de nivel medio que necesitan tomar decisiones ante la computadora, ayudándose con las herramientas "front office" que utilizan con toda naturalidad (planillas electrónicas, procesadores de texto, graficadores, correos electrónicos, etc.).

Existía un conjunto de lenguajes "front end" como, por ejemplo, Delphi, Foxpro, Powerbuilder, SQL Windows, Visual Basic, lenguaje C, Perl, etc.

Las ventajas que nos dan las formas en Internet y el Web en particular que nos permite conectarnos a su servicio. Realizar formas y la presentación gráfica que nos ofrece. Existe un conjunto de variantes de la Arquitectura Cliente / Servidor, dependiendo de donde se ejecutan los diferentes elementos involucrados: [a] administración de los datos, [b] lógica de la aplicación, [c] lógica de la presentación.

1.4.1 Presentación Distribuida.

La primera variante que tiene algún interés es la llamada Presentación Distribuida, donde tanto la administración de los datos, como la lógica de la aplicación, funcionan en el servidor y la lógica de la presentación se divide entre el servidor (parte preponderante) y el cliente (donde simplemente se muestra).

Por lo que se obtiene una mejor presentación y ciertas capacidades mínimas para vincular las transacciones clásicas con el entorno Windows.

1.4.2 Administración de Datos Remoto.

Una segunda alternativa plausible es la Administración de Datos Remota, donde dicha administración de los datos se hace en el servidor, mientras que tanto la lógica de la aplicación, como la de la presentación, funcionan en el Cliente.

Desde el punto de vista de las necesidades de potencia de procesamiento, esta variante es la optima: se minimiza el costo del procesamiento en el Servidor (solo se dedica a administrar la base de datos, no participando en la lógica de la aplicación que, cada vez, consume mas recursos), mientras que se aumenta en el cliente, donde es irrelevante, teniendo en cuenta las potencias de Cliente necesarias, de todas maneras, para soportar el sistema operativo Windows.

El otro elemento a tener en cuenta es él transito de datos en la red. Esta variante podrá ser optima, buena, mediocre o pésima de acuerdo a este transito.

En el caso de transacciones o consultas activas, donde prácticamente todos los registros seleccionados son necesarios para configurar las pantallas a mostrar, este esquema es optimo.

Una variante interesante es la de complementar el procesamiento en el cliente con procesamiento en el servidor.

Se tiene total libertad para escoger donde se coloca la lógica de la aplicación: en el cliente, en el servidor de base de datos, o en otro(s) servidor(es). También se tiene total libertad para la elección del lenguaje a utilizar. En nuestro caso utilizaremos dos lenguajes como lo son el lenguaje Perl y C.

Los programas serán óptimos desde el punto de vista del performance.

No existe compromiso alguno con el uso de lenguajes propietarios, por lo que las aplicaciones serán totalmente portables sin cambio alguno.

Puede determinarse en que servidor(es) se quieren hacer funcionar estos procedimientos. En aplicaciones criticas se pueden agregar tantos servidores de aplicación como sean necesarios, de forma simple, y sin comprometer en absoluto la integridad de la base de datos, obteniéndose una escalabilidad muy grande sin necesidad de tocar el servidor de dicha base de datos.

1.5 Internet y la arquitectura cliente/servidor.

Un fenómeno que no podemos dejar de considerar es el crecimiento permanente de Internet. Actualmente se utiliza para un conjunto de propósitos (correo electrónico, transferencia de archivos, WWW). La disponibilidad de los WWW ha modificado mucho las cosas y los cambios mayores aun están por producirse.

Hoy en día gracias a los CGI's se pueden aprovechar a la Internet para formalizar negocios dentro de las empresas, y que estén interactuando con la base de datos.

Este tipo de facilidad da al uso WWW una mucho mayor proyección y posibilitara, en gran escala, ventas al detalle, ventas de información y mucho más servicios.

La figura 1.2 muestra la arquitectura Cliente/Servidor a través del Web:

El "cliente" trabaja en una computadora local; puede pasar archivos a otros programas para mostrarlos.

El "Servidor" trabaja en una computadora remota, este provee la seguridad y la autorización para obtener los archivos.

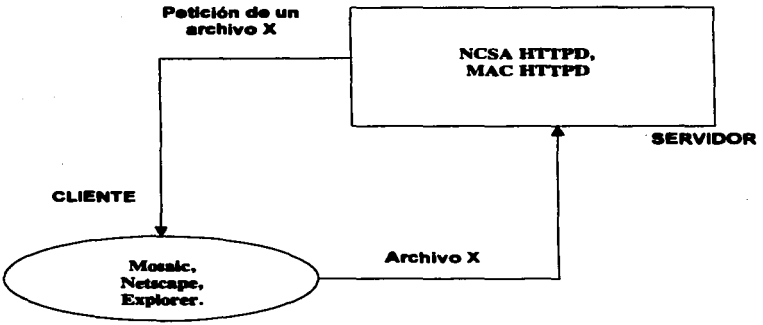

FIGURA 1.2 Relación Cliente / servidor.

El usuario final accede al WWW por medio de un programa "cliente" llamado "navegador" (Browser)[2] o navegador desde una PC, Macintosh o estación Unix. Este navegador lee los documentos que le solicita al servidor. Para hacer este pedido el programa le mostrara los "lazos" (links). Una vez que el pedido llega al servidor, este obtiene dicho archivo y lo envía hacia el cliente.

1.5.1 Terminología Básica.

Navegador o Visualizador: (Browser) Programa interactivo usado para acceder a la información del WWW.

Cliente (Client): Un "visualizador" es considerado como un Cliente WWW, es un programa que trabaja en la maquina local en diversas plataformas (Unix, Mac, PC, etc.). El cliente envía las peticiones al programa servidor.

Servidor (Server): Programa que interpreta la solicitud del Cliente, y le envía la información, este termino también indica el sistema en que el programa servidor esta trabajando.

La arquitectura cliente/servidor ha sido impulsada y complementada por desarrollos como:

-Sistemas Operativos como NT (ya que hace más transparente la migración hacia está arquitectura).

1. -Internet porque es la mejor forma de accesar a la información depositada en servidores ubicados en todo el mundo.

-Java, por ser el nuevo lenguaje de programación para Internet que hará posible desarrollar aplicaciones independientes del Sistema Operativo en el que se esté trabajando.

---

[1] ODBC Interfaz propietaria que proporciona acceso a la base de datos desde el exterior. ODBC significa Open Database Conectivity (Conectividad Abierta para Bases de Datos).

[2] Existen personas que les llaman navegadores, y cualquiera de los términos son correctos, en nuestro caso utilizaremos el término navegador, de ahora en adelante.

Capítulo II

Servicios de Internet para el Web

2.1 Introducción al funcionamiento de WWW.

El WWW (World Wide Web) o simplemente Web, ha despertado grandes expectativas en los usuarios de Internet y el público en general. Esto ha creado una demanda de los niveles de dirección en el personal de sistemas para tener una "página en el Web" lo antes posible y una de las formas de hacer esto posible es tener su propio servidor del Web.

WWW utiliza como protocolo de comunicación el HyperText Transfer Protocol (HTTP). A través del Web se pueden ver imágenes (en formato GIF y JPG), películas (video-clips) en formato MPEG, AVI y QUICKTIME, sonido en formato AU, y texto en formato de alta resolución (en formatos PostScript, DVI de TeX y Acrobat, además de HTML). HTTP facilita la distribución de documentos en hipertexto.

2.2 Definición de WWW.

El WWW es un sistema de información usado en Internet para la comunicación y comparición de datos operando bajo el modelo cliente/servidor. Los clientes Web o navegadores, como se hará mención por este nombre desde este momento, pueden accesar a la información por medio de diferentes protocolos e hipermedia usando un esquema de direccionamiento.

La figura 2.1 representa la organización técnica del Web basada en esta definición.

El WWW esta hecho en base al hipertexto. La información presentada en el Web no es necesario que este construida en forma lineal. En términos matemáticos, el Web esta direccionado gráficamente, en el cual los nodos (paginas Web de hipertexto) son conectados por vectores (ligas de hipertexto en el Web). Las áreas sobre las paginas Web, llamadas anclas (anchors), nos indica que podemos consultar otro documento con información relacionada con el tema (conocidos como "hotspots").

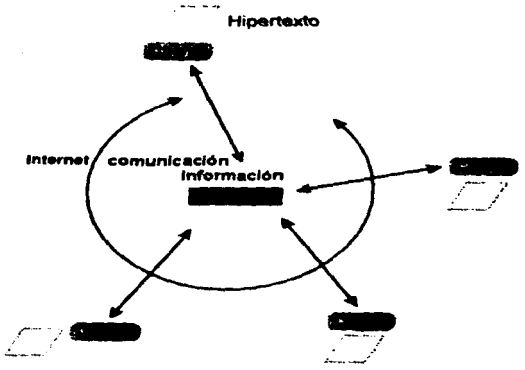

Figura 2.1 Organización del Web.

La figura 2.2 representa la organización básica del hipertexto. Las ligas entre las paginas, muestran como son dirigidas las flechas, se conectan a una ancla sobre una página de hipertexto hacia otra página o hacia una localización especifica sobre la pagina. Estas anclas son con frecuencia mostradas como texto iluminado o subrayado que el usuario puede seleccionar, con frecuencia usando un click en el punto de interface.

Este tipo de sistema la información se puede recorrer en una forma no lineal. El usuario puede seleccionar una liga sobre una pagina e iniciar la lectura en otra pagina; alternativamente, el usuario puede saltar o cambiar a una diferente liga sobre una subsecuente lectura de la misma información.

Otra característica del hipertexto es el concepto de navegar. De esta forma, la información sobre el Web, puede ligarse diferentes documentos escritos por otros autores.

Así es como el hipertexto ofrece la característica de no limitarse en su contenido con un simple trabajo escrito por un solo autor.

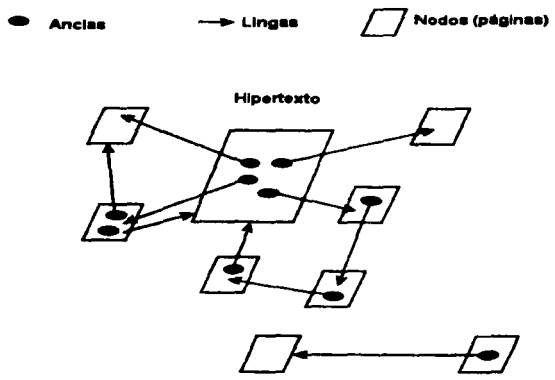

Figura 2.2 Organización del hipertexto

La figura 2.3 ilustra como el hipertexto incluyen ligas que pueden cruzar los limites del trabajo.

El hipertexto sobre el Web es escrito usando Lenguaje de Marcas de Hipertexto (HTML Hypertext Markup Languaje), una aplicación del Estándar Generalizado del Lenguaje de Marcas (SGML , Standard Generalized Markup Languaje)[1]. La filosofía detrás SGML es habilitar el formato de información para sistemas de publicación u otras aplicaciones que facilitan el compartir la información. HTML es definida por SGML y tiene la intención de ser un lenguaje semántico de señalización (marcas).

[1] SGML es un estándar internacional (ISO 8879) para el procesamiento de información de texto.

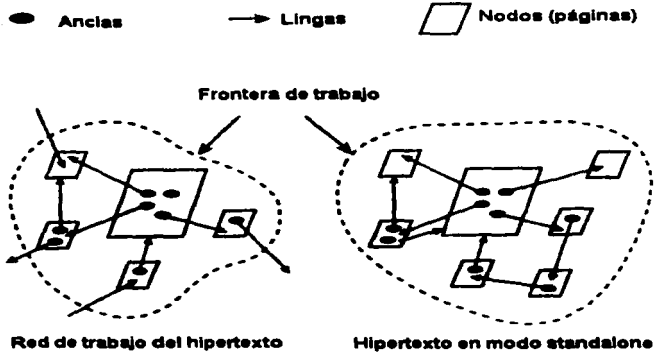

Figura 2.3 Las fronteras de trabajo del hipertexto.

El Web permite difundir y reunir información (a través de la capacidad de las formas del HTML). De este modo, el Web no es simplemente un sistema para difundir información, permite la comunicación interactiva. Usando formas con programas gateways, los desarrolladores en el Web pueden crear sistemas para la manipulación o cambios en una estructura de hipertexto.

La forma distribuida de las actividades de peticiones y atenciones del modelo cliente/servidor permite mucho rendimiento. Porque el software cliente interactúa con el servidor de acuerdo con el protocolo predefinido, el software cliente puede llevarse en la computadora anfitriona del usuario. (El servidor no tiene que preocuparse por las particularidades del software y del hardware del cliente). Por ejemplo, un navegador puede ser desarrollado para computadoras Macintosh y estas pueden accesar a cualquier servidor. De esta misma manera un servidor Web puede ser accesado por un navegador Web escrito

para una Workstation de Unix corriendo es sistema X Windows. Así aunque se encuentren en diferentes plataformas, se permite la comparición de información de una manera transparente y fácil.

Los navegadores son multiprotocolos de información, esto es porque ellos pueden accesar a una gran variedad de servidores que proveen información usando un conjunto de reglas para comunicaciones (protocolos). Los navegadores y las conexiones dentro de los documentos Web pueden hacer referencia a servidores usando alguno de los siguientes protocolos:

- HTTP (Protocolo de Transferencia de Hipertexto). Este es el protocolo nativo del Web, permite la transmisión de hipertexto sobre las redes.
- FTP (Protocolo de Transferencia de Archivos). Este protocolo permite a los usuarios transferir archivos de texto o binarios entre computadoras anfitrionas a través de la red.
-Gopher Este protocolo fue diseñado para compartir información mientras es presentada usando un sistema de menús, documentos, o conexiones de sesiones Telnet.
- News (Network News Transfer Protocol, NNTP). Este es un protocolo para la distribución de noticias Usenet. Usenet es un sistema asíncrono de texto de discusión sobre algunos tópicos subdivididos en grupos llamados newsgroups.
- Telnet. Este protocolo es usado para realizar sesiones remotas a computadoras anfitrionas.

De esta manera, un navegador funciona como un cliente Gopher cuando este accesa a un servidor Gopher, y como un cliente de News cuando este accesa a un servidor de noticias Usenet. La figura 2.4 muestra la variedad de relaciones cliente/servidor dentro de Internet. Aunque algunos clientes son especializados (por ejemplo, un cliente Gopher solo es usado para accesar solo a servidores Gopher), los clientes Web pueden accesar diferentes tipos de servidores.

Las direcciones que hacen referencia a un documento o recurso disponible en el Web (o en la Internet en general) son llamadas como Uniform Resource Locator, o URL. Un URL esta formado en una sintaxis particular que expresa en donde se encuentra un

recurso, incluye información como el nombre de la maquina anfitriona y la ruta en donde se encuentra el recurso, así también como otra información. Por ejemplo:

http://www.w3.org/hypertext/WWW/TheProject.html

Este URL hace referencia a un servidor Web (indicado por el *http* al inicio), el cual indica que sé esta utilizando "el protocolo de transferencia de hipertexto". Dentro del servidor Web llamado www.w3.org, existe un archivo llamado TheProject.html en el directorio hypertext/WWW/.

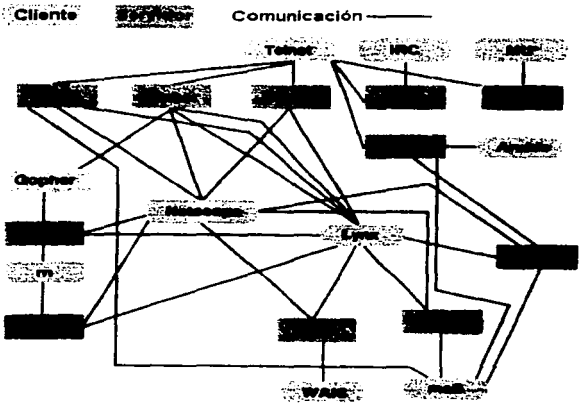

Figura 2.4 Relaciones Cliente/servidor dentro de Internet.

ftp://ftp.w3.org/pub/

Este URL hace referencia a un anfitrión (ftp.w3.org) en el cual se puede accesar usando "el protocolo de transferencia de archivos" (FTP). El URL hace referencia al directorio pub/

que se encuentra en el anfitrión, el cual mostrara una lista de archivos, directorios que se encuentran en el directorio **pub** o posiblemente se encuentre el directorio vacío.

news:comp.infosystems.www.misc

Este URL hace referencia a un grupo de noticias Usenet. Cuando el usuario selecciona este URL, el navegador muestra el actual grupo de títulos de artículos en el grupo de noticias Usenet **comp.infosystems.www.misc**, como un grupo dedicado para discutir de tópicos diversos (misc), acerca del World Wide Web (www), computadoras (comp) y sistemas de información (infosystesms).

A diferencia de los dos anteriores URL's, este no hace referencia a un anfitrión en particular, para esto se definió el servidor en el navegador cuando este fue instalado. Este servidor de noticias es generalmente definido dentro de una red de área local.

2.3 El papel del Web dentro del ciberespacio.

Como una aplicación que se usa en la Internet, el Web tiene un papel muy importante dentro de todas las comunicaciones en línea. Esta sección presenta el papel del Web en el ciberespacio[2] como un camino para ayudara comprender el desarrollo del Web y verlo como un sistema de comunicación en la red, nótese que la comunicación con los gateways permiten transferir datos a través de la red y los sitios en donde se encuentra la información definida por el protocolo.

2.3.1 La topología del ciberespacio.

El ciberespacio puede ser considerada como el más grande contexto en comunicación en línea a través de computadoras.

[2] El termino ciberespacio es usado para referirnos a la colección de computadoras que son usadas para la visualización, interacción, y recuperación de información.

La estructura para el ciberespacio consiste de una gran variedad de redes globales así como también sistemas que no pertenecen a la red, para comunicaciones e interacciones. (Como un CD-ROM). El uso del Web en forma global requiere de la comunicación en línea, ahora la discusión se enfoca dentro de la topología de la región en línea del ciberespacio. En esta región en línea, se encuentran miles de redes y sistemas por todo el mundo que los usuarios hacen uso para intercambiar información y para comunicarse. Estos sistemas y redes pueden usar diferentes protocolos para el intercambio de información, y pueden usar diferentes medios para la transmisión de mensajes (que va desde cable de cobre hasta cable de fibra óptica, satélites y otros sistemas de comunicación). Estas redes pueden también variar de tamaño.

2.3.2 La Internet y el Web dentro del ciberespacio.

Dentro del gran contexto global, el ciberespacio en línea, en donde hay muchas redes de computadoras que permiten a cualquier persona el intercambio de información y la comunicación. La Internet es referida como un sistema global de comunicación y difunde la información. Las aplicaciones de información en Internet también se basan en la recuperación de información en el Web. Es por eso que el Web se considera dentro de la Internet. El Web es una aplicación que es usada como herramienta en la Internet como un medio de comunicación y transporte de información.

2.3.3 El Web dentro de Internet.

El poder del Web consiste en ligar los recursos que se encuentran en la Internet a través del sistema de hipertexto. El lenguaje de señalización de hipertexto (HTML) es lo utilizado para este fin. Un archivo HTML contiene ligas hacia otros recursos dentro de la Internet. La figura 2.5 ilustra las conexiones dentro de un documento HTML hacia otros recursos y como son relacionadas a través de un navegador, los servicios de información, y los archivos localizados en el servidor.

Los recursos mostrados en la figura 2.5 incluyen una conexión remota hacia un servidor a través del protocolo Telnet, una liga hacia un archivo que se encuentra en un servidor FTP, una liga hacia un menú dentro de un servidor Gopher y una liga hacia otro documento HTML que se encuentra en otro servidor de Web. De este modo, el Web liga diferentes recursos esparcidos a través de la Internet.

2.4 URL sitios de información en el Web.

Las relaciones de información en Internet a través de ligas es la característica principal del Web. Para referirnos a estos recursos se creo un esquema de sitios de información estructurada llamado Localizador de Recursos Uniformes o URL.

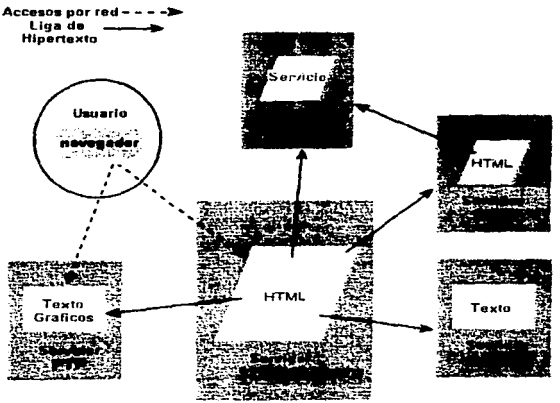

Figura 2.5 Ligas hacia los recursos de Internet.

Un URL consiste de una cadena de caracteres que identifican a un único tipo de recurso. Cuando un navegador abre un URL particular, el usuario tiene acceso al recurso referido por el URL.

El formato básico para muchos (pero no todos) de un URL es el siguiente:

```
esquema://host:puerto/ruta
```

en donde esquema es la regla o protocolo que se va utilizar para recibir o enviar información, como es FTP, Gopher, Telnet u otros.

host es la computadora anfitriona en donde residen los recursos.
puerto es un numero en particular que identifica la petición del servicio desde el servidor; este numero es proveído si el servicio se encuentra instalado en un puerto distinto al de por omisión para ese servicio.
ruta es la localización en donde se encuentra el recurso en particular dentro de la computadora anfitriona.

Capítulo III

Conectividad de Sybase con servicios del Web

3.1 La información y las Bases de Datos.

Sabemos que vivimos en la Era de la Información. A la fecha, hemos producido información al por mayor, generando datos acerca de cualquier cosa imaginable. Todos los días se producen teorías, se hacen descubrimientos y se gestan tendencias para informar al público. Esta información a ido a parar a bases de datos.

Los manejadores de Bases de Datos permiten mantener un control y un adecuado mantenimiento de los datos. El manejador de Base de Datos *Sybase* permite dar integridad a los datos, velocidad de acceso o eficiencia en la recuperación de información.

Para obtener información de una base de datos, se utiliza un lenguaje de consultas llamado *Structured Query Languaje* o SQL[1].

La mayoría de las bases de datos vienen con una Interfaz para programación de Aplicaciones (API) que proporciona a los programadores llamadas a funciones para comunicarse con la base de datos. Los programadores pueden crear una interfaz en la cual los usuarios pueden insertar datos, los cuales se alimentan a la base utilizando las llamadas a función provistas. Para informar si la adición de datos se completó o falló, se devuelve un código que explica que sucedió.

3.2 El Web y las Base de Datos.

Por el crecimiento de las empresas y el respaldo que nos da un manejador de base de datos como Sybase. Se necesita un lenguaje estándar que funcione igual en cualquier máquina, sin importar hacia dónde transfiera su base de datos, ni desde qué máquina trata de accesar la base. El Web podría ser la solución, sea que usted desee accesar su información desde cualquier parte del mundo o simplemente desde una red local.

[1] SQL permite a los usuarios elaborar enunciados de consulta precisa y compleja para realizar búsquedas y presentar la información que coincida con las restricciones de la consulta.

Como lo mencionamos en el capítulo anterior lo que hace al Web lo que hace al Web tan atractivo es:

1. Escritura de aplicaciones el lenguaje estándar: HTML.
2. Interfaz Gráfica de Usuario (GUI).
3. Soporte de plataforma cruzadas.
4. Soporte de red.

Todas estas características dan al usuario potentes herramientas para accesar sus datos con un mínimo de gasto y esfuerzo y, utilizando herramientas que funcionan sin importar el tipo de base de datos, sistema operativo o equipo de cómputo que se utilice.

3.3 Evaluación y características de las bases de datos.

Las características ofrecidas por el manejador de base de datos satisface o no las necesidades. Las que se consideran como evaluación es:

1. Control de concurrencia de base de datos.
2. Soporte de transacciones.
3. Soporte de tipos de datos.
4. Definición de tipos de datos personalizados.
5. Cumplimiento de requerimientos ANSI.
6. Vistas.
7. Restricción de datos.

3.3.1 Control de concurrencias.

En un ambiente multiusuario la base de datos debe de asegurar la integridad de la información cuando múltiples usuarios intenten el acceso a los mismos datos

simultáneamente. Esto significa que la información debe permanecer completa y precisa todo el tiempo.

En este tipo de ambientes es inevitable que un proceso de una base de datos se interrumpa antes de que termine, para dar paso a la ejecución de otro, el cual también puede detenerse para dar paso a otro más antes de que la CPU regrese al proceso original. Esto es inevitable y es una de las razones principales por las que alguna forma de bloqueo de las bases de datos es muy importante. Un *bloqueo* evita que otros procesos de base de datos tengan acceso a los datos hasta que el primero haya finalizado completamente la alteración de éstos.

Si todos estos procesos simultáneos trabajan sobre diferentes datos, no hay problema. Si lo hacen sobre datos que se traslapan, y la base no cuenta con algún sistema de bloqueo, pueden suscitarse serios problemas.

Para evitar este tipo de alteración de datos, una base utiliza un bloqueo para proteger la información.

Pero un mal manejo de bloqueo puede crear conflictos entre operaciones de base de datos y, consecuentemente, decrementar el desempeño. Sybase utiliza el método de bloqueo a nivel de página.

Sybase almacena físicamente los datos en el disco duro, así la información se guarda en unidades llamadas *páginas*. Una página puede contener muchos registros o parte de un registro. La figura 3.1 nos muestra como Sybase bloquea la página completa si un solo registro se está modificando. Cualquier otro proceso que intenta accesar un registro diferente en esa página se mantiene en espera de que la primera operación se lleve a cabo.

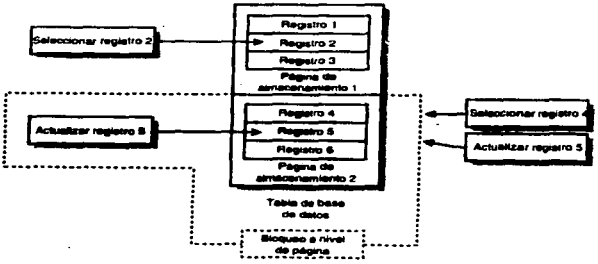

Figura 3.1 Bloqueo a nivel de página.

3.3.2 Soporte de transacciones.

Las operaciones de base de datos frecuentemente pueden actualizar información en muchas partes de la base al mismo tiempo. Las actualizaciones tienen que realizarse en un cierto orden, y si cualquier segmento de la operación falla, todo lo que se hizo hasta ese punto se deshace para que la base de datos quede en su estado original. Una operación inconclusa puede dejar a los datos en una condición inservible. A pesar de que el motor de la base puede poner un bloqueo sobre un solo registro, si una operación afecta a varios registros esparcidos en toda la base, los bloqueos tradicionales no pueden proteger a todos los datos al mismo tiempo. Se requiere de otro mecanismo para proteger grandes cantidades de datos, incluso si varias operaciones se ejecutan una tras otra.

El mecanismo usado se llama transacción. Si una operación se inicia y falla antes de completarse, los cambios efectuados hasta ese punto retroceden regresivamente (*rolled back*) de manera que los datos permanezcan intactos. Algunas bases de datos aceptan

transacciones que pueden revertirse. Esto de gran importancia interactuando con Internet, ya que se aseguran toda operación que se realice a la base de datos.

Sybase admite transacciones *definidas por el usuario*; en donde el usuario define dónde comienza y dónde termina una transacción. Si una transacción comienza y tiene un problema en cualquier punto antes del final, todos los cambios se revierten. Esto permite a un usuario ejecutar múltiples comandos de base de datos y cambiar datos en cualquier parte de la base con la misma protección ofrecida por los tipos tradicionales de bloqueo.

3.3.3 Soporte de tipo de datos.

Los tipos de datos que soporta una base de datos son de gran importancia cuando se tienen requerimientos especiales de información.

La tabla 3.1 muestra los tipos de datos con los que cuenta Sybase.

3.3.4 Vistas.

La información almacenada en la base puede incluir una amplia variedad de aspectos funcionales, y los usuarios podrían estar interesados sólo en ciertas partes de la información.

Una vista es una forma alternativa de ver la información de la base de datos. Se utilizan las vistas para limitar la cantidad de información que le estará permitido ver a una persona o para presentar datos de manera diferente, dependiendo de la finalidad de cada usuario. Las vistas también evitan tener varias bases de datos, conteniendo toda la información en una sola y definiendo vistas diferentes para desplegarla.

En Sybase una tabla puede soportar múltiples vistas de datos contenidos en una o a través de un conjunto de tablas.

Tipo	Almacenamiento	Rango	Muestra
int	4 bytes	$-2^{31} \leftrightarrow 2^{31} - 1$	476536
smallint	2 bytes	$-2^{15} \leftrightarrow 2^{15} - 1$	23335
tinyint	1 byte	$0 \leftrightarrow 255$	88
float	8 bytes	$\pm 10^{-308} \leftrightarrow \pm 10^{307}$	3.1417
real	4 bytes	$\pm 10^{-308} \leftrightarrow \pm 10^{307}$	1.5723e10
money	8 bytes	±$922,337,203,685,447.5807	$49.95
smallmoney	4 bytes	-214, 748.3648 ↔ +214, 748.3648	
char(n)	n bytes	0 ↔ 255 caracteres	"nieve"
varchar(n)	longitud de la cadena variable	0 ↔ 255 caracteres	"Fast Track to Sybase"
binary(n)	n bytes	1 ↔ 255 caracteres	0xf3c6
varbinary(n)	longitud de los datos	1 ↔ 255 caracteres	0x7f
text	longitud del texto	2K ↔ 2000 millones de caracteres	
image	longitud de la imagen	2K ↔ 2000 millones de caracteres	
datetime	8 bytes	Jan 1, 1753 00:00 ↔ Dec 31, 9999 23:59	"3-19-86"
smalldatetime	4 bytes	Jan 1, 1900 ↔ June 6, 2079	
bit	empaquado en bytes	0 ó 1	1

Tabla 3.1 Tipos de datos en Sybase.

3.3.5 Restricción de datos.

Si la exactitud de sus datos es de importancia crucial, o si necesita evitar errores tipográficos que pudieran suceder durante miles de transacciones, la base de datos debe de incluir la capacidad de aplicar restricciones de datos.

3.4 Características del manejador de Bases de Datos Sybase.

Las principales características de Sybase son:

1. Administra los Datos y la memoria
2. Administra diversas Bases de Datos y múltiples Usuarios

3. Mantiene un mapeo de la descripción lógica de los datos hacia el almacenamiento físico de los datos
4. Mantiene los datos y procedimientos en memoria principal
5. EL servidor SQL entiende el lenguaje SQL
6. Compila y ejecuta procesos "batch" de instrucciones SQL
7. Entrega los resultados a los programas de los clientes
8. Mejora el desempeño de los procesos reduciendo la sobrecarga del S.O.
9. El diseño del Servidor reduce el tráfico de la red significativamente.
10. El servidor SQL administra por sí mismo a múltiples usuarios
11. Determina automáticamente como realizar las tareas de las bases de datos de forma más eficiente

3.5 Consulta de información de una base de datos a través del Web.

Cuando alguien usa un navegador para accesar una base de datos, hay varios componentes que intervienen para transferir la consulta del usuario a la base de datos y devolver los resultados al navegador. La acción se desarrolla de la manera siguiente:

1. Un usuario llama un programa para conectarse al Web y a una base de datos (*gateway*).
2. El navegador reúne la información escrita por el usuario y la envía a un programa CGI.
3. El navegador contacta al servidor HTTP en la máquina donde reside el programa CGI, pidiéndole que localice al programa CGI.
4. El servidor HTTP corrobora si el solicitante tiene autorización de acceso al programa CGI.
5. Si el usuario tiene acceso, el servidor HTTP localiza el programa gateway y le transfiere la información. La figura 3.2 muestra los pasos del 1 al 5.

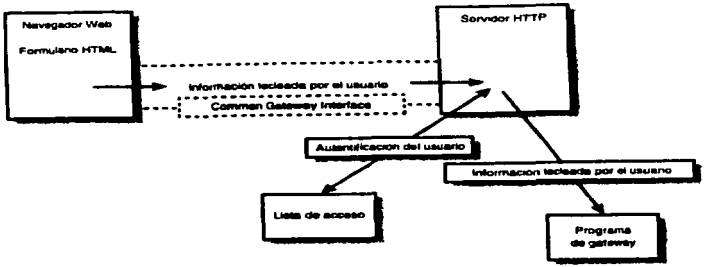

Figura 3.2 Conexión entre un cliente y un CGI.

6. Se ejecuta el programa gateway.
7. El programa gateway convierte la información recibida a un formato que la base de datos sea capaz de entender.
8. El gateway usa el módulo de la base de datos para transferir la consulta al software de la base de datos (*interfaz*) de la base.
9. La interfaz encuentra de la base de datos analiza la sintaxis de la consulta para asegurar que sea precisa.
10. Si se encuentra un error de sintaxis en la consulta, se envía un mensaje de error al programa gateway.
11. El mensaje se envía al servidor HTTP y se le despliega al usuario. Y el proceso se detiene aquí. La figura 6.3 muestra los pasos de 6 al 11.
12. Si no hay error, la interfaz envía la consulta a la base de datos.
13. La base de datos atiende la consulta y devuelve los resultados al programa gateway a través de la interfaz.
14. El programa gateway formatea los resultados y los envía al servidor, por medio del CGI.

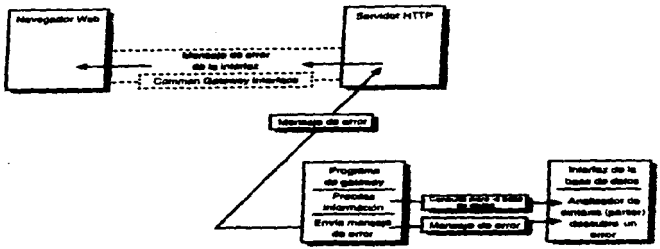

Figura 6.3 Error al consultar la Base de Datos

15. El navegador Web despliega los resultados. Los pasos 12 al 15 se muestran en la siguiente figura 3.4:

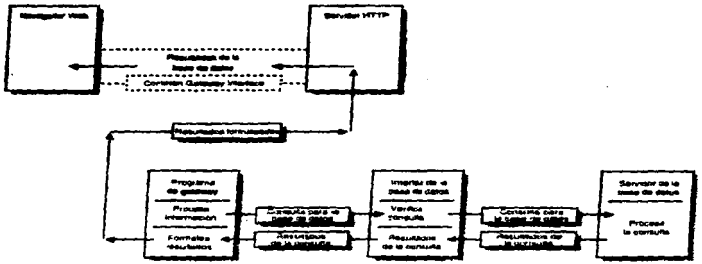

Figura 3.4 Consulta a la Base de Datos a través del CGI.

En muchos casos, el módulo de la base de datos y la interfaz de ésta son partes integrales del software de gateway. El módulo puede ser una librería Perl de subrutinas que se analiza como parte del programa Perl de gateway. En otros casos, la interfaz puede ser una librería C compilada o vinculada en un programa de gateway ejecutable. Aunque el módulo o la interfaz de la base es algo necesario para dar conectividad a la base, no necesariamente existe como un programa independiente.

Capítulo IV

Configuración del Servidor para Web

4.1 Servidor HTTP.

Un servidor de HTTP[1] es un programa encargado de responder a requisiciones que se efectúan a través del puerto 80, según la definición del protocolo HTTP. Dado la URL de una requisición, el servidor debe encontrar el recurso, archivo o programa que corresponde a dicha URL y si es necesario, ejecuta los programas CGI (Common Gateway Interface) y SSI[2] (Server Side Includes) asociados a dicha página. Un programa CGI se ejecuta localmente en el servidor. La información que produce es servida al cliente como respuesta a su requisición. Así mismo el servidor se encarga de labores de autentificación y seguridad en los casos en que sea requerido.

4.2 Elección de un servidor de HTTP.

La elección de un servidor de *http* depende de muchos factores como son:

1. dinero disponible para comprarlo
2. sistema operativo que se utiliza
3. si se requiere de transacciones seguras, etc.

Pero las estadísticas muestran que el mercado es dominado por el software de dominio público. Según estimaciones, los servidores de NCSA y de CERN tienen más del 70% del mercado. Ambos son robustos y ofrecen características que satisfarían las necesidades de la mayor parte de los sitios no comerciales ya que son rápidos y ofrecen diversos niveles de acceso a las páginas (a través de passwords y de nombre de dominio).

Otros servidores que están tomando popularidad son Netscape Communications y Apache. En nuestro caso utilizaremos el servidor NCSA HTTPd.

Este servidor desarrollado por el NCSA de la Universidad de Illinois en Urbana-Champaign. La versión que se utilizó es la versión 1.5, de la que se destacan las siguientes

[1] Acrónimo de HyperText Transport Protocol (que es el protocolo de transferencia de hipertexto), que es el protocolo que permite la conexión con un servidor de http y servir páginas HTML.
[2] Acrónimo de Side Server Include, que son programas que se ejecutan durante el acceso a una página Web.

características: es pequeño, rápido e incluye muchas funciones para restringir el acceso a la información que éste contiene.

4.3 Instalación del servidor NCSA HTTPd.

La instalación del servidor consta de cuatro pasos.

1. Bajar y compilar el servidor.
2. Configurar el servidor.
3. Verificar accesos.
4. Levantar el servidor.

4.3.1 Compilación del servidor.

El servidor que se instaló es la versión 1.5 del servidor de *httpd* de la NCSA. En una máquina SpartStation 5 con Solaris 2.4. Al bajar el archivo se procedió a descomprimirlo (uncompress) y separar (tar -xvf). Es una buena idea no ser *root*, sino hasta que sea imprescindible.

A continuación hay que correr el comando `make` dos veces, una primera vez para que nos muestre la lista de sistemas operativos para ver si se modificará el archivo *Makefile*, como nuestro sistema operativo si se encuentra en la lista, no se hace nada al archivo. La segunda vez debemos correr `make Solaris2`, el cuál compilara todo lo necesario.

4.3.2 Configuración del servidor.

Para modificar los archivos de configuración se deben de considerar las siguientes reglas :

1. Es indistinto el uso de mayúsculas o minúsculas (excepto en los nombres de directorios, ya que en Unix sí son distintos).
2. Los comentarios comienzan con el signo #.
3. Los espacios extras se ignoran.
4. Cada línea debe contener a lo más una directiva.

4.3.2.1 Configuración básica del archivo httpd.conf

Este archivo se encarga de la configuración del servidor para que este pueda arrancar.

Para levantar el servidor de httpd de NCSA en forma standalone para él publico en general y que no tenga restricciones hacia él publico, se tiene que realizar las directivas que se tienen que modificar son:

1. Modificar *User* y *Group* con cualquier nombre que se quiera. Puede ser definido con él id del usuario y él id del grupo que tendrá los permisos del servidor standalone usa cuando corre. Se recomienda el usuario *nobody* y el grupo -1, que son dados por omisión.
2. Asegurarse que los nombres existan en él `/etc/passwd` y `/etc/group`, respectivamente.
3. Modificar `ServerAdmin` y poner la dirección de correo electrónico del administrador del servidor. Es con esta persona donde se comunicarán para reportar los problemas que puedan suceder con el servidor de *httpd*.
4. Cambiar `ServerRoot` por la ruta absoluta donde se planea poner el programa *httpd*. Aquí también sé donde estarán otros archivos relacionados, aunque este archivo podría no contener estos archivos.
5. Especificar el nombre del servidor en `ServerName`. Aunque también se puede definir un alias.
6. Cambiar `ServerType` si se desea que el servidor corra como *inetd* en lugar de *standalone* (que es el modo por omisión).

Las directivas anteriormente mencionadas significan lo siguiente:

ServerType. Si bien el servidor puede correr a través de *inetd*, es mejor utilizar *standalone*, pues en el primero ejecutará una copia del servidor por cada requisición y en la segunda se ejecutará una sola vez y permanecerá siempre en memoria.

Port. El estándar es utilizar el puerto 80. HTTPd puede ser ejecutado por cualquier usuario sólo si se utiliza un puerto arriba del 1024.

User. Para maximizar la seguridad es importante que tenga un usuario en su máquina llamado *nobody* o también puede ponerse el valor de -1.

Group. Igualmente que en `User` hay que asignar el grupo *nobody* o -1.

ServerAdmin. Contiene la dirección de correo electrónico del administrador del servidor del Web.

ServerRoot. Es la ruta del directorio en donde se instalara el servidor del Web.

4.3.2.2 Configuración básica del archivo srm.conf.

Este archivo se encarga de la configuración de recursos, controla donde el servidor de *httpd* encuentra los documentos, los iconos y los scripts (programas cgi's), para inicializar el servidor sin restricciones de acceso al público, se tiene que :

1. Modificar la directiva `DocumentRoot` al punto superior en donde se quiere que sea nuestro árbol de documentos. Este será el directorio principal donde el *httpd* puede ofrecer sus documentos.
2. Modificar la directiva `UserDir` y para deshabilitar donde se inicie el servidor. Esto evita que los usuarios ofrezcan archivos al publico desde su home directories (un potencial resquicio de seguridad).
3. Modificar las directivas `Alias` o `ScriptAlias` si se desea poner el subdirectorio *icon* o *cgi-bin* en una localización que no sea la de por omisión. Adicionalmente estas directivas que algunos documentos y scripts se encuentren en otro directorio. Estos también se conocen como directorios virtuales.

4.3.2.3 Configuración básica del archivo access.conf.

Este archivo se encarga del control de acceso global. Esta configuración controla que tipo de accesos tienen los clientes servidores Web ya sea total o ha ciertos directorios solamente. A menos que se haya modificado el *srm.conf*, no será necesario hacer ninguna modificación a este archivo.

Los parámetros por omisión del archivo access.conf es :

```
<Directory /usr/local/etc/httpd/cgi-bin>
     Options Indexes FollowSymLinks
</Directory>
<Directory /usr/local/etc/httpd/htdocs>
     Options Indexes FollowSymLinks
     AllowOverride All
     <Limit GET>
          order allow,deny
          allow from all
     </Limit>
</Directory>
```

Estos parámetros por omisión permiten mostrar todo al browser sin ninguna precaución de seguridad. El control de seguridad acerca del control de accesos se verá en un capítulo posterior. Por ahora, para inicializar el servidor sin restricciones de acceso al público, se modifican las directivas siguientes :

1. El primer `Directory` si el directorio *cgi-bin* se encuentra en un directorio distinto.
2. `Options` directiva asociada con el directorio *cgi-bin* para remover la opción `Indexes`. Esta no es una buena idea el permitir que los usuarios puedan poner sus scripts en cualquier parte del directorio.
3. El segundo `Directory` si no coincide con la ruta que se definió en `DocumentRoot` en el archivo de configuración *srm.conf*.
4. Modificar `AllowOverride` de `All` a `None`. Así se evita que otros usuarios puedan escribir sin resguardo de seguridad.

4.3.3 Instalación del Servidor.

Terminando de verificar que el servidor del *http* funciona correctamente se puede mover el servidor de `httpd` y los archivos y directorios hacia el `ServerRoot` previo para la inicialización del servidor del *http*. Asumiendo que ya no se tenga que modificar la localización de `ServerRootX` y desempaquetar el servidor en el directorio deseado, se puede usar los siguientes comandos :

```
%cd /usr/local/src/httpd_1.3
%mkdir /usr/local/etc/httpd
%cp -r httpd conf logs icons cgi-bin /usr/local/etc/httpd
```

Hay que recordar, que el directorio *logs* podrían ser escritos por los usuarios cuando el servidor este corriendo (como se define en la directiva `User` dentro de la configuración del archivo *httpd.conf)*. El camino más fácil para arreglarlo es cambiar los permisos:

```
%cd /usr/local/local/etc/httpd
%chown http logs
```

4.3.4 Inicialización del servidor.

Uno puede inicializar el servidor del *http* de diferente manera dependiendo de que uno quiera correr el servidor bajo `inetd` o `standalone`. Básicamente es buena idea correr bajo `inetd` mientras se tenga solo para examinar el servidor y tenerlo como prototipo, pero cuando se tenga un trabajo pesado es mejor usar `standalone`.

Pero antes de levantar el servidor, resumiremos las tres opciones del comando en línea del `httpd`. Pero hay que recordar que le servidor tiene diferentes opciones para trabajar, pero estos son controlados desde los archivos de configuración. Las tres opciones para ejecutar el comando `httpd` son:

-d directory Si uno cambia el `ServerRoot` en el *httpd.conf*, uno puede también especificar el lugar del directorio desde esta comando. Esto se controla mediante el `httpd` verificando los archivos de configuración.

-f file De esta manera sé especifica una manera alternativa del archivo de configuración *httpd.conf* para que sea leído por el `httpd`.

-v Con esta opción se muestra la versión del servidor de `httpd` que se está utilizando.

4.3.4.1 Inicialización del servidor bajo Standalone.

Si uno utiliza todos los parámetros por omisión de los archivos de configuración, para inicializar el servidor, simplemente hay que ejecutar el binario como root:

```
%httpd &
```

httpd al ser inicializado como root utilizará el puerto 80 (que es el puerto reservado para este servicio) y abrirá los tres archivos de log. A continuación hay que cambiar los permisos de UID para el grupo/user especificado por las directivas de *Group* y *User* del archivo de configuración *httpd.conf*, por razones de seguridad.

Si uno utiliza el `ServerRoot` en otra ruta que no sea la de por omisión (`/usr/etc/local/etc/httpd`) se necesita iniciar el demonio de la siguiente manera:

```
%httpd -d /ruta_del_servidor &
```

en donde /ruta_del_servidor es lo puesto en la directiva `ServerRoot`.

Si se desean hacer cambios a los archivos de configuración, especialmente de donde se ponen los datos y los scripts se necesita reinicializar el servidor. Para esto se necesita matar el proceso de la siguiente manera:

```
% kill -9 `cat ./logs/httpd.pid`
```

A continuación se debe de volver a levantar al servidor como anteriormente se realizó.

4.3.4.2 Inicialización del servidor bajo inetd.

Si se modifica la directiva `ServerType` en la configuración del archivo *httpd.conf*, para iniciar el servidor bajo `inetd`, se debe de realizar lo siguiente :

1. Editar `/etc/services` para agregar la línea que se asemeje a :

```
http port-number/tcp
```

en donde *port-number*, es el puerto especificado en la directiva *Port* del archivo *httpd.conf*. Normalmente es le puerto 80.

2. Editar */etc/inetd.conf* para agregar la línea semejante a :

```
http stream tcp nowait nobody /usr/local/etc/httpd/httpd httpd
```

Reemplazar `/usr/local/etc/httpd/httpd` con la ruta en donde se encuentra el binario del servidor, y *nobody* con el nombre del usuario del cuál cumpla para las peticiones. La referencia final `httpd` (solo) se necesita con los argumentos de `-d ServerRoot` cuando esta directiva fue modificada dentro del archivo *httpd.conf*.

3. Reiniciar `inetd` para encontrar el proceso inetd se utiliza el comando `ps`.

Usando este modo, nunca se va a necesitar reiniciar el servidor *http*, como esto es reiniciar cuando es necesario por el proceso `inetd` . Por supuesto, bajará el desempeño del servidor ya que tendrá que leer todos los archivos de configuración al ser reinicializado.

Por último como *root* se necesita hacer lo siguiente al terminar de levanta el servidor:

Hay que crear los directorios adecuados, ponerle los permisos adecuados, levantar el servidor y borrar aquellos archivos que ya no sirven.

Capítulo V

Seguridad y control de accesos para el Web

5.1 Riesgos al instalar un servidor del Web.

El instalar el servidor de Web puede generar los siguientes riesgos:

1. Usuarios no autorizados tengan acceso a documentos confidenciales. El principal objetivo de un servidor es hacer información disponible a quien desea leerla o a un público seleccionado, si el acceso es restringido. Pero no se debe tener cuidado de no permitir que se tenga accesos a archivos confidenciales y solo dejar disponible la información que se desea mostrar al público. Por lo que es importante escudar la información critica del sistema y no permitir a nadie el acceso a ella. Como lo son los archivos de *passwords*, etc.

2. Si se necesita tener un negocio por medio de Internet y este maneje números de tarjetas de crédito, es necesario que se maneje la criptografía.

3. No hay que permitir que se generen scripts que permitan a extraños ejecutar comandos que pongan en riesgo el sistema.

5.2 Control de accesos y autenticación de usuarios.

Los accesos pueden ser controlados usando los siguientes métodos:

1. Control de acceso a nivel de dominio, donde aceptar o rechazar las conexiones son basadas por la dirección Internet del sistema donde está corriendo el navegador.

2. Autenticación del usuario, donde la aceptación o rechazo de las conexiones están basado por una autenticación de una clave de usuario y un *password*[1].

3. Los accesos también pueden ser controlados usando una combinación de estos dos métodos.

[1] Password (clave de acceso o palabra de acceso). Palabra o clave privada utilizada para confirmar una identidad en un sistema remoto que se utiliza para que una persona no pueda usurpar la identidad de otra.

5.2.1 Archivos de control de acceso (access.conf) .

El control de acceso y la autenticación de usuario puede ser configurado para todo el servidor o basado para accesar de directorio en directorio.

1. Un control de acceso por todo el servidor y accesos por directorio es controlado por el ACF (Access Control File, Archivo del control de accesos). Este ACF en el servidor *httpd* de la NCSA es llamado *access.conf*.

2. Si se elige accesos por directorios, estos accesos por directorios son controlados por omisión por el archivo *.htaccess*. El archivo ACF por directorio puede ser cambiado usando la directiva `AccessFileName` en el archivo de configuración *srm.conf*. Los ACFs por directorio pueden restringir completamente todos los accesos por directorios.

El control de acceso por archivo no está disponible. Si se necesita que se proteja un archivo, se puede poner el archivo dentro de un directorio protegido.

Un ejemplo de como se puede tener un control de acceso en dos directorios que pertenece al árbol de directorios se muestra a continuación. Con la primera entrada, cada subdirectorio (`subdirectorio-1` y `subdirectorio-2`) tiene la directiva `<Directory>` en el ACF global:

```
<Directory /usr/local/etc/httpd/htdocs/subdirectorio-1>
      <Limit GET>
      order,deny,allow, &require directives
      </Limit>
</Directory>
<Directory /usr/local/etc/httpd/htdocs/subdirectorio-2>
      <Limit GET>
      order,deny,allow, &require directives
      </Limit>
</Directory>
```

Si cada acceso a cada directorio necesita ser controlado por diferente persona, ambos archivos *.htaccess* dentro del directorio directorio-1 y *.htaccess* dentro del directorio directorio-2 pueden contener:

```
<Limit GET>
     order,deny,allow, &require directives
</Limit>
```

Fuera de la sección `<Limit>`, pero dentro de la sección `<Directory>`, en el ACF global, la directiva `Options` permite cuáles son herramientas avanzadas que permite el servidor del Web, y `AllowOverride` define si se permite anular la autorización por directorios.

5.2.2 Control de acceso por niveles de dominio

Se puede permitir o negar el acceso a nuestro servidor del Web, a usuarios, basado en el dominio de la dirección Internet. Este mecanismo permite que se controle los accesos de una organización o departamento sin tener que considerar un nombre específico de usuario. El control de acceso por nivel de dominio no lo da totalmente el servidor del Web, debe de trabajar junto con el navegador del Web, con diferente autenticación.

5.2.2.1 Directivas del control de dominios

Los archivos del control de acceso son divididos en secciones por directorio con la directiva `<Directory>`. Sin tener que dividir el directorio, la directiva del control de accesos se encuentra dentro de la sección `<Limit>`.

Las directivas del control de accesos son:

1. `order` Define el orden en que las directivas `deny` y `allow` serán evaluadas dentro de la sección *limit*.
2. `allow` Define que anfitriones pueden accesar al directorio.
3. `deny` Define que anfitriones será negado el acceso al directorio.

5.2.2.2 Acceso general sin restricciones.

La configuración que se da por omisión se vio en el capítulo anterior (instalación del servidor del Web), está configuración es por omisión y permite que todos los usuarios puedan accesar al servidor, es decir sin restricciones.

5.2.2.3 Acceso de dominio local.

Para restringir el acceso en un campo local o una organización local. Se debe de modificar el archivo de configuración *access.conf*, asumiendo que se tomo por omisión de la directiva *DocumentRoot* . Después de modificar hay que volver a levantar el servidor.

```
<Directory /usr/local/etc/httpd/htdocs>
      Options Indexes FollowSymlinks
      AllowOverride None
      <Limit GET>
            order deny, allow
            deny from all
            allow from 132.248.*
      </Limit>
</Directory>
```

Si se compara con la configuración del control de accesos sin restricción se puede ver que:

1. La directiva `order` tiene como cambio de `allow, deny` a `deny, allow`. Esto expresa que *httpd* evalúa primero la directiva `deny` y entonces concede las excepciones basadas en la directiva .

2. La directiva `deny from all` fue agregada.

3. La directiva `allow` fue cambiada de `all` por el dominio local.

También se puede extenderse los accesos hacia subdominios como puede ser 132.248.27.*

Uno puede extenderse hacia un directorio en un nivel más bajo por lo que se va incluyendo en la sección `<Limit>`, esta sección se encuentra en un archivo llamado *.htaccess* , que se debe de encontrar en el subdirectorio que se quiere proteger.

```
<Limit GET>
      order deny, allow
      deny from all
      allow from 132.248.*
</Limit>
```

Para que se cumpla esta secuencia se debe de poner el archivo *.htaccess* en un subdirectorio específico, uno puede permitir que todos tengan acceso al servidor del Web, pero se puede limitar un área para que sólo tengan acceso las personas del dominio local. Está técnica también trabaja para proteger el directorio HTML personal, si se pone el archivo *.htaccess* en el subdirectorio `public_html`.

5.2.2.4 Acceso de organización múltiple.

Se puede permitir el acceso a un conjunto de organizaciones, las cuáles se les puede dar un acceso total al servidor del Web. Para lograr esto a la configuración del archivo `access.conf` se le debe de agregar en la directiva `allow` los segmentos que se quieran agregar, por ejemplo que se le quiera dar permiso a las organizaciones `132.248.*`, `155.247.*` y `188.23` la directiva `allow` tendría lo siguiente:

```
allow 132.248.* 155.247.* 188.23.*
```

De esta forma cualquiera que pertenezca a uno de los tres dominios podrá observar la información contenida en el servidor del Web, y denegar el acceso a cualquier otro.

5.2.3 Autenticación del usuario.

Mediante la autenticación de usuario, se puede permitir o negar el acceso al servidor del Web o a un documento en el árbol de directorios en base a un nombre de usuario y *password*.

Cuando el acceso a páginas es protegido con este mecanismo, se obtienen dos indicadores (nombre de usuario y *password*) los cuáles deben ser respondidos correctamente para que se permita el acceso. Una vez autentificado, cualquiera puede navegar de página a página sin tener que repetir la autenticación. Esto se debe a que el navegador del Web recuerda el nombre del anfitrión y la ruta del directorio, y nombre y *password* para subsecuentes recuperaciones.

La autenticación del usuario requiere la cooperación entre el navegador del Web y el servidor del Web, es decir, el navegador del Web debe soportar la autenticación del usuario.

Para el uso de la autenticación, se tiene que tener un archivo privado para el uso del hipertexto, que contenga el nombre del usuario y los *password*. Por convención, este archivo es llamado *.htpasswd*.

5.2.3.1 Manejo del archivo .htpasswd.

Para manipular el archivo *.htpasswd*, se necesita el programa `htpasswd`. Para obtenerlo se necesita compilar el código fuente *httpasswd.c* y este se encuentra dentro del subdirectorio `support/`. Para invocar el `htpasswd` se realiza lo siguiente:

```
%htpasswd [-c] .htpasswd username
```

username es el nombre del usuario del que se desea agregar o editar. La bandera -c, si se encuentra presente, forza al `htpasswd` a crear un nuevo archivo de *password* o editar uno ya existente.

Si el `htpasswd` encuentra el usuario especificado, este pregunta si se desea cambiar el *password* del usuario. Es necesario teclear dos veces el nuevo *password*. *httpd* entonces actualiza el archivo.

Es posible tener múltiples archivos *.htpasswd*, o puede tener diferente nombre. Pero cada archivo necesita encontrarse en los diferentes directorios que se desean proteger y que se localizan fuera de la estructura del árbol de documentos.

5.2.3.2 Autenticación individual.

La autenticación individual es realizada usando una combinación de directivas de accesos de control y el archivo privado *.htpasswd.*

Asumiendo que se utiliza por omisión el `DocumentRoot`, se puede modificar el archivo de configuración *access.conf* con los siguientes parámetros.

```
<Directory /usr/local/etc/httpd/htdocs>
     Options Indexes FollowSymlinks
     AllowOverride None
     AuthUserFile /usr/local/etc/httpd/htdocs/conf/ .htpasswd
     AuthGroupFile /dev/null
     AuthName By Unicamente el password secreto.
     AuthType Basic
     <Limit GET>
          require user username
     </Limit>
</Directory>
```

Los parámetros que se cambian comparados con la configuración sin restricción de acceso son:

1. La directiva `AuthUserFile` es agregada, para especificar la ruta absoluta del archivo de los *password*.

2. La directiva `AuthGroupFile` también fue agregada, pero fue puesto con `/dev/null`, que indica en un ambiente Unix que el archivo no existe.

3. La directiva `AuthName` especifica el indicador que se mostrará al usuario para el *username*. (En este caso : Unicamente el password secreto

4. La directiva `AuthType` tiene la única autorización permitida, por el momento, que es Basic.

5. Todas las directivas `Auth` deben localizarse fuera de la sección de directiva `Limit`.

6. Las directivas `order` y `allow` son quitadas de la sección de directivas `<Limit` y son reemplazadas por la directiva `require`. Esto indica al `httpd` que muestre el indicador de *username* y *password*. El usar la directiva `require` es necesario incluir las directivas `AuthUserFile`, `AuthGroupFile`, `AuthName` y `AuthType`.

A continuación, es necesario crear el archivo de *password* para especificar los usuarios que tendrán derecho a accesar y esto se especifica en la configuración del archivo `access.conf` de la siguiente forma:

```
%htpasswd -c /usr/local/etc/httpd/conf/ .htpasswd username
```

`htpasswd` se utiliza para el *password* del usuario. Hay que recordar que cualquier cambio en los archivos de configuración (incluyendo *.htpasswd*) para que tomen efecto se necesita reiniciar el servidor *httpd* (a menos que sé este corriendo el servidor bajo *inetd*). Después de reinicializarse, el acceso será restringido solo a las personas que conozcan el *username* y el *password*.

Si se quiere usar una autenticación individual sobre niveles de directorios, como proteger el directorio `/usr/local/etc/httpd/htdocs/subdirectorio-3`, se

necesita tener las siguientes directivas en el archivo *.htaccess* en el directorio que se desea proteger:

```
AuthUserFile /usr/local/etc/httpd/conf/ .htpasswd
AuthGroupFile /dev/null
AuthName Unicamente el password secreto.
AuthType Basic
<Limit GET>
     require user username
</Limit>
```

5.2.3.3 Autenticación por Grupo.

Además de la autenticación individual de usuarios, se puede invitar a grupos de usuarios completos. Los pasos para este tipo de proceso son:

1. Modificar el archivo *access.conf*.
2. Crear el archivo *.htgroup* con los usuarios que pertenecen al grupo.
3. Checar que las personas que se localicen en el archivo *.htgroup* se encuentren el archivo *.htpasswd*.

Un ejemplo podría ser:

```
<Directory /usr/local/etc/httpd/htdocs>
     Options Indexes FollowSymlinks
     AllowOverride None
     AuthUserFile /usr/local/etc/httpd/htdocs/conf/ .htpasswd
     AuthGroupFile /usr/local/etc/httpd/htdocs/conf/ .htgroup
     AuthName Unicamente el password secreto.
     AuthType Basic
     <Limit GET>
          require group groupname
     </Limit>
</Directory>
```

Hay que notar que, comparada con la configuración de protección individual, los cambios son:

1. La directiva `AuthGroupFile` se modifico de `/dev/null` por la ruta donde se localiza el archivo *.htgroup*, que en este caso se localiza en el mismo directorio del archivo *.htpasswd*.

2. La directiva `require` es modificada de `user` por `group` y de username por *groupname*.

3. A continuación hay que crear el archivo .htgroup en la ruta que se especifico (`/usr/local/etc/httpd/htdocs/conf/`), con cualquier editor de textos y definiendo a los elementos del grupo de la siguiente manera:

groupname: username1 username2 username3 usernameN

Los miembros del grupo son separados por espacios. Pueden haber diferentes grupos, pero estos deben localizarse uno por línea.

Si se definen personas asociadas a grupos es necesarios reiniciar el servidor para que los cambios tomen efecto.

Si se agregan personas al archivo *.htgroup*, se necesita también que agregues a las personas por medio del programa htpasswd para que estos se den de alta en el archivo .htpasswd. Por ejemplo:

```
%htpasswd /usr/local/etc/httpd/conf/ .htpasswd
%htpasswd /usr/local/etc/httpd/conf/ .htpasswd username
```

Nótese que htpasswd es invocado sin la bandera -c, ya que el archivo *.htpasswd* ya existe y los nuevos usuarios se agreguen.

5.2.3.4 Control de acceso simultáneo y autenticación de usuarios.

La autenticación puede ser usada en conjunción con el control de dominio de accesos. Un ejemplo, sería el de permitir accesos a personas que pertenecen al dominio `132.248` y además que sean de un grupo específico. Para hacer esto se tendría que modificar la sección `<Limit>` y quedaría de la siguiente manera:

```
<Limit GET>
     order deny,allow
     deny from all
     allow from 132.248
     require group groupname
</Limit>
```

Utilizando estos conjuntamente se tiene una mayor seguridad en el servidor Web.

5.3 Seguridad.

Cuando se configura el control de accesos hacia el servidor del Web, es porque se quiere restringir el acceso para cualquier persona que quisiera causar daño al servidor. Mas aun se necesita aumentar la seguridad cuando se tienen gateways[2] en el servidor.

5.3.1 Ligas fuera del árbol de documentos.

El servidor del Web no contiene una herramienta como las que contiene FTP y Gopher (chroot( )) que permite restringir el acceso a documentos que se localizan dentro del árbol de documentos. Sin embargo, los clientes Web no pueden ver documentos que se localicen fuera del área que fue definida como `DocumentRoot`. Pero esto solo se da si no se tienen ligas hacia otros documentos, ya que *httpd* no permite que los usuarios que se localicen bajo el árbol de directorio.

El mayor daño que puede tener un administrador del Web es permitir que los usuarios pongan ligas fuera del árbol de documentos. Para protegerse de esto, el administrador del

[2] Gateway. Script o programa invocado por un servidor del Web que acepta entradas y devuelve un documento HTML, un URL o cualquier otro tipo de dato que pueda devolver a través del servidor del Web.

servidor Web puede deshabilitar la opción `FollowSymLinks`[3] en el archivo *access.conf* o por lo menos cambiar por SymLinksOwnerMatch[4].

5.3.2 Control de accesos y autenticación de usuarios.

Con un control de accesos y la autenticación de usuarios hacen que tu servidor del Web sea relativamente seguro, pero no del todo ya que:

1. Con un control de accesos basado en dominios, el servidor del Web solo es seguro si el DNS (Servidor de Nombres) es confiable.
2. Con la autenticación del usuario, el *password* viaja en claro por la red. Por lo tanto la red debe ser confiable.

5.4 Seguridad en los cgi's.

Cuando se escriban programas o scripts CGI's se tiene que tener cuidado con los parámetros que pueda recibir, así mismo un CGI no debe permitir recibir comandos como parámetros, ya que esto pondría en una difícil situación al sistema, porque podría permitir ejecutar otros programas y este programa podría permitir accesos no autorizados o dañar el sistema (borrar archivos como un ejemplo). Es por eso que hay que revisar los CGI's para ver que sean seguros.

[3] Opción de las directivas de control que permite que el servidor permita seguir las ligas que se localicen dentro del directorio.
[4] Esta opción permite que el servidor siga las ligas en las que el dueño del archivo/directorio sea dueño del documento ligado.

Capítulo VI

Gateways y Formas

6.1 Introducción al Common Gateway Interface (CGI).

Uno de los elementos más potentes de HTML es el Common Gateway Interface (CGI). HTML contiene etiquetas que le permiten al usuario la elaboración rápida de documentos de aspecto profesional. Pero HTML no tiene etiquetas mediante las cuáles los usuarios puedan ejecutar programas con un navegador Web, procesar los resultados y presentarlos en un documento HTML. Como esto no se puede hacer los diseñadores de HTML diseñaron el CGI.

La especificación del Common Gateway Interface (CGI) permite a servidores Web ejecutar otros programas e incorporar en su salida texto, gráficas y audio que envían al cliente (navegador). El servidor y el programa CGI trabajan juntos para elevar las capacidades del Web.

Esto es de gran ayuda ya que en un principio el Web fue estático, gracias al CGI empezarón a ser dinámicos, informativos y útiles. Además el CGI permite al WWW interactuar con otras aplicaciones, servicios de información o bases de datos.

6.2 Definición de CGI.

Hay personas que definen al CGI como la interface entre el servidor Web y otros recursos de la computadora anfitriona en donde se localiza el servidor del Web.

El Common Gateway[1] Interface (CGI), es un estándar para programas que sirven de puente entre el Web y una aplicación. Esto se podría definir como una entrada estándar HTML que permite a los programadores ejecutar un programa escrito en cualquier lenguaje, como lo es el lenguaje C, Perl, scripts de Unix, etc. CGI proporciona a los programadores

[1] Gateway (Puerta de acceso). Enlace dinámico entre dos servicios telemáticos en línea que permite acceder a uno de ellos desde el otro. Son traductores de protocolos.

un modo de que las páginas Web en HTML puedan ejecutar programas externos y presentar resultados. Hay que resaltar que los programas siempre se ejecutan del lado del servidor.

Para que un programa común pueda convertirse en CGI tiene que generar como resultado (por la salida estándar) un documento HTML y debe cumplir con los lineamientos MIME para especificar que tipo de documento se generará, comúnmente text/html.

6.3 Aplicaciones de CGI's

Los CGI's se aplican frecuentemente en :

1. En combinaciones con acceso a bases de datos.
2. Como interprete de formas de recolección de información, donde el CGI recibe información proporcionada por el usuario, la selecciona y realiza alguna acción con ella, desde almacenarla hasta deducir algún hecho de ella.
3. Los CGI's se emplean también en robots, para localizar alguna información de manera autónoma, conectarse a diversas páginas o copiando páginas de un servidor a otro, traducir información de un tipo de formato a otro, como podría sé el caso de tomar información ASCII de un correo electrónico y pasarlo a formato HTML.

6.4 Proceso realizado detrás del CGI.

Existen varios pasos a seguir para que un programa funcione adecuadamente:

1. El cliente llama a un programa CGI haciendo un click sobre un vínculo u oprimiendo un botón.
2. El cliente solicita autorización al servidor de Web para ejecutar el programa CGI.
3. El servidor revisa la configuración y los archivos de acceso para permitir o negar el acceso al programa CGI.
4. El servidor verifica si existe el programa CGI.

5. Si existe el programa, éste se ejecuta.
6. Cualquier resultado producido por el programa CGI se devuelve al cliente.
7. El cliente Web despliega el resultado.

La figura 6.1 muestra el proceso que se realiza cuando se invoca un CGI.

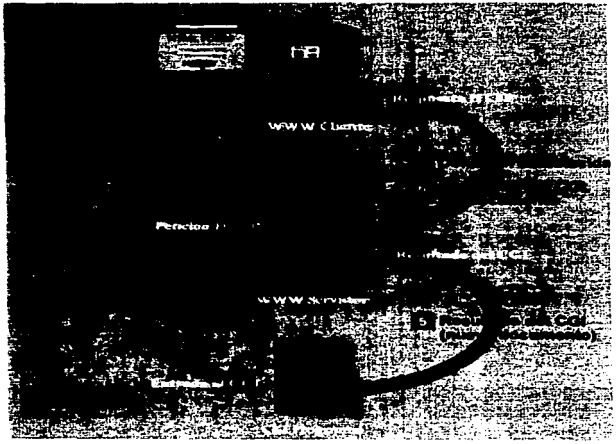

Figura 6.1 Proceso realizado cuando se invoca un CGI.

El cliente Web puede transferir información al programa CGI de varias maneras, y el programa puede devolver los resultados en formato HTML, como texto sencillo o como imagen. Esto proporciona una herramienta muy poderosa para ejecutar prácticamente cualquier programa, y permite a los programadores el acceso a cualquier base de datos externa, que proporcione una interfaz de programación. Además, en vez de diseñar un

cliente para una base de datos específica, el cliente queda igual, sin importar el tipo de base de datos que se utilice.

6.5 Recuperación de datos devueltos por un CGI.

Para que los datos puedan ser desplegados por el navegador el programa CGI debe llevar un encabezado en la primera línea del resultado, esto es para indicarle a este como debe desplegar el resultado del programa CGI.

Hay que hacer notar que después del encabezado debe seguir una línea en blanco que le indica al navegador que la sección de encabezado ha terminado y que cualquier cosa que continué debe procesarse como texto especificado.

Los encabezados pueden ser:

Content – type: text/html	Para documentos de texto con formato HTML.
Content – type: text/plain	Para documentos de texto sencillo.
Content – type: image/gif	Para imágenes con formato GIF.

Esos encabezados son parte del estándar Multiporpose Internet Mail Extensions (MIME). MIME fue creado originalmente para permitir que los mensajes de correo electrónico incluyeran tipos de medios diferentes de texto sencillo. Web utiliza también este estándar.

6.6 Lenguajes comunes para programación de CGI's en Unix.

Existen muchos lenguajes disponibles para escribir programas CGI. No hay un solo lenguaje que satisfaga las necesidades de cada programa. Los siguientes son los tres principales lenguajes utilizados para la programación CGI en ambiente Unix:

1. Un script shell de Unix son utilizados fundamentalmente para efectuar comandos del sistema. Se puede utilizar cualquier shell como C-shell, Korn shell, tsch, etc.
2. Un programa ejecutable escrito en lenguaje C es utilizado para la velocidad de ejecución o la seguridad del código fuente es un aspecto importante. Además los proveedores de base de datos proporcionan una Interfaz para Programación de Aplicaciones (API)[2] .
3. Un script en Perl[3] para la facilidad de operación y un conjunto enriquecido de funciones.

6.7 Métodos para transferir información de un navegador hacia un CGI.

La forma de transferir información hacia un programa CGI depende de cómo se llame al programa CGI en el documento HTML. Los siguientes métodos son utilizados:

1. Transferencia de parámetros en la línea de comandos.
2. Transferencia de variables de ambiente a programas CGI.
3. Transferencia de datos a programas CGI mediante entrada estándar.

Cada método se nombra dé acuerdo con la forma como envía información al programa CGI. El método utilizado determina la forma como debe leer la información el programa CGI y el tipo de procesamiento que debe hacerse con la información antes de que ésta pueda utilizarla el programa.

6.7.1 Transferencia de parámetros en la línea de comandos.

Un programa CGI puede leer parámetros de línea de comandos. El lenguaje HTML proporciona un método sencillo de línea de comandos llamado ISINDEX.. Este es la única

[2] API es generalmente es una librería de llamada de funciones que los programadores aprovechan para crear programas de acceso a base de datos.
[3] Perl es la abreviatura de Practical Extraction and Report Language. Este lenguaje de programación apoya las tareas comunes que tienen que ver con el "shell" del Sistema Operativo.

manera de enviar parámetros en línea de comandos a un programa CGI. Para utilizar este método el documento HTML y el programa CGI deben estar contenido dentro del mismo código. Ya que el utilizar ISINDEX no hay forma de indicar a ISINDEX donde enviar los parámetros de línea de comandos, y por lo tanto se envían al programa mismo.

6.7.2 Transferencia de variables de ambiente a programas CGI.

Un servidor del Web utiliza las variables de ambiente de manera similar a los ambientes de Unix o de DOS. El servidor del Web establece los valores de distintas variables de ambiente cada vez que se llama un CGI. El CGI puede accesar los contenidos de esas variables si las necesita y las usa dentro del programa.

Las variables de ambiente trasladadas por el servidor de Web y el CGI sirven para intercambiar información adicional que permite al cliente conocer más sobre los datos de entrada o estado del servidor y son:

1. Relativas al ordenador cliente:

REMOTE_HOST: Nombre del ordenador cliente que solicita la ejecución del CGI.

REMOTE_ADRESS: IP del ordenador cliente.

REMOTE_USER: Nombre del usuario que, desde el ordenador cliente, ha solicitado la ejecución del CGI.

AUTH_TYPE: Tipo de autorización asignada al usuario remoto.

HTTP_* :Conjunto de variables recibidas por el servidor en el encabezamiento de la solicitud del cliente, y que siempre tomarán la forma HTTP_Nombre_Variable

2. Relativas al ordenador servidor.

SERVER_NAME: Nombre o dirección IP del ordenador servidor encargado de ejecutar el CGI.

SERVER_PORT: Número de puerto por el que se gestiona la comunicación.

SERVER_PROTOCOL: Nombre y versión del protocolo de comunicaciones utilizado por el cliente y el servidor.

SERVER_SOFTWARE: Nombre y versión del software HTTPd empleado por el servidor.

GATEWAY_INTERFACE: Versión de la especificación CGI soportada por el software HTTPd.

SCRIPT_NAME: Ruta completa que, a través de distintos directorios, nos conduce al programa CGI cuya ejecución ha sido solicitada.

3.Relativas a la comunicación y recepción de datos:

QUERY_STRING: Toda la secuencia de caracteres que siguen a la <?> en la URL encargada de solicitar la ejecución del CGI y pasarle los parámetros especificados por el cliente.

REQUEST_METHOD: Al solicitar la ejecución de un CGI a través de un formulario HTML, esta variable adoptará los valores POST o GET, dependiendo del modo establecido para la transmisión de parámetros.

CONTENT_TYPE: Tipo de datos enviados por el cliente al servidor, generalmente "application/x-www-form-urlencoded".

CONTENT_LENGTH: Extensión de los datos enviados por el cliente al servidor.

Las tres últimas variables se utilizan exclusivamente en el caso en que la llamada al CGI se efectúe a través de un formulario HTML. Conviene señalar que, si nos servimos para la programación de CGI de las librerías disponibles para los distintos lenguajes, cada una de ellas referencia a estas variables con distintas designaciones.

6.7.3 Transferencia de datos a programas CGI mediante entrada estándar.

Si se desea utilizar formularios HTML para proporcionar a los usuarios campos de texto, casillas de verificación, listas de selección, cuadros de entrada de textos, y otras características interactivas disponibles en HTML, la información estructurada y la

transferencia de datos mediante entrada estándar es la manera como los formularios envían datos al servidor HTTP y al programa CGI.

Entrada, salida y error estándares son canales de datos proporcionados por el sistema operativo para conectar un dispositivo de entranda/salida (E/S) a un programa en ejecución.

A diferencia de cuando se utiliza el método ISINDEX, el servidor HTTP no procesa los argumentos antes de enviarlos al programa CGI. Los argumentos llegan codificados al programa tal y como los envía el navegador, y depende del programa decodificarlos al formato correcto. La transmisión de datos entre el cliente y el servidor sigue las siguientes convenciones:

1. Los espacios en blanco se convierten en el signo <+>.
2. Todos los caracteres se transforman en un número hexadecimal (equivalente al correspondiente ascii del carácter en cuestión), precedido por el signo <%>.
3. La información de cada campo va codificada en una colección de parejas de nombre de campo y contenido del campo separados por el símbolo '&' de la siguiente manera:

campo=contenido&campo=contenido&...

Una forma sencilla de decodificar la entrada transferida desde formularios HTML es utilizando Perl. Perl se está convirtiendo en estándar para la programación CGI, sobre todo en plataformas Unix. Perl combina las capacidades de los variados shells de Unix, capacidades de sed y awk, así como del lenguaje C. Lo que hace atractivo a este lenguaje es su uso de arreglos asociativos, de expresiones regulares y manejo de archivos.

Hay dos métodos que utilizan las formas para enviar información y son GET y POST.

El método GET lo utiliza ISINDEX y los argumentos se transfieren en la variable de ambiente QUERY_STRING. Cuando los formularios HTML se utilizan para enviar

información a la entrada estándar, el método utilizado es POST. Cuando se utiliza este último se necesita ver la variable ambiente CONTENT_LENGTH ya que contiene la longitud de los datos transferidos a la entrada estándar. Ya que el servidor HTTP no procesa los argumentos antes de enviarlos al programa CGI. Los argumentos llegan codificados al programa tal y como los envía el navegador, y depende del programa decodificarlos al formato correcto.

Una forma sencilla para decodificar la entrada transferida desde un formulario de HTML es utilizar el lenguaje de programación Perl. Perl se está convirtiendo rápidamente en el estándar para la programación CGI, principalmente en plataformas Unix.

El siguiente fragmento de código del programa en Perl permite codificar la información que envió el formulario HTML hacia el CGI. Este programa se encarga de separar los valores de la variable utilizada y su contenido. Sustituye el símbolo + por espacios en blanco y convierte los caracteres hexadecimales en sus equivalentes en caracteres ASCII.

```
#!/usr/local/etc/perl

sub ReadParse {
  local (*in) = @_ if @_;
  local ($i, $key, $val);

  # Lectura del texto
  if (&MethGet) {
    $in = $ENV{'QUERY_STRING'};
  } elsif ($ENV{'REQUEST_METHOD'} eq "POST") {
    read(STDIN,$in,$ENV{'CONTENT_LENGTH'});
  }

  @in = split(/&/,$in);

  foreach $i (0 .. $#in) {
# Conversion de los espacios en blanco
    $in[$i] =~ s/\+/ /g;
# Separacion de la variable y su contenido
    ($key, $val) = split(/=/,$in[$i],2);
# Conversion de los caracteres hexadecimales a valores ASCII
    $key =~ s/%(..)/pack("c",hex($1))/ge;
    $val =~ s/%(..)/pack("c",hex($1))/ge;
# Asocia el contenido de la variable con esta.
```

```
    $in{$key} .= "\0" if (defined($in{$key})); # \0 es el separador.
    $in{$key} .= $val;
  }
  return length($in);
}

sub ReadPath {
  local (*param) = @_ if @_;
  local ($i, $key, $val);

  # Lectura del texto
    $temp = $ENV{'PATH_INFO'};
    $param = substr($temp,1);

  @param = split(/&/,$param);
  foreach $i (0 .. $#param) {
    # Conversion de los espacios en blanco.
    $param[$i] =~ s/\+/ /g;
    # Separacion del contenido de la variable
    ($key, $val) = split(/=/,$param[$i],2);

    # Conversion de las variables en hexadecimal
    $key =~ s/%(..)/pack("c",hex($1))/ge;
    $val =~ s/%(..)/pack("c",hex($1))/ge;

    # Asociar la llave con el contenido de la variable
    $param{$key} .= "\0" if (defined($param{$key}));
    $param{$key} .= $val;
  }
  return length($param);
}
```

6.8 Formas.

Los formularios son un elemento esencial del CGI. Los formularios permiten introducir en un cliente del Web información estructurada que puede utilizarse como datos de entrada de una aplicación CGI. Para el soporte de formularios el lenguaje HTML permite la definición de distintos tipos de campos.

Un formulario en HTML se limita con los marcadores <FORM> y </FORM>. Un formulario puede incluirse en cualquier parte de un documento HTML, y sobre un mismo documento pueden insertarse distintos formularios.

La sintaxis del marcador de formulario es la siguiente:

<FORM METHOD="método" ACTION="URL del CGI">

- Método puede ser GET o POST.
- El URL del CGI es el nombre de la aplicación CGI que procesará la información del formulario.

6.8.1 Campos en formularios.

Los formularios en HTML soportan distintos tipos de campos. La sintaxis general del marcador de campo en HTML es la siguiente:

Los tipos válidos de campos de formulario son los siguientes:

<INPUT TYPE="tipo" NAME="nombre" VALUE="valor">

•text. Campo de tipo texto.

<INPUT TYPE="text" NAME="nombre" VALUE="valor" SIZE=20 MAXLENGTH=20>

•password. Palabra de paso. Lo que se introduce en este campo no es visible.

<INPUT TYPE="password" NAME="nombre" VALUE="valor" SIZE=15 MAXLENGTH=15>

•radio . Botón de selección redondo. Selección múltiple.

<INPUT TYPE="radio" NAME="nombre" VALUE="valor">

•checkbox. Boton de selección.

<INPUT TYPE="checkbox" NAME="nombre" VALUE="valor">

•submit. Botón de aceptación de la información introducida en el formulario.

<INPUT TYPE="submit" VALUE="Submit">

•reset. Borra el contenido completo del formulario.

<INPUT TYPE="reset" VALUE="Reset">

•image. Imagen que actua como boton.

<INPUT TYPE="image" SRC="ruta de la imagen" NAME="nombre">

•hidden. Campo oculto. No es visible al cliente WWW.

<INPUT TYPE="hidden" NOMBRE="nombre" VALUE="valor">

Estos controles admiten los modificadores siguientes, ya indicados:

•size. Tamaño del campo.
•maxlength. Longitud máxima que tendrá la variable.

Otros controles tienen una sintaxis especial.

La siguiente contrucción permite crear un control de selección sobre una lista de opciones predefinidas.

<SELECT NAME="nombre">
<OPTION >Texto de la seleccion 1
<OPTION >Texto de la seleccion 2
</SELECT>

El control <OPTION> permite los modificadores siguientes:

•Size.

El siguiente control permite introducir texto de cualquier longitud.

<TEXTAREA NAME="nombre">.
Texto que aparecerá en el recuadro. Es opcional.
</TEXTAREA>

Este control acepta los atributos específicos

•ROWS. Número de filas.
•COLS. Número de columnas.

Capítulo VII

Implantación de un servicio Cliente/Servidor a través del Web

IMPLANTACION DE UN SERVICIO CLIENTE/SERVIDOR A TRAVES DEL WEB

La pantalla de presentación nos pedirá que nos identifiquemos mediante un login y un password, como es mostrado en la figura 7.1. Este tipo de control de acceso se realiza por medio del servidor de httpd como se vio en la sección de control de acceso y autenticación de usuario. Utilizando el archivo .htaccess, que se localiza dentro del directorio protegido.

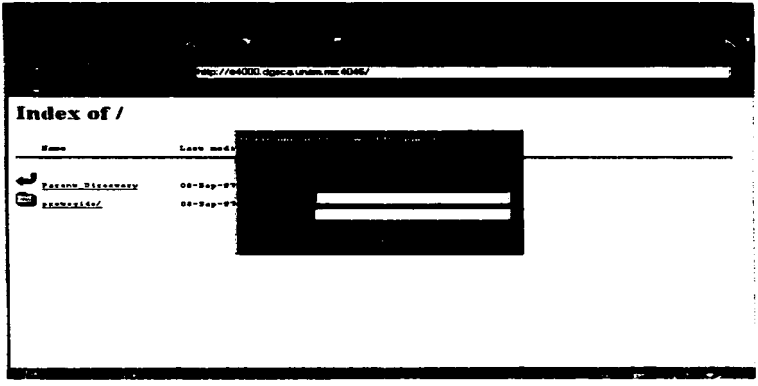

Figura 7.1 Página de entrada al directorio protegido.

El archivo .htaccess contiene las siguientes directivas:

```
AuthUserFile /usr/home3/cliserv/web/.htpasswd
AuthName "Favor de introducir login y password"
AuthType Basic
<Limit GET>
  require user donaciano ruben adriana
</Limit>
```

Si la contraseña y el password son correctos, permitirá al usuario ver la pagina1.html como se muestra en la figura 7.2.

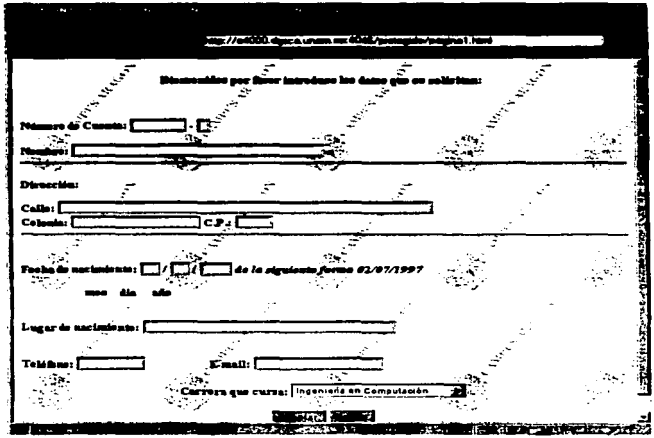

Figura 7.2 página de introducción de datos.

El script pagina1.html permite introducir información a la base de datos.

Al introducir los datos que se solicitan estos serán enviados al CGI evaluar_entrada.pl, que nos permitirá validar los datos y al mismo tiempo se encarga de enviar los datos a un programa escrito en lenguaje C el cual insertará los datos a la tabla ALUMNOS que se encuentra en la base de datos ENEP.

El código en Perl de evaluar_entrada.pl es el siguiente:

```
#!/usr/bin/perl
use CGI;
$query = new CGI;
print $query->header;

#CGI que permite recoger los datos que manda la pagina1.html, valida si
#los datos son correctos y los envia al programa introducir.
#Manda mensajes de error si los hubiera.

$cta1 = $query->param('cuenta1');
$cta2 = $query->param('cuenta2');
$nombre = $query->param('nombre');
$calle = $query->param('calle');
$colonia = $query->param('colonia');
$cod_postal = $query->param('cod_post');
$mes = $query->param('mm');
$dia = $query->param('dd');
$ano = $query->param('aa');
$lugar_nac = $query->param('lugar');
$telefono = $query->param('tel');
$mail = $query->param('mail');
$carrera = $query->param('carrera');

if (! $ENV{'SYBASE'}) { $ENV{'SYBASE'} = "/sybase/system_XI";}
if (! $ENV{'DSQUERY'}) { $ENV{'DSQUERY'} = "CSR_SYBASE";}

$num_cuenta=join("-",$cta1,$cta2); #esta funcion pega dos datos distintos
#para formar el numero de cuenta.

print "<hr> ";

&valida_entrada;
&valida_fecha;

open(EVALUA,"./introducir \"$num_cuenta\" \"$nombre\" \"$calle\"
\"$calle\" \"$colonia\" \"$cod_post\" \"$fecha_nac\" \" \"$lugar_nac\"
\"$telefono\" \"$mail\" \"$carrera\" |");
#en esta funcion se manda a llamar a un programa escrito en lenguaje C,
```

```
#el cual permite introducir datos a la B.D.

 while (<EVALUA>)
   {
    &error_base if (/ERROR/);
   }

print "<center>Datos Actualizados<br><p>";
print "<a href=\"/evaluacion/pagina1.html\"> regresar a la pagina anterior. </a>";
print "</center> <hr> ";

sub error_base
   {
      print "<H2><BLINK><CENTER>ERROR EN LA BASE DE DATOS </CENTER></BLINK></$
      print "<CENTER>NOTIFIQUELO A SU ADMINISTRADOR</CENTER>";
      exit(1);
   }

sub valida_entrada{
  if ( ($num_cuenta eq "") || ($nombre eq "") || ($mes eq "") || ($dia eq "")
     || ($ano eq "") || ($calle eq "") || ($colonia eq "") ||
     ($cod_postal eq "") || ($lugar_nac eq "") || ($carrera eq ""))
   {
    print " <h1> Te faltaron datos </h1><p> ";
    exit(1);
   }

   if (($cta1=~/[^0-9]/)||($cta2=~/[^0-9]/))
   {
      print ("Esta incorrecto el n&uacute;mero de cuenta ");
     exit(1);
   }

   if (($cod_postal=~/[^0-9]/))
   {
      print ("Esta incorrecto el codigo postal ");
      exit(1);
   }
}

sub valida_fecha{

#Arreglo para mantener los dias maximos validos para cada mes

%FECHAS=(1,31,2,28,3,31,4,30,5,31,6,30,7,31,8,31,9,30,10,31,11,30,12,31);

$dia=~s/\s+//g;
$mes=~s/\s+//g;
$ano=~s/\s+//g;

if (($dia ne "")&&($mes ne "")&&($ano ne ""))
{
      if (($dia eq "")||($mes eq "") ||($ano eq ""))
       {
```

```
            print ("Debe proporcionar la fecha completa");
            exit(1);}
      if (($dia=~/[^0-9]/)||($mes=~/[^0-9]/)||($ano=~/[^0-9]/))
      {
            print ("Las fechas deben consistir solo de n&uacute;meros");
            exit(1);
      }
      $dia=int($dia);$mes=int($mes);$ano=int($ano);
      if (($mes > 12)||($mes < 1))
      {
            print ("Puede proporcionar un mes entre 1 y 12");
            exit(1);
      }

      if (($dia < 1))
      {
            print ("El d&iacute;a debe ser mayor a 1");
            exit(1);
      }
#Febrero tiene 29 dias en los anios bisiestos;

      $temporal=$FECHAS{2};
      if (($ano %4 ==0) && ($ano %400 != 0) && ($mes == 2))
      {
            $FECHAS{2}++;
      }

      if ($dia > $FECHAS{$mes})
      {
print("Ese d&iacute;a no es v&aacute;lido para el mes que propo$
            exit(1);
      }
      $FECHAS{2}=$temporal;
      $fecha_nac=join("-",$mes,$dia,$ano);
 }
}
```

El código del programa en lenguaje C es el siguiente:

```
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <stdio.h>
#include <string.h>

/*Programa que permite insertar datos a B.D. ENEP en la tabla ALUMNOS*/

char num_cuenta[8];
char nombre[40];
char calle[60];
char colonia[20];
char cod_postal[5];
```

```
char resp[50];
char fecha_nac[10];
char lugar_nac[40];
char telefono[10];
char mail[20];
char carrera[5];

void inicia (void);
void lista(void);
void salva(void);
void fin(void);

main(int argc, char *argv[])
{
 strcpy(nom_cuenta,argv[1]);
 strcpy(nombre,argv[2]);
 strcpy(calle,argv[3]);
 strcpy(colonia,argv[4]);
 strcpy(cod_postal,argv[5]);
 strcpy(fecha_nac,argv[6]);
 strcpy(lugar_nac,argv[7]);
 strcpy(telefono,argv[8]);
 strcpy(mail,argv[9]);
 strcpy(carrera,argv[10]);
 strcpy(clave,"cliserv");
 strcpy(password,"TEMPO1");

inicia();
lista();
salva();
fin();
}/*fin del main*/

void inicia(void)
{
   LOGINREC *login;
   DBPROCESS *dbproc;
   DBCHAR nada[100];

   dbinit();
   login=dblogin();
   DBSETLUSER(login,clave);
 DBSETLPWD(login,password);

   dbproc=dbopen(login,"ENEP");
   if(dbproc!=NULL)
    {

       dbfcmd(dbproc,"begin transaction"); /*para que no haya inconsistencia*/
       if(dbsqlexec(dbproc)==FAIL)
        {
         printf("ERROR AL ENVIAR EL SQLEXEC 1");
         exit(1);
        }
```

```
    }
   else
    {
     printf("ERROR AL ACCESAR EL SERVIDOR DE LA BASE DE DATOS");
     exit(1);
    }
}/*fin de inicia*/

void ilista(void)
{
   LOGINREC *login;
   DBPROCESS *dbproc;
   DBCHAR nada[100];

   dbinit();
   login=dblogin();
   DBSETLUSER(login,clave);
   DBSETLPWD(login,password);

   dbproc=dbopen(login,"ENEP");
   if(dbproc!=NULL)
     {
       dbfcmd(dbproc,"select * from ALUMNOS");
       if(dbsqlexec(dbproc)==FAIL)
        {
          printf("ERROR AL ENVIAR EL SQLEXEC 1\n");
          exit(1);
        }
      dbresults(dbproc);
       dbbind(dbproc,8,NTBSTRINGBIND,0,nada);
       id=0;
       while(dbnextrow(dbproc)!=NO_MORE_ROWS)
       {
         id++;
       }
 dbexit();
    }
   else
    {
     printf("ERROR AL ACCESAR EL SERVIDOR DE LA BASE DE DATOS\n");
     exit(1);
    }
}/* FIN DE LA FUNCION */

void salva(void)
{
   LOGINREC *login;
   DBPROCESS *dbproc;
   dbinit();
   login=dblogin();
   DBSETLUSER(login,clave);
   DBSETLPWD(login,password);

   dbproc=dbopen(login,"ENEP");
```

```
if(dbproc!=NULL)
  {

     dbfcmd(dbproc,"insert into ALUMNOS values(\"%s\",\"%s\",\"%s\",\"%s\"
          ,\"%s\",\"%s\",\"%s\",\"%s\",\"%s\",\"%s\",\"%s\")"
          ,num_cuenta,nombre,calle,colonia,cod_postal,
          fecha_nac,lugar_nac,telefono,mail,carrera);

     if(dbsqlexec(dbproc)==FAIL)
      {
       printf("ERROR AL ENVIAR EL SQLEXEC 2");
       exit(1);
      }
     dbclose(dbproc);
  }
 else
  {
  printf("ERROR AL ACCESAR EL SERVIDOR DE LA BASE DE DATOS");
  exit(1);
  }

}/*fin de salva1*/

void fin(void)
{

  LOGINREC *login;
  DBPROCESS *dbproc;

  dbinit();
  login=dblogin();
  DBSETLUSER(login,clave);
  DBSETLPWD(login,password);

  dbproc=dbopen(login,"ENEP");
  if(dbproc!=NULL)
    {

      dbfcmd(dbproc,"commit");
      if(dbsqlexec(dbproc)==FAIL)
       {
        printf("ERROR AL ENVIAR EL SQLEXEC 8");
        exit(1);
       }
      dbclose(dbproc);
    }
   else
    {
    printf("ERROR AL ACCESAR EL SERVIDOR DE LA BASE DE DATOS");
    exit(1);
    }

 }/*fin de fin*/
```

La página para hacer consultas a la Base de Datos se muestra en la figura 7.2.

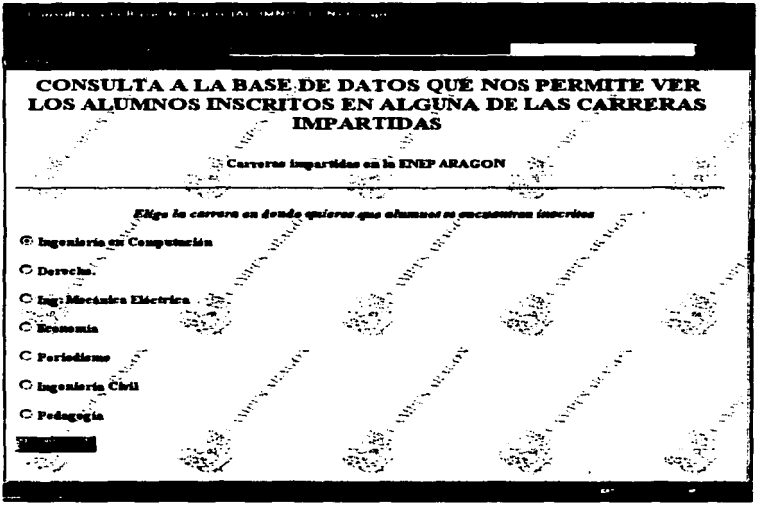

Figura 7.2 Página para realizar búsqueda por carrera.

Para realizar consultas a la información que se encuentra en la base de datos, se utiliza el script consulta.pl que está escrito en Perl el cual nos permite desplegar la búsqueda de todas las personas inscritas en alguna carrera en especial. Este script se auxilia de un programa escrito en lenguaje C y llamado consulta.c y este es quien realizara la petición a la Base de Datos.

El código del script en Perl es el siguiente:

```
#!/usr/bin/perl

#ESTE PROGRAMA DESPLIEGA LOS DATOS ALMACENADOS EN LA TABLA ALUMNOS
use CGI;
$query = new CGI;
$carrera = $query->param('carrera');

if (! $ENV{'SYBASE'}) { $ENV{'SYBASE'} = "/sybase/system_XI";}
if (! $ENV{'DSQUERY'}) { $ENV{'DSQUERY'} = "CSR_SYBASE";}
print "Content-type: text/html\n\n";

print <<END;
<HTML>
<HEAD>
<TITLE> Consulta</TITLE>
</HEAD>
<FORM>
<TT><H3><CENTER>DATOS DEL ALUMNO</CENTER></H3></TT> <P> <P> <HR> <P>
<P>
<P>
END

open(SYBASE,"./consulta \"$carrera\" |");
$cont=1;
 while (<SYBASE>)
   {
      &error_base if (/ERROR/);
      print "$_<P>\n";
      print "<hr><hr>\n";
   }
  print "<center><a href=\"/consulta.html\">Hacer otra
consulta</center>";

sub error_base
{
       print "<H2><BLINK><CENTER>ERROR EN LA BASE DE DATOS
</CENTER></BLINK></H2>";
        print "<CENTER>NOTIFIQUELO A SU ADMINISTRADOR</CENTER>";
        exit(1);
}

print "</FORM>";
print "</BODY>";
print "</HTML>";
```

El código escrito en Lenguaje C es el siguiente:

```
#include <sybfront.h>
#include <sybdb.h>
#include <syberror.h>
#include <stdio.h>
```

```
#include <string.h>

/*Este programa realiza busca a todas las personas que se encuentran en
una carrera solicitada y enviada por el script consulta.pl*/
char CUENTA[100][8];
char NOMBRE[100][40];
char busqueda[5];

void consulta(void);
int y=0;
int cont=0;
void despliega(void);

main(int argc, char *argv[])
{
  strcpy(busqueda,argv[1]);
  consulta(); /*funcion que realiza la busqueda en la tabla ALUMNOS*/
  despliega();/*funcion que despliega el resultado*/
}

void consulta(void)
{
     LOGINREC *login;
     DBPROCESS *dbproc;
     DBCHAR num_cuenta [8];
     DBCHAR nombre[40];

     dbinit();
     login=dblogin();
     DBSETLUSER(login,"cliserv");
     DBSETLPWD(login,"TEMPO1");
     dbproc=dbopen(login,"ALUMNOS");
      if(dbproc!=NULL)
        {
          dbcmd(dbproc,"select al_numcta, al_nombre  from ALUMNOS where
                al_carrera=\"%s\"",busqueda);
     if(dbsqlexec(dbproc)==FAIL)
        {
          printf("ERROR AL ENVIAR EL SQLEXEC");
          exit(1);
        }
          dbresults(dbproc);
          dbbind(dbproc,1,NTBSTRINGBIND,0,num_cuenta);
          dbbind(dbproc,2,NTBSTRINGBIND,0,nombre);
          y=0;
          while(dbnextrow(dbproc)!=NO_MORE_ROWS)
          {
            strcpy(CUENTA[y],num_cuenta);
            strcpy(NOMBRE[y],nombre);
            y++;
          }
         dbexit();
        }
      else
```

```
      printf("ERROR AL ACCESAR EL SERVIDOR DE LA BASE DE DATOS");
}/*fin de funcion main*/

void despliega(void)
{
     int x=0;
     for(x=0;x< y ;x++)
     {
       printf ("NUMERO DE CUENTA : %s\n",CUENTA[x]);
       printf ("NOMBRE : %s\n",NOMBRE[x]);
     }
}
```

La figura 7.3 muestra el resultado del script consulta.pl

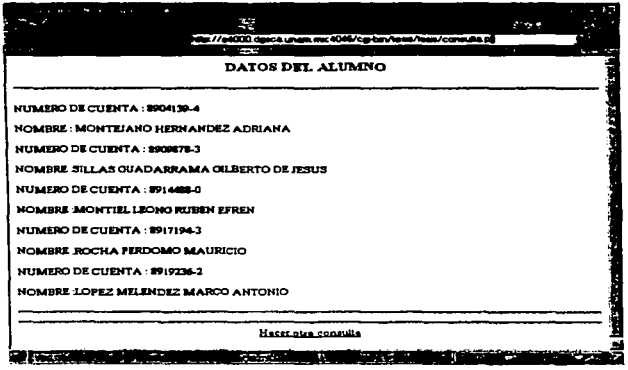

Figura 7.3 Resultado de la búsqueda.

Conclusiones

CONCLUSIONES

Una de las principales ventajas de está arquitectura es que aunque los proveedores de las bases de datos ofrecen herramientas propias de desarrollo, el poder de la arquitectura cliente/servidor radica en la variedad de aplicaciones que se hacen con una gran variedad de herramientas que se encuentran disponibles. Como se demostró utilizando el API en lenguaje C para trabajar con Sybase, Perl para la realización de CGI's y HTML como la aplicación cliente (front-end) que nos permite dar la presentación.

Otro objetivo que se demostró es la facilidad con que sistemas diferentes puedan interactuar entre sí para obtener un fin, en este caso el poner a disposición la información que se encuentra en una base de datos al mundo entero, sin tener que invertir mucha cantidad de dinero.

Se demostró como esta tecnología es la base de desarrollo de sistemas de Bases de Datos y tecnología de redes como se observa que es la base en los sitemas que ofrece Internet, en el caso particular del World Wide Web. Nos permite la implementación de servicios.

El World Wide Web es la parte de Internet que más usuarios nuevos atrae y esto trae como resultado el interés por tener un servidor de HTTP para poder proporcionar servicios. La instalación y configuración de este servidor es posible conociendo las directivas y tipos de archivos que se tienen que modificar. Pero el poner a disposición de todo el público información tiene que involucrar aspectos de seguridad por lo que se demostró algunos tipos de seguridad que ofrece el propio servidor

Por ultimo se explicó que es él termino CGI y como han venido a cambiar las páginas en el Web. Tipos de lenguajes que se utilizan para crear CGI's bajo ambiente UNIX. Aunque hay lenguajes más robustos como es Java para aplicaciones no tan complicadas el uso de lenguajes como HTML, Perl y C, dan como resultado la implantación de servicios para el Web.

Bibliografía

BIBLIOGRAFIA

- Creacion de Servidores de Base de Datos para Internet con CGI.
Jeff Rowe. Ed. Prentice Hall.

- Developing Client/Server Systems Using Sybase SQL Server System 10.
Sanjiv Purba. Ed. Wiley-QED Publication.

- CGI Programming with Perl 5 in a week.
Herrmann Erick. Ed. Sams net

- USING CGI.
Jeffry Dwigth and Michael Erwin. Ed. Que

- HTML & CGI Fundamentos de Programacion .
Ed Tittel, Mark Gaither, Sebastian Hassinger y Mike Erwin. Ed. Anaya Multimedia

- A Guide Developing to Client/Server SQL Applications.
Setrag Khoshafian, Arvola Chan, Anna Wong y Harry K. T. Wong. Ed. Morgan Kaufmann Publishers.

- How to Program CGI with Perl 5.0.
Stephen Lines. Ed. Ziff-Davis Press.

- Managing Internet Information Services.
Cricket Liu, Jerry Peex, Russ Jones, Bryan Buus y Adrian Nye. Ed. O'Really & Associates, Inc.

- HTML, JAVA, CGI, VRML, SGML, Web Publishing Unleashed.
William Robert Stanck. Ed. Sams Net.

- Programming in Perl.
Larry Wall. Ed. O'Really & Associates Inc.

- Learning Perl.
Randal L. Schwartz & Tom Christiansen Foreword y Larry Wall. Ed. O'Really & Associates Inc.

INTERNET

http://hoohoo.ncsa.uiuc.edu/cgi/

http://blackcat.brynumarr.edu/~nswobuda/prog-html.html

http://www.espanet.com/pcar/glosario.html

http://www.pntic.see.mec.es/pntic/ayudas/manual.htm

http://www.sti.nasa.gov/thesaurus/R/word13048.html

http://www.adp.unc.edu/info/syperl.html

http://www.artech.com.uy/white/spanish/csarti.htm

http://www.artech.com.uy/white/indice_art.html

http://www.tel.uva.es/~celgom/seguridad/seguridad.html

http://spin.com.mx/~mdiharce/ibero1.html

http://www.bib.udec.cl/manual/manual.html#conceptos

http://info.cern.ch/hypertext/WWW/Addressing/URL/URL_Overview.html

ftp://ds.internic.net/rfc/rfc1630.txt

http://www.ncsa.uiuc.edu/demoweb/url-primer.html

http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/Docs/web-index.html#Tutorials

http://luna.bearnet.com/perlman

http://ds.internic.net

http://www.w3.org

http://www.web3.es

http://www.gi.net/NET/PM-1994/94-08/94-08-09/0007.html

http://www.eit.com/projects/s-http/index.html

http://www.bio.com.ac.uk/cgi-lib/