UNIVERSIDAD LASALLISTA BENAVENTE

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

Con estudios incorporados a la
Universidad Nacional Autónoma de México
CLAVE: 8793-16

"ACCESO A BASES DE DATOS EN PÁGINAS WEB CON PHP Y MYSQL"

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

LUIS FERNANDO ZAMORANO VÁZQUEZ

Asesor: ING. MIGUEL ÁNGEL JAMAICA ARREGUÍN

Celaya, Gto. **Febrero de 2003**

A

A mis padres
y a mi esposa
por su gran apoyo.

Contenido

INTRODUCIÓN

Actualmente el flujo de información es cada vez más rápido y tanto las instituciones como las organizaciones necesitan información rápida, actual y en diferentes lugares. Para esto Internet ha sido de gran ayuda, y el desarrollo de sitios web dinámicos de vital importancia en el área del desarrollo de software. Una potente herramienta para el desarrollo de sitios web dinámicos es PHP.

La tecnología PHP está en constante crecimiento ya que es una tecnología de código abierto. Ésta a tenido una tendencia de uso a la alza ya que es simple, poderosa, y además compatible con la mayoría de las plataformas actuales.

El fuerte de PHP es el acceso a base de datos, para el cual tiene como manejador predeterminado a MySQL con el cual funciona excelente. Además también es compatible con un gran número de manejadores de bases de datos usados en la actualidad.

El objetivo de el siguiente trabajo es mostrar en un principio un poco la historia y los servicios básicos de Internet. Después la sintaxis básica de el lenguaje HTML para la creación de paginas web. Y ya entrando un poco en materia las bases del funcionamiento de PHP, su sintaxis y el paso de variables de HTML a PHP. En cuanto al las bases de datos: qué son, que tipos existen, y por ultimo las reglas del lenguaje SQL y la sintaxis de MySQL. Para terminar se explicará e ilustrará con un caso practico, como realizar una conexión con PHP a una base de datos MySQL.

F

CAPITULO I

ANTECEDENTES

1.1 HISTORIA DE LA INTERNET

Después de la Segunda Guerra Mundial, los países se subdividieron en dos bloques: por un lado, Estados Unidos y sus aliados, en América y Europa, y por el otro lado, la Unión de Repúblicas Soviéticas Socialistas (URSS) y sus aliados. Comenzando la denominada "Guerra Fría", esto generó un clima de tensión y estrategia militar.

Con un temor latente por un ataque de grandes magnitudes, en 1962 Paul Baran, de Rand Corporation, presenta un informe titulado "*On distributed communications network*", donde se cuestiona sobre la manera de crear un sistema de comunicaciones que permitiera a la milicia de U.S.A. sobreponerse a cualquier ataque.

Dicho sistema estaría basado en una extensa red de muchas ramificaciones (universidades, instituciones de investigación y desarrollo y la milicia de los países aliados), comunicado mediante paquetes conmutados de información. Estas ideas inspiraron a la defensa norteamericana para crear la red militar ARPANET, la cual quedó instalada con cuatro nodos iniciales (el primero de la Universidad del Sur de California en Los Angeles), operando bajo el protocolo Network Control Protocol N.C.P.

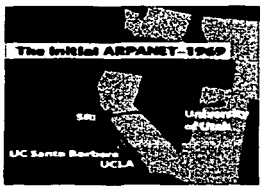

Figura 1. Primeros nodos de la red ARPANET[1]

[1] *Imagen tomada de:* ***www.sri.com/about/timeline/ timeline2.html.***

Entre septiembre y diciembre de 1969; para abril de 1971, se hallaban conectadas 23 computadoras denominadas *anfitriones (hosts)* en 15 nodos ubicados en la Universidad de California en Los Angeles, el Instituto de Investigaciones de la Universidad de Stanford, la Universidad de California en Santa Bárbara, la Universidad de UTA en SALT Lake City, BBN Communications, el CDS, la Universidad de Harvard, los laboratorios de Lincoln, la Universidad de Illinois en Urbana Champagne, la Case Western Reserve University, la Universidad Carnegie-Mellon y el Centro de Investigaciones Ames de la NASA.

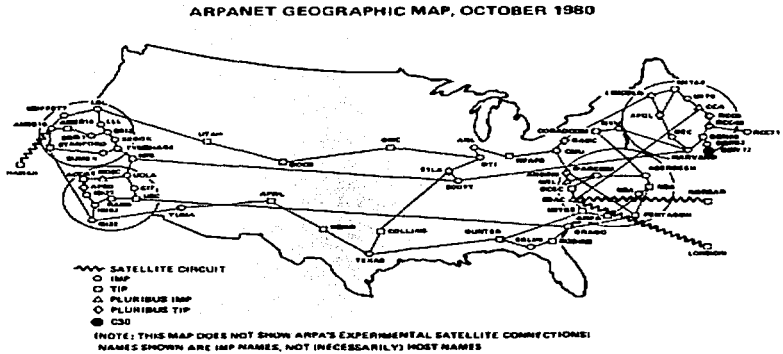

Figura 2. Mapa geográfico de la red ARPANET en octubre de 1980[2]

El crecimiento de la red ARPANET fue lento al principio, de tal manera que para 1984 había 1024 anfitriones conectados; sin embargo, a partir de la segunda mitad de la década de los 80, se nota un explosivo crecimiento que la lleva a terminar el año 1989 con aproximadamente 160 000 y en menos de 3 años, en 1992, alcanzar la cifra de 1000 000 de computadoras y más de 8000 redes conectadas a la red.

En 1973 se realiza la primera conexión internacional de ARPANET con el nodo de University College of London en Inglaterra.

[2] *Imagen tomada de* ***http://www.mappa.mundi.net/maps/maps_001/.***

En 1974, Vint Cerf y Bob Kahn publican las especificaciones de un nuevo protocolo más abierto y estándar en su "A Protocol for Packet Network Intercommunication". Este evento marca el inicio del que se utiliza en la actualidad: Transmisión Control Protocol (TCP).

Figura 3. Vint Cerf pionero del protocolo TCP[3]

Figura 4. Bob Kahn pionero del protocolo TCP[4]

1.1.1 Historia del World Wide Web

El servicio gráfico de Internet conocido como *World Wide Web* o WWW, es realmente una herramienta nueva comparada con la Internet. Antes de la WWW, los investigadores de las universidades utilizaban aplicaciones de Internet como el correo electrónico en sus comunicaciones, y el Telnet para acceder a computadoras remotas, también creaban

[3] *Imagen tomada de:* ***http://www.comsoc.org/confs/IPv6/.***
[4] *Imagen tomada de:* ***http://www.cnri.reston.va.us/bios/kahn.html.***

directorios para el almacenamiento de archivos que se podían compartir y buscar mediante buscadores antiguos como Gopher, Archie y Verónica, o Finger, para conocer datos sobre un usuario conectado a la red; todo esto a través de la clásica pantalla negra de una terminal UNIX o mediante una PC que emulaba la terminal UNIX con un programa de comunicaciones en modo texto, usando en protocolo TCP/IP.

El WWW es un sistema distribuido de información basado en *hipertexto* e *hipermedia.* Fue desarrollado en 1990 por un grupo de investigadores bajo la dirección de Tim Berners-Lee en el *Laboratorio Europeo de Física en Partículas CERN,* ubicado en Suiza. En ellos definieron los conceptos HTTP, HTML y URL, que son los elementos base para construir, localizar y acceder a la paginas del Web en cualquier nodo o red conectada a Internet.
En octubre de 1990 se define el nombre del nuevo servicio de información compartida, quedando como se conoce ahora: World Wide Web; comienzan los primeros desarrollos de un *ojeador/editor* que permite crear ligas de *hipertexto* en línea.

Figura 5. Tim Berners-Lee, el padre de la Web.[5]

Tan rápido cundió la necesidad de compartir información en forma de *hipertexto* e *hipermedia,* que para el mes de octubre de 1993 ya existían 200 servidores de Word Wide

[5] *Imagen tomada de:* ***www.samogden.com/ tim_berners-lee_photo3.html.***

Web utilizando el protocolo HTTP, que significan HyperText Transport Protocol o Protocolo de Transporte de Hipertexto.

Diseñado como una herramienta para facilitar la transmisión de documentos compuestos de texto, gráficos y sonidos, el lenguaje HTML (HiperText Markup Language) es el estándar para el diseño y creación de las paginas del Web. URL es el nombre del localizador de los recursos de Internet Uniform Resource Locator o Localizador Uniforme de Recursos.

Para el 2001 existen más de 100,000,000 sitios web, aunque esta cifra esta creciendo considerablemente como se muestra en la figura 6.

Figura 6. Numero de sitios web hasta el 2001 [6]

1.1.2 Internet en México

En México, los primeros intentos de conectar computadoras en red se realizaron en las universidades; una de ellas, constituida por computadoras Apple II y Apple II+, del Campus Monterrey del Instituto Tecnológico y de Estudios Superiores de Monterrey

[6] *Imagen tomada de:* ***http://www.kioscosnet.com/empresa.html.***

(ITESM), primera institución que se incorpora a la super–red en 1988, con un nodo hacia la Universidad de Texas en San Antonio, en Estados Unidos.

La Universidad Nacional Autónoma de México (UNAM) se conectaba, permanentemente, mediante una línea privada de 9600 bps al nodo del ITESM. El ITESM renueva sus equipos y conexiones entre su nodo y la Universidad de Texas San Antonio, en noviembre de 1988, y se conecta a Internet. Quedando así como la primera institución latinoamericana que se enlaza a la red.

1.2 REDES LOCALES

Una red local es un sistema de interconexión entre computadoras que permite compartir recursos de información. Para ello, es necesario contar, además de computadoras, con tarjetas de red, cables de conexión, dispositivos periféricos y software apropiado.

Según su cobertura, se pueden distinguir 3 tipos de redes:

- **Red de Área Local:** Si se conectan computadoras dentro de un mismo edificio con una distancia entre sí no mayor a los 100 metros, se denomina LAN (Local Area Network).
- **Red de Área Extendida:** Si se interconectan redes instaladas en edificios diferentes, se denomina una WAN (Wide Area Network).
- **Red de Área Metropolitana:** Si se interconectan equipos distribuidos en distancias no superiores al ámbito urbano, se denomina una MAN (Metropolitan Area Network).

1.2.1 Ventajas de las redes locales

- Posibilidad de compartir periféricos costosos, como impresoras, modem, fax, etc.
- Posibilidad de compartir grandes cantidades de información a través de distintos programas, bases de datos, etc., de manera que sea más fácil su uso y actualización.
- Reduce, e incluso elimina, la duplicidad de trabajos.
- Permite utilizar el correo electrónico para enviar o recibir mensajes de diferentes usuarios de la misma red e incluso de redes diferentes.
- Reemplaza o complementa mini-computadores de forma eficiente y con un coste bastante más reducido.
- Establece enlaces con mainframes; de esta forma, una computadora de gran potencia actúa como servidor, haciendo que los recursos disponibles estén accesibles para cada una de las computadoras personales conectadas.
- Permite mejorar la seguridad y control de la información que se utiliza admitiendo la entrada de terminados usuarios, accediendo únicamente a cierta información o impidiendo la modificación de diversos datos.

Para poder interconectar las computadoras y compartir periféricos, se necesita configurar uno o más computadores como servidores de la red, el resto de los computadores se denominan estaciones de trabajo, y desde ellos se facilita a los usuarios el acceso a los periféricos de la red.

Una red pequeña puede tener un servidor de archivos y varias estaciones de trabajo, pero una red puede llegar a tener varios servidores de archivos, de impresión, de comunicaciones para escoger el modelo de red adecuado.

1.2.2 Protocolo TCP/IP

El nombre TCP/IP proviene de dos de los protocolos más importantes de los inicios de Internet: el Transmisión Control Protocol (TCP) y el Internet Protocol (IP).

La principal virtud de TCP/IP es que está diseñado para enlazar computadoras de manera indistinta al tipo de éstas, incluyendo PC, minicomputadoras y mainframes, que ejecuten sistemas operativos distintos sobre redes de área local y redes de área extensa, y por tanto permite la conexión de equipos distantes geográficamente.

El protocolo TCP/IP transfiere datos en paquetes. Cada paquete comienza con una cabecera que contiene información de control seguida de los datos. El Internet Protocol (IP) permite a las aplicaciones ejecutarse de forma transparente sobre las redes interconectadas. De esta forma, las aplicaciones no necesitan conocer qué hardware está siendo utilizado en la red, por lo que la misma aplicación puede ejecutarse en una topología Ethernet, Token-Ring o X.25.

Todos los ordenadores en Internet utilizan el protocolo TCP/IP, y gracias a ello se consigue eliminar la barrera de la heterogeneidad de los ordenadores y resolver los problemas de direccionamiento.

1.3 SERVICIOS DE INTERNET

Sobre la base la infraestructura de transporte de datos que proporciona el protocolo TCP/IP, se han construido otros protocolos más específicos que permiten, por ejemplo, enviar correo electrónico (SMTP), establecer conexiones y ejecutar comandos en máquinas remotas (TELNET), acceder a foros de discusión o news (NNTP), transmitir ficheros (FTP), conectarse con un servidor web (HTTP), etc. A estas capacidades de Internet se les llama servicios. A continuación se revisan los más conocidos.

1.3.1 Correo electrónico

El correo electrónico o *e-mail* permite mantener correspondencia con usuarios en cualquier parte del mundo.

Respecto al correo tradicional, tiene la ventaja de que es mucho más rápido y sencillo de utilizar: es una manera muy fácil de enviar o recibir mensajes y archivos.

El correo electrónico tiene también ventajas económicas: es más barato que los servicios comerciales de correos y carece de sobrecargas por larga distancia, siendo a su vez rápido y efectivo en el costo.

El protocolo que se utiliza para el correo es el llamado SMTP *(Simple Mail Transfer Protocol).*

1.3.2 Telnet

Mediante Telnet es posible conectarse a un ordenador remoto en el que se tiene una cuenta de usuario o simplemente que está abierto a cualquier usuario. Tradicionalmente, Telnet se ha utilizado para acceder a servicios de bases de datos y catálogos de bibliotecas. Telnet abre la posibilidad de conectarse a una cuenta remota gracias a Internet.

El servicio Telnet hace que se pueda estar conectado a un servidor remoto mediante una consola Unix, de igual manera, que si la conexión se realizara en el propio ordenador. Todo lo que se escribe desde un teclado es redireccionado al ordenador remoto. De igual manera, todo lo que el ordenador remoto devuelva como respuesta es direccionado al monitor del usuario. No importa la distancia que haya entre ambos.

A diferencia del *e-mail*, Telnet establece una conexión permanente y síncrona entre los ordenadores cliente y servidor, conexión que permanece hasta que explícitamente es cortada por una de las dos partes.

1.3.3 FTP

El servicio FTP o Protocolo de Transferencia de Archivos es una parte importante de Internet. FTP permite transferir bidireccionalmente cualquier tipo de archivos con cualquiera de los miles de ordenadores remotos que tengan un servidor FTP.
Al igual que Telnet, FTP establece conexiones síncronas y permanentes. Para utilizar el servicio FTP suele ser necesario proporcionar un nombre de usuario y un password.

1.3.4 WWW

Es el servicio grafico más usado de la Internet, también conocido como *World Wide Web,* o simplemente *Web, que* utiliza el protocolo HTTP (Hypertext Transfer Protocol) y proporciona el acceso a páginas web escritas en el lenguaje HTML (HyperText Markup Language), el cual utiliza el hipertexto y la hipermedia.

Las páginas web no son más que documentos que contienen información util para sí mismas y también contiene frases, palabras y objetos que vinculan a otros documentos web.

1.4 EL PROTOCOLO HTTP

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes

Web y los servidores http, propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web que puede ser un documento HTML, fichero multimedia o aplicación CGI, es conocido por su URL.

Los recursos u objetos que actúan como entrada o salida de un comando HTTP están clasificados por su descripción MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido. La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permitirá que el receptor trate adecuadamente los datos.

1.4.1 Etapas de una transacción HTTP

Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario accede a una URL, seleccionando un enlace de un documento HTML o introduciéndola directamente en el campo *location* del cliente web.

2. El cliente web decodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
4. Se realiza la petición. Para ello, se envía el comando necesario (get, post, head,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor.
5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
6. Se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertadas cuatro para las imágenes. cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML.

En la actualidad, se ha mejorado este procedimiento, permitiendo que una misma conexión se mantenga activa durante un cierto periodo de tiempo, de forma que sea utilizada en sucesivas transacciones. Este mecanismo, denominado HTTP Keep Alive, es empleado por la mayoría de los clientes y servidores modernos.

1.4.2 Comandos del protocolo

Los comandos de HTTP representan las diferentes operaciones que se pueden solicitar a un servidor HTTP. El formato general de un comando es:

Nombre del comando	Objeto sobre el que se aplica	Versión de HTTP utilizada

Cada comando actúa sobre un objeto del servidor, normalmente un fichero o aplicación, que se toma de la URL de activación. La última parte de esta URL, que representa la dirección de un objeto dentro de un servidor HTTP, es el parámetro sobre el que se aplica el comando. Se compone de una serie de nombres de directorios y ficheros, además de parámetros opcionales para las aplicaciones CGI

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

Get

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

Head

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

Post

Sirve para enviar información al servidor, por ejemplo, los datos contenidos en un formulario. El servidor pasará esta información a un proceso encargado de su tratamiento.

La operación que se realiza con la información proporcionada depende de la URL utilizada. Se utiliza, sobretodo, en los formularios.

Un cliente Web selecciona automáticamente los comandos HTTP necesarios para recoger la información requerida por el usuario. Así, ante la activación de un enlace, siempre se ejecuta una operación GET para recoger el documento correspondiente.

El envío del contenido de un formulario utiliza GET o POST, en función del atributo de <FORM METHOD="...">. Además, si el cliente Web tiene un caché de páginas recientemente visitadas, puede utilizar HEAD para comprobar la última fecha de modificación de un fichero, antes de traer una nueva copia del mismo.
Posteriormente se han definido algunos comandos adicionales, que sólo están disponibles en determinadas versiones de servidores HTTP, con motivos eminentemente experimentales. La versión 1.1 de HTTP, recoge éstas y otras novedades, que se pueden utilizar, por ejemplo, para editar las páginas de un servidor Web trabajando en remoto.

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

Delete elimina el documento especificado del servidor.

Link crea una relación entre documentos.

Unlink elimina una relación existente entre documentos del servidor.

Las cabeceras

Son un conjunto de variables que se incluyen en los mensajes HTTP, para modificar su comportamiento o incluir información de interés. En función de su nombre, pueden

aparecer en los requerimientos de un cliente, en las respuestas del servidor o en ambos tipos de mensajes. El formato general de una cabecera es:

Nombre de la variable : Cadena ASCII con su valor

1.4.3 Cookies

Las cookies son pequeños ficheros de texto que el servidor web solicita almacenar en el disco duro del cliente, para solucionar una de las principales deficiencias del protocolo HTML, la falta de información de estado entre dos transacciones. Los servidores web las utilizan para almacenar y recuperar información. Fueron introducidas por Netscape, y ahora han sido estandarizadas en el RFC 2109.

Por lo general, las *cookies* son usadas para que el servidor web pueda recordar cierta información del cliente.

1.4.4.1 *Uso de las Cookies*

Una *cookie* es simplemente una serie de líneas de texto, con pares variable/valor.

Existe un conjunto predefinido de nombres de variable, necesarias para el correcto funcionamiento de las *cookies*, pero se pueden crear nuevas variables para cubrir las necesidades de una aplicación concreta.

Cuando se accede a una URL que verifica el par dominio/path registrado, el cliente enviará automáticamente la información de los diferentes campos de la *cookie* con la cabecera HTTP Cookie:

elemento	Contenido
nombre	Nombre de la *cookie*
valor	Valor asociado a la *cookie*(se envía empleado la codificación URL
fecha de expiración	Fecha de expiración de la *cookie*
path	Subconjunto de URL para los que la *cookie* es válida
dominio	Rango de dominios para los que la *cookie* es valida
segura	Indica si la *cookie* se debe transmitir exclusivamente sobre conexio seguras HTTPS

Figura 6. Valores en una Cookie [7]

[7] *Figura realizada por el autor.*

CAPITULO II

INTRODUCCION AL LENGUAJE HTML

2.1 ¿COMO ES EL CÓDIGO HTML?

El código HTML se basa en etiquetas (tags), las cuales nos indican un formato o la inserción de un objeto en la página web, estas etiquetas se representan entre paréntesis angulares "<etiqueta>"; un ejemplo puede ser:

```
<B>EL TEXTO ESTA EN NEGRITAS </B>
```

La etiqueta indica que el texto siguiente debe ir con negritas hasta que llega la etiqueta , la diagonal indica el fin del formato.

2.2 ESTRUCTURA BÁSICA DE UNA PAGINA HTML

Una página web tiene dos partes principales, una es el encabezado y otra es el cuerpo; en el encabezado se muestra información de la página, como puede ser el título, y en el cuerpo se muestra el contenido de la página. Un ejemplo básico podría ser:

```
<html>
<head>
<title> Ejemplo basico de una Pagina Web </title>
</head>
<body>
Este es un ejemplo muy simple de una Pagina HTML
</body>
</html>
```

Donde <html> y </html> indican al navegador que es una página HTML. La etiqueta <head> delimita el encabezado, donde <title> indica el título de la página. Y por ultimo, <body> indica el cuerpo de la misma. Esto se vería de la siguiente manera:

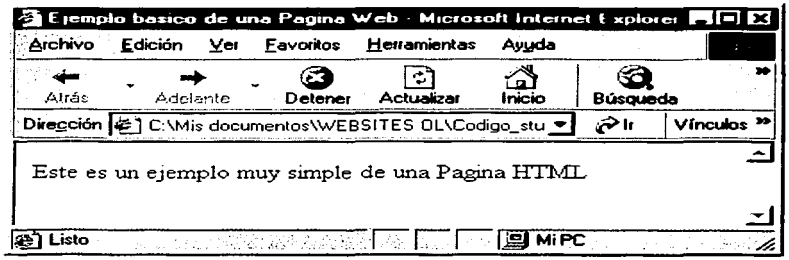

Figura 8. Ejemplo de página web básica [8]

2.3 FORMATO DE TEXTO

A continuación se mostrarán algunas etiquetas para dar formato al texto:

Para poner el texto en **negritas**:

```
<B> Este texto aparecerá en negritas </B>
```

Para hacerlo en *cursivas*:

```
<I> Este texto aparecerá en cursivas </I>
```

En subrayado:

```
<U> Este texto aparecerá subrayado </U>
```

[8] *Figura realizada por el autor.*

2.4 FORMATO EN PÁRRAFOS

Las siguientes etiquetas sirven para generar formato en párrafos;

<P> que genera un párrafo,

<HR> genera una línea divisoria,

 genera un salto de línea y

<H1></H1> hasta <H6></H6> generan encabezados.

Un ejemplo utilizando formatos de texto, párrafos y encabezados sería:

```
<html>
<head><title> Formato de  texto, parrafos y encabezados </title></head>
<body>
<HR>
Negritas:<br>
<B> Este texto aparecerá en negritas </B><BR>
Para hacerlo en cursivas:<BR>
<I> Este texto aparecerá en cursivas </I><BR>
En subrayado:<BR>
<U> Este texto aparecerá subrayado </U><BR><HR>
Encabezados: <BR>
<H1>Encabezado 1</H1>
<H2>Encabezado 2</H2>
<H3>Encabezado 3</H3>
<H4>Encabezado 4</H4>
<H5>Encabezado 5</H5>
<H6>Encabezado 6</H6><BR><HR><P> generando un parrafo...
</body></html>
```

Esto se vería así:

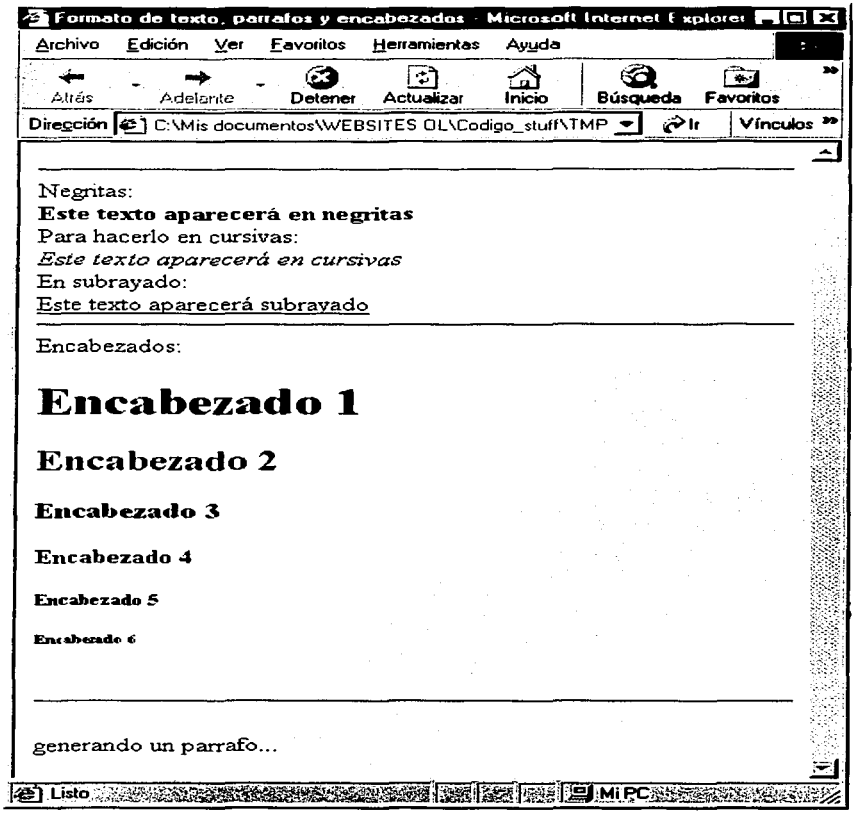

Figura 9. Ejemplo utilizando formatos de texto, párrafos y encabezados [9]

[9] *Figura realizada por el autor.*

2.5 LISTAS

En cuanto a enumerar textos o crear listas, las etiquetas serían:

Para crear una lista enumerada;

```
<OL>
<LI>Primer elemento
<LI>Segundo elemento
etc...
</OL>
```

Una lista de definiciones;

```
<DL>
<DT>Primer término
<DD>Primera definición
<DT>Segundo término
<DD>Segunda definición
etc...
</DL>
```

Una lista con viñetas;

```
<UL>
<LI>Primera viñeta
<LI>Segunda viñeta
etc...
</UL>
```

Un ejemplo de lo anterior es:

```
<html>
<head>
<title> Formato de  Listas </title>
</head>
<body>
<OL>
<LI>Primer elemento
<LI>Segundo elemento
</OL>
Una lista de definiciones;
<DL>
<DT>Primer termino
<DD>Primera definición
<DT>Segundo termino
<DD>Segunda definición
</DL>
Una lista con viñetas;
<UL><LI>Primera viñeta <LI>Segunda viñeta</UL>
</body>
</html>
```

El resultado del código anterior se muestra en la imagen de la siguiente página:

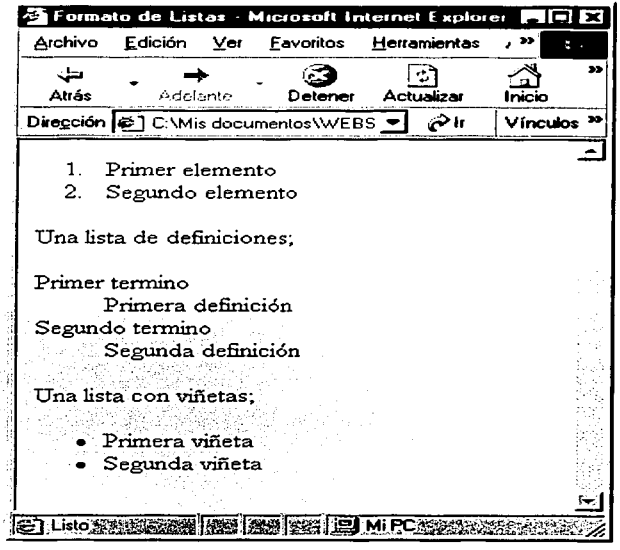

Figura 10. Ejemplo de listas [10]

2.6 VÍNCULOS

Puede haber dos tipos de vínculos (links): internos, que mandan llamar alguna parte de la misma página, y externos, que mandan llamar otra página web.

En cuanto a los vínculos externos, pueden vincular no sólo a páginas web, sino también a otros servicios de Internet. A continuación se muestran los diferentes tipos de vínculos externos.

Vínculo a página externa;

```
<A HREF="URL de la pagina">Texto de vinculo</A>
```

[10] *Figura realizada por el autor.*

A correo electrónico;

```
<A HREF="mailto:dirección de correo electrónico">Texto de vinculo</A>
```

A un FTP;

```
<A HREF=ftp://host/directorio/nombre del archivo>Texto de vinculo</A>
```

Gopher;

```
<A HREF="gopher://host/">Texto de vinculo</A>
```

Usenet;

```
<A HREF=news:nombre_de_grupo_de_noticias>Texto de vinculo</A>
```

Telnet;

```
<A HREF=telnet://host>Texto del vinculo</A>
```

2.7 IMÁGENES

Para insertar una imagen en la página web se utiliza;

```
<IMG SRC="nombre_del_archivo" ALIGN="posición" ALT="texto alternativo">
```

Donde los tipos de archivo comunes son gif, jpg o bmp. El parámetro ALIGN indica la posición de la imagen, puede ser TOP (superior), MIDDLE (en medio), BOTTON (inferior); en cuanto a ALT, es el comentario que sale cuando el cliente sitúa el ratón sobre la imagen. Tanto ALIGN como ALT son opcionales.

2.8 TABLAS

En HTML se pueden crear tablas para organizar la información. A continuación se muestran las etiquetas para generarlas.

<TABLE> y </TABLE> delimita la tabla.

<TR> y </TR> genera una fila.

<TH> y </TH> genera un encabezado de columna.

<TD> y </TD> genera una celda en la fila.

<CAPTION> y </CAPTION> genera texto de pie de tabla.

Una tabla básica sería:

```
<HTML>
<HEAD><TITLE>Tabla básica</TITLE></HEAD>
<BODY>
<TABLE border="1">
<TR>
<TH>COLUMNA UNO</TH><TH>COLUMNA DOS</TH>
</TR>
<TR>
<TD>fila uno celda uno</TD><TD>fila uno celda dos</TD>
</TR>
<TR>
<TD>fila dos celda uno</TD><TD>fila dos celda dos</TD>
</TR>
</TABLE></BODY></HTML>
```

Esto se vería así:

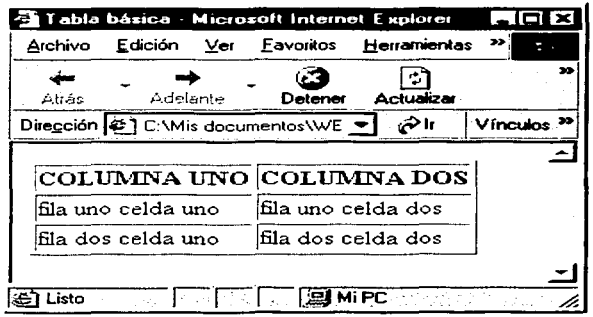

Figura 11. Ejemplo básico de una tabla [11]

2.8.1 Alineación horizontal y vertical

Las celdas se pueden alinear tanto horizontal como verticalmente con los siguientes parámetros dentro de las etiquetas.

En cuanto a la alineación horizontal se introduce el parámetro ALIGN con valores LEFT (izquierda), CENTER (al centro) y RIGHT (a la derecha).

En lo que respecta a la alineación, vertical se introduce el parámetro VALIGN con valores TOP (arriba), MIDDLE (en medio), BOTTON (abajo).
Un ejemplo podría ser:

```
<TD ALIGN=LEFT VALIGN=TOP>texto</TD>
```

Esto va a posicionar el contenido de la celda arriba y a la izquierda.

[11] *Figura realizada por el autor.*

```
<TH ALIGN=CENTER VALIGN=BOTTON>titulo de columna</TD>
```

Esto sitúa el titulo de la columna al centro y hacia debajo de la celda.

2.8.2 Celdas que abarcan columnas y filas

Se pueden generar celdas que abarquen varias columnas y varia filas, usando dos parámetros. Uno es COLSPAN, el cual indica el número de columnas que abarcará la celda.

El otro es ROWSPAN el cual indica el número de filas que abarcará la celda.

Unos ejemplos serían:

```
<TD COLSPAN="numero de columnas"></TD> y
<TD ROWSPAN="numero de filas"></TD>
```

Esta sería la estructura de una tabla usando estos atributos:

```
<HTML><HEAD><TITLE>Tabla 2</TITLE>
</HEAD>
<BODY>
<HTML border=1>
<TABLE>
<CAPTION ALIGN=BOTTON>Ejemplo de tablas 2</CAPTION>
<TR><TH>***</TH><TH COLSPAN=3>Abarca 3 columnas</TH></TR>
<TR><TD ROWSPAN=2>Abarca dos filas</TD><TD>1</TD><TD>2</TD><TD>3</TD></TR>
<TR><TD>4</TD><TD>5</TD><TD>6</TD></TR>
</TABLE>
</HTML>
```

Lo cual tiene una salida así.

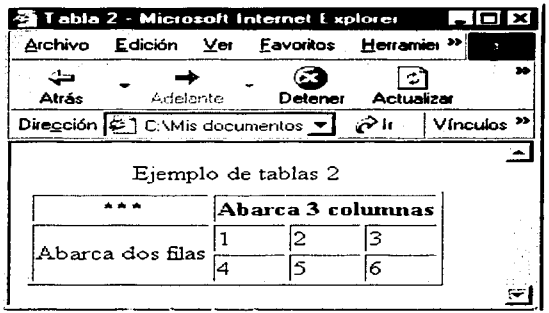

Figura 12. Ejemplo de tabla con celdas que abarcan varias columnas y filas [12]

2.9 FORMULARIOS

Los formularios sirven para capturar información del cliente mediante cajas de texto, casillas de verificación, listas desplegables (por mencionar algunas) y enviar esta información a un correo electrónico o a una aplicación que la procese.

Empezamos por generar el formulario con la etiqueta <FORM>, la cual tiene dos atributos. Uno es ACTION, que le dice al navegador a donde enviar los datos del formulario. El segundo es METHOD, el cual indica la forma de envío, la cual puede ser GET o POST.

Este es el formato general:

```
<FORM ACTION="url" METHOD="metodo"></FORM>
```

[12] *Figura realizada por el autor.*

Teniendo la forma, ahora se necesitará lo que reciba los datos y el botón que dé la orden de mandar la información al destino ya indicado con el parámetro ACTION.

2.9.1 Botones

Existen dos tipos de botones para la forma: el botón de envío "SUBMIT", que es el que da la orden de enviar la información a la dirección indicada en el atributo ACTION de la etiqueta <FORM>, y el de reinicio "RESET", que no hace más que borrar o poner en *default* la información del formulario. La sintaxis básica de los botones es la siguiente:

```
<INPUT TYPE=SUBMIT VALUE="Enviar">
<INPUT TYPE=RESET  VALUE="Borrar">
```

La mayoría de los elementos del formulario son variaciones de la etiqueta <INPUT> y se colocan dentro de las etiquetas <FORM> y </FORM>. En el ejemplo se puede ver en la primera línea un botón de envío, el cual mediante el atributo TYPE con valor SUBMIT se indica que es un botón de envío. En la segunda línea se genera un botón de reinicio, de igual manera con el atributo TYPE y el valor RESET. Tanto el botón de envío como el de reinicio contienen un atributo llamado VALUE que no es más que el texto que tendrá el botón.

2.9.2 Cajas de texto

Para entradas de texto de una sola línea, se pueden usar las cajas de texto, estas son las más usadas en los formularios. Un ejemplo básico sería:

```
<INPUT TYPE=TEXT NAME="nombre del campo">
```

Donde el atributo TYPE con valor "TEXT" indica que es una caja de texto, NAME le da un nombre único a la caja en todo el formulario. La caja de texto puede tener los siguientes atributos:

VALUE	rellena con un texto predeterminado la caja.
SIZE	determina la longitud de la caja tomando como medida caracteres.
MAXLENGTH	Restringe la longitud de entrada de caracteres.

Un ejemplo sería:

```
<INPUT TYPE=TEXT NAME="texto1" VALUE="escribe aquí" SIZE=10  MAXLENGTH=5 >
```

Donde el nombre de la caja es "texto1", el texto predeterminado que rellena la caja es "escribe aquí", el tamaño de la caja es de 10 caracteres y por último soporta la entrada de como máximo 5 caracteres.

2.9.3 Áreas de texto

El área de texto es similar a una caja pero con varias líneas o renglones. La sintaxis es la siguiente:

```
<TEXTAREA NAME="Area1"  ROWS=5 COLS=50 WRAP></TEXTAREA>
```

Donde NAME es el nombre del campo, ROWS indica las filas que contendrá el área de texto, COLS las columnas tomando como medida caracteres y WRAP determina el ajuste automático del texto en líneas cuando éste se aproxime al borde derecho.

Un ejemplo de lo que hemos visto hasta ahora de formularios sería:

```
<HTML>
<HEAD>
<TITLE>Ejemplo de Formularios con cajas y áreas de texto</TITLE>
<HEAD>
<BODY>
<FORM  ACTION="envia.php" METHOD="GET" >
Primera caja: <INPUT TYPE=TEXT NAME="caja1">
<P>
Segunda caja: <INPUT TYPE=TEXT NAME="caja2">
<P>Area de Texto:
<TEXTAREA NAME="Area1" ROWS=5 COLS=10 WRAP>
</TEXTAREA><P>
<INPUT TYPE=SUBMIT VALUE="Enviar">
<INPUT TYPE=RESET  VALUE="Borrar">
</FORM>
</BODY>
</HTML>
```

Esto generará lo que se muestra en la imagen de la siguiente página:

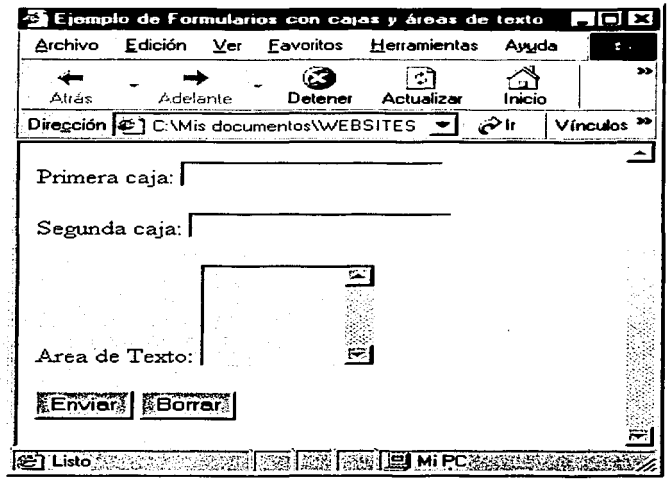

Figura 13. Ejemplo de formulario[13]

2.9.4 Casillas de Verificación

Si se desea seleccionar de varias opciones algunas, las casillas de verificación serán útiles. La sintaxis es al siguiente:

```
<INPUT TYPE=CHECKBOX NAME="nombre del campo">
```

Con el valor CHECKBOX en el atributo TYPE de la etiqueta INPUT se indica que es una casilla de verificación, NAME sería el nombre del campo, también se le puede agregar el

[13] *Figura realizada por el autor.*

atributo CHECKED que indica que la casilla se encuentra seleccionada de manera predeterminada. Esto sería:

```
<INPUT TYPE=CHECKBOX NAME="nombre del campo" CHECKED>Opción uno
```

2.9.5 Botones de opción

Sí se desea seleccionar una opción de varias, los botones de opción sería apropiados. La sintaxis es la siguiente:

```
<INPUT TYPE=RADIO NAME="nombre del campo" VALUE="valor">
```

Donde el valor RADIO en el atributo TYPE de la etiqueta INPUT indica que es un botón de opción, el atributo NAME indica el nombre del campo, para que el navegador reconozca a varios botones como un grupo se les debe asignar el mismo nombre.

Por último, contiene el atributo VALUE que indica el valor que tendrá, además también se puede usar el atributo CHECKED para seleccionar de forma predeterminada una opción.

En la página siguiente se muestra, un ejemplo de casillas de verificación y botones de opción:

```
<HTML>
<HEAD>
<TITLE> Ejemplo de casillas de verificación y botones de opción</TITLE>
</HEAD>
<BODY>
<FORM ACTION="enviar.php" METHOD="POST">
Casillas de verificación <BR>
Selecciona algunas de las siguientes opciones:
<BR>
<INPUT TYPE=CHECKBOX NAME="casilla1" CHECKED>Opción uno <BR>
<INPUT TYPE=CHECKBOX NAME="casilla1" >Opción dos <BR>
<INPUT TYPE=CHECKBOX NAME="casilla1" >Opción tres <BR>
<INPUT TYPE=CHECKBOX NAME="casilla1" >Opción cuatro <BR>
<BR>
<HR>
<BR>
Botones de opción <BR>
Selecciona una opción de las siguientes:<BR>
<INPUT TYPE=RADIO NAME="opcion" VALUE="uno">Opcion Uno<BR>
<INPUT TYPE=RADIO NAME="opcion" VALUE="dos">Opcion Dos<BR>
<INPUT TYPE=RADIO NAME="opcion" VALUE="tres">Opcion Tres<BR>
<INPUT TYPE=RADIO NAME="opcion" VALUE="ninguna" CHECKED>Ninguna<BR>
<INPUT TYPE=SUBMIT VALUE="Enviar">
<INPUT TYPE=RESET   VALUE="Borrar">
</FORM>
</BODY>
</HTML>
```

Esto será:

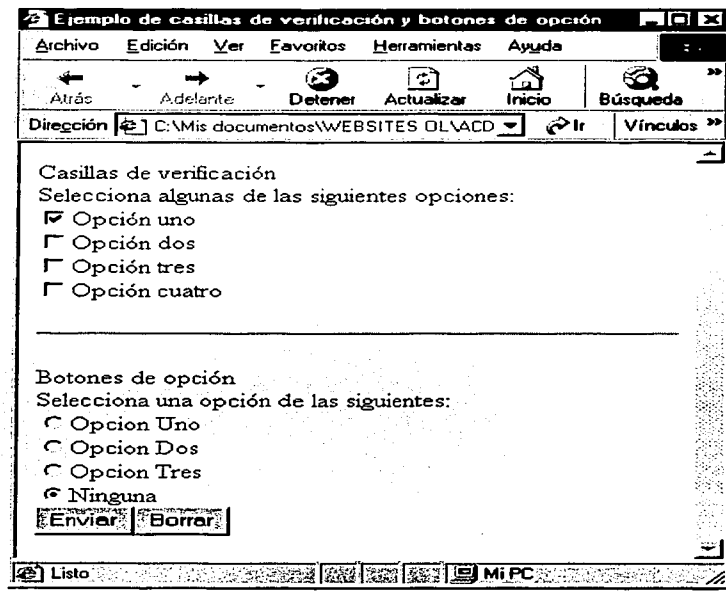

Figura 14. Ejemplo de Casillas de verificación y botones de opción [14]

2.9.6 Listas de selección

Cuando las opciones son más de cinco, los botones de opción suelen ser menos eficientes. Para esto existen las listas de selección, las cuales tienen la siguiente sintaxis:

```
<SELECT NAME="nombre del campo" SIZE= elementos>
<OPTION> Primer elemento</OPTION>
<OPTION> Segundo elemento</OPTION>
<OPTION> Tercer elemento</OPTION> etc... </SELECT>
```

[14] *Figura realizada por el autor.*

Donde las etiquetas <SELECT> y </SELECT> indican que lo contenido en ellas es una lista de selección y las etiquetas <OPTION> y <OPTION> indican un elemento de la lista.

El atributo SIZE en la etiqueta SELECT indica el número de elementos que se va a mostrar; si no se pone este atributo se generará una lista desplegable.

También si se desea tener un elemento preseleccionado hay que agregarle el atributo SELECTED, y si se desea seleccionar más de un elemento en la lista hay que agregar en la etiqueta MÚLTIPLE.

Un ejemplo del código para generar listas de opción se muestra en la siguiente página:

```
<HTML>
<HEAD><TITLE>  Ejemplo  de  casillas  de  verificación  y  botones  de
opción</TITLE></HEAD>
<BODY><FORM ACTION="enviar.php" METHOD="POST">
Selecciona de la lista desplegable<BR>
<SELECT NAME="desplegable1">
<OPTION>UNO</OPTION>
<OPTION>DOS</OPTION>
<OPTION>TRES</OPTION>
<OPTION>CUATRO</OPTION>
</SELECT><BR>Selecciona de la lista de opciones:<BR>
<SELECT NAME="lista1"  SIZE=4>
<OPTION>UNO</OPTION>
<OPTION>DOS</OPTION>
<OPTION SELECTED>TRES</OPTION>
<OPTION>CUATRO</OPTION>
</SELECT><BR>
Selecciona varias opciones de la lista:<BR>
<SELECT NAME="lista1"  SIZE=4 MULTIPLE>
<OPTION>UNO</OPTION>
<OPTION>DOS</OPTION>
<OPTION>TRES</OPTION>
<OPTION>CUATRO</OPTION>
</SELECT><BR>
<INPUT TYPE=SUBMIT VALUE="Enviar">
<INPUT TYPE=RESET   VALUE="Borrar">
</FORM>
</BODY>
</HTML>
```

Esto resulta:

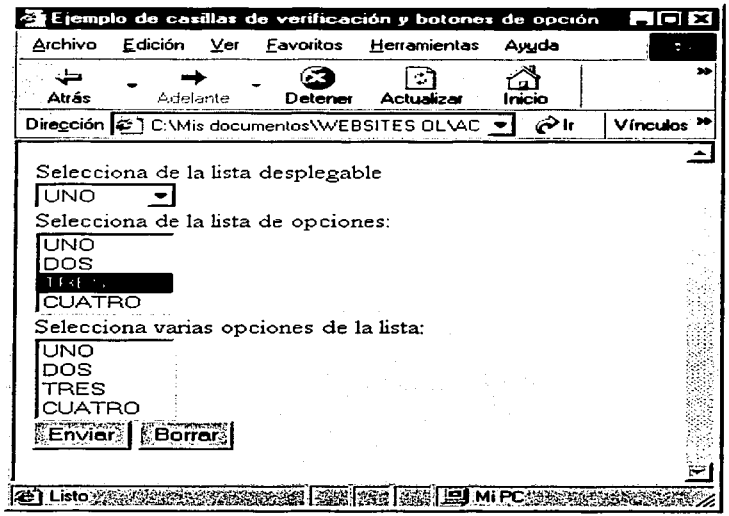

Figura 15. Ejemplo de listas de opción [15]

[15] *Figura realizada por el autor.*

CAPITULO III

EL LENGUAJE PHP

3.1 HISTORIA DE PHP

PHP fue creado en 1994 por Rasmus Lerdorf; la primera versión disponible para el público a principios de 1995 fue conocida como "herramientas para páginas web personales" (Personal Home Page Tools).

Figura 16. Rasmus Lerdorf , creador de PHP [16]

Era muy simple, sólo entendía una serie de utilidades comunes en las páginas web de entonces, un libro de visitas, un contador y otras cosas. El analizador sintáctico fue reescrito a mediados de 1995 y fue nombrado PHP/FI versión 2. Luego viene de otro programa que Rasmus había escrito y que procesaba los datos de formularios. Así que combinó las "herramientas para páginas web personales", el "intérprete de formularios"; añadió soporte para *msql* y PHP/FI vio la luz. PHP/FI creció a gran velocidad y la gente empezó a contribuir en el código.

Se estima que a finales de 1996 PHP/FI se estaba usando al menos en 15,000 páginas web alrededor del mundo. A mediados de 1997 este número había crecido a más de 50,000. A mediados de 1997 el desarrollo del proyecto se convirtió en un proyecto de grupo mucho más organizado.

[16] *Imagen tomada de:* ***http://www.phpbox.de/grundlagen/historie.php.***

El analizador sintáctico se rescribió desde el principio por Zeev Suraski y Andi Gutmans, y este nuevo analizador estableció las bases para PHP versión 3. Gran cantidad de código de PHP/FI fue aportado a php3 y otra gran cantidad fue escrito completamente de nuevo.

A finales de 1999, tanto PHP/FI como PHP3 se distribuyen en un gran número de productos comerciales tales como el servidor web "c2's stronghold" y redhat linux.

Una estimación muestra que más de 1,000,000 de servidores alrededor del mundo usan PHP. Para hacernos una idea, este número es mayor que el número de servidores que utilizan el "Netscape's Enterprise Server" en Internet.

En el año 2002, PHP se encuentra en su versión 4, que utiliza el motor "Zend", desarrollado con más calma, pensando en las necesidades actuales y en solucionar algunos inconvenientes de las versiones anteriores. Una de las ventajas de esta versión es su rapidez, ya que primero compila y luego se ejecuta, mientras que antes se ejecutaba cuando se estaba interpretando. Otra puede ser la independencia con el servidor web, creando versiones de PHP para más plataformas.

3.2 QUÉ ES PHP

PHP es acrónimo de "Hipertext PreProcessor", originalmente "Personal Home Page" como se vio anteriormente. PHP es un lenguaje de programación de scripts de lado del servidor de estilo clásico con variables, sentencias, condiciones, bucles y funciones, similar a C o Perl. Los scripts de PHP se compilan y ejecutan en el servidor web, lo cual da por respuesta páginas HTML dinámicas. Con esto se protege el código PHP, el cual nunca verá el cliente.

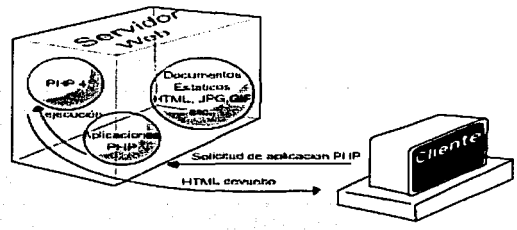

Figura 17. Ejecución de aplicación PHP [17]

En la Figura 17 se explica cómo un cliente realiza la petición de una aplicación PHP, el servidor se la envía al módulo PHP, el cual la compila, ejecuta y da como respuesta una página HTML en el navegador del cliente.

3.2.1 Código PHP y HTML

Para que el servidor web reconozca las páginas PHP y las envié su interprete PHP, éstas deben tener extensiones especiales como por ejemplo *.php*, eso esta en la configuración del servidor web.

El código PHP se introduce en el HTML mediante etiquetas (tags) delimitadoras que nos permiten saber cuando empieza y acaba un script PHP.

Existen varias maneras de delimitar el código PHP:

- Una es utilizando las etiquetas <?php y ?>.
- Otra es usando las "etiquetas cortas" (short tags) <? y ?>.
- Tambien con la etiqueta <SCRIPT> de HTML: <SCRIPT LANGUAGE="php"> y </SCRIPT>.

[17] *Figura realizada por el autor.*

- Y por ultimo usando las asp_tags <% y %>, que son las mismas se usan en las páginas ASP (Active Server Pages).

Para usar tanto las short_tags como las asp_tags, deben estar habilitadas en la configuración del servidor web.

El servidor web interpreta el código HTML y al toparse con los delimitadores PHP, pasa el código que esté dentro de estas etiquetas al interprete PHP, luego continúa con el HTML, y así sucesivamente, hasta que termine con el código de la página.

A continuación un ejemplo del código PHP dentro del HTML:

```
<HTML>
<HEAD>
<TITLE>EJEMPLO PHP</TITLE>
</HEAD>
<BODY>
   <? ECHO "HOLA, ESTE ES UN EJEMPLO CON PHP!"; ?>
</BODY>
</HTML>
```

Lo que muestra con este código es:

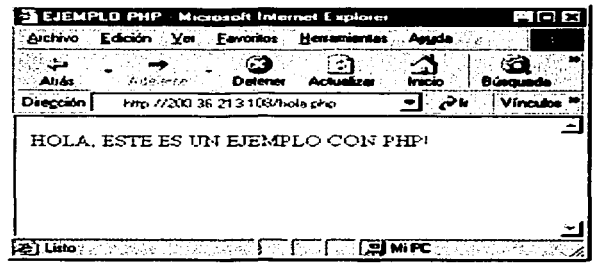

Figura 18. Ejemplo básico con PHP[18]

[18] *Figura realizada por el autor.*

3.2.2 capacidades de el PHP

Con PHP se puede fácilmente hacer todo lo que con un CGI script, como procesar información de formularios, generar páginas con contenidos dinámicos, o transferencia de cookies.

PHP soporta una gran cantidad de base de datos como pueden ser entre los más comunes Oracle, Informix, Dbase, ODBC, MS SQL Server, Interbase, Sybase, MySQL, PostgreSQL.

3.3 MOSTRANDO INFORMACIÓN

Existen varias formas de mostrar información en PHP, hay que tomar en cuenta que el interprete PHP va a ejecutar el código PHP y devolver como respuesta código HTML al navegador del cliente.

Las órdenes para mostrar información en PHP son: *echo*, *print*, *printf*, *sprintf*, a continuación se explica la más usada, que es *echo*.

3.3.1 echo

La sintaxis de *echo* es la siguiente:

```
echo "argumento" Ó
echo (argumento1, argumento2,...);
```

Donde puede ser una variable o una cadena de caracteres, si es así debe estar entre comillas. Por ejemplo si escribimos:

```
<?
$var1 = "Hola mundo";
echo "El valor de la variable \$var es " . $var ;
?>
```

La salida del código anterior será:

```
El valor de la variable $var1 es Hola mundo
```

Primero se inicializó la variable $var1 con el valor "Hola mundo", luego con *echo* se pido que mostrara la cadena entre comillas; si se tiene un nombre de variable o un carácter reservado de PHP que quiere ser mostrado se deben proteger antecediendo un "\". Aquí se puede ver con "\$var1", al final lo que se hizo fue concatenar la cadena con el valor de la variable $var1.

Si nos interesa generar HTML, podría ser así:

```
<?
$a = "Escribe aquí tu nombre";
Echo "<input name=\"CajaTexto\" value=\"$a\">";
?>
```

Lo anterior generaría una caja de texto HTML con como valor inicial "Escribe tu nombre".

3.4 VARIABLES

En PHP todas las variables comienzan con el símbolo de pesos $ y no es necesario definir una variable antes de usarla. Tampoco tienen tipos, es decir, que una misma variable puede contener un número y luego puede contener caracteres; por poner un ejemplo:

```
<html>
<head><title>ejemplo de php</title></head>
<body>
<?
   $a = 1;
   $b = 6.5;
   $c = "hola mundo";
   echo $a,"<br>",$b,"<br>",$c;
 ¿>
</body>
</html>
```

En este ejemplo hemos definido tres variables: $a, $b y $c, y con la instrucción "echo" hemos impreso el valor que contenían, insertando un salto de línea entre ellas.

Existen 2 tipos de variables: las variables locales que sólo pueden ser usadas dentro de funciones y las variables globales que tienen su ámbito de uso fuera de las funciones; podemos acceder a una variable global desde una función con la instrucción *global* nombre de la variable.

De esto se despliega la página siguiente:

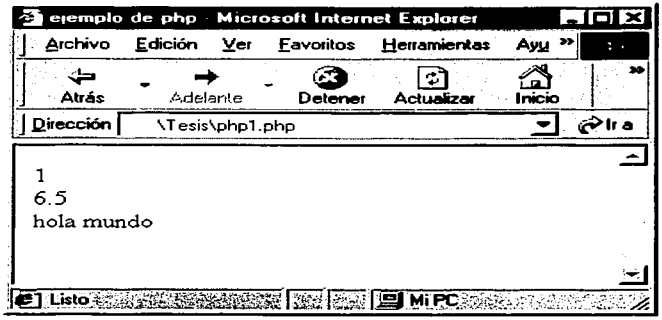

Figura 19. Despliegue de variables [19]

[19] *Figura realizada por el autor.*

3.5 OPERADORES ARITMETICOS

Los operadores de PHP son muy parecidos a los de C. Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

OPERADOR	NOMBRE	EJEMPLO	DESCRIPCIÓN
+	suma	5 + 6	suma dos números
-	resta	7 - 9	resta dos números
*	multiplicación	6 * 3	multiplica dos números
/	división	4 / 8	divide dos números
%	módulo	7 % 2	devuelve el resto de dividir ambos unmeros, en este ejemplo el resultado es 1
++	suma	1 $a++	suma 1 al contenido de una variable.
--	resta	1 $a--	resta 1 al contenido de una variable.

3.6 OPERADORES RELACIONALES

Los operadores de comparación son usados para comparar valores y así poder tomar decisiones.

operador nombre ejemplo devuelve cierto cuando:

==	igual	$a == $b	$a es igual $b
!=	distinto	$a != $b	$a es distinto $b
<	menor que	$a < $b	$a es menor que $b
>	mayor que	$a > $b	$a es mayor que $b
<=	menor o igual	$a <= $b	$a es menor o igual que $b
>=	mayor o igual	$a >= $b	$a es mayor o igual que $b

3.7 OPERADORES LÓGICOS

Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de éstas.

operador	nombre	ejemplo	devuelve cierto cuando:
&&	y	(7>2) && (2<4)	devuelve verdadero cuando ambas condiciones son verdaderas.
and	y	(7>2) and (2<4)	devuelve verdadero cuando ambas condiciones son verdaderas.
||	o	(7>2) || (2<4)	devuelve verdadero cuando al menos una de las dos es verdadera.
or	o	(7>2) or (2<4)	devuelve verdadero cuando al menos una de las dos es verdadera.
!	no	! (7>2)	niega el valor de la expresión.

3.8 SENTENCIAS CONDICIONALES

Las sentencias condicionales nos permiten ejecutar o no unas ciertas instrucciones dependiendo del resultado en una condición. Las más frecuentes son la instrucción if y la instrucción switch.

3.8.1 Sentencia if ... else

La sentencia if ejecuta una serie de instrucciones sólo si la condición es verdadera; en caso contrario, ignora este bloque y pasa al bloque que inicia con *else.* La sintaxis es la siguiente:

```
<?
if (condición)
{
sentencias a ejecutar cuando la  condición es cierta;
}
else
{
 sentencias a ejecutar cuando la condición es falsa;
}
?>
```

3.8.2 Sentencia elseif

Sí en un *if* la sentencia *else* es otro *if* se pueden sustituir ambas por *elseif* por ejemplo:

antes

```
if (expresión)
   sentencia simple o compuesta;
else
    if (expresión)
    sentencia simple o compuesta;
```

sería equivalente a

```
if (expresión)
  sentencia simple o compuesta;
elseif (expresión)
  sentencia simple o compuesta;
```

elseif tiene las mismas características que *if*.

3.8.3 Sentencia switch

La sentencia switch toma como referencia el valor de una variable y la busca entre varias opciones; en caso de encontrar una opción igual al valor de la variable, ejecuta un bloque de instrucciones respectivo. La sintaxis es la siguiente:

```
switch ($variable)
{
case valor1:
bloque de sentencias;
break;
case valor2:
bloque de sentencias;
break;
case valor3:
bloque de sentencias;
break;
default:
bloque de sentencias por defecto;
}
```

En cada opción después de su bloque de instrucciones respectivo se le agrega la instrucción *break*, la cual omite las sentencias siguientes dentro de la instrucción switch. También se puede agregar la instrucción *default*, la cual, en caso de que ningún valor concuerde con la variable procede a ejecutar un bloque de instrucciones.

3.9 CICLOS DE REPETICION

A continuación se presentarán los ciclos de repetición que utiliza PHP, los cuales son similares a los ya conocidos de C.

3.9.1 Sentencia while

Con *while* se permite repetir una sentencia un número indeterminado de veces dependiendo de una condición. *While* repetirá el ciclo mientras la condición sea verdadera. Se puede modificar la condición para cortar o reactivar el ciclo de repetición con *break* y *continue*.

La sintaxis de *while* es la siguiente:

```
while (expresión)
{
Sentencia simple  o compuesta;
}
```

3.9.2 Sentencia do..while

Este también permite ejecutar la sentencia hasta que la condición sea falsa, con la diferencia respecto al *while* que en éste al menos se ejecuta una vez.

También se puede utilizar *break* para terminar el ciclo y *continue* para volver al principio.

La sintaxis de *do..while* es la siguiente:

```
do
{
sentencia;
}
while (expresión);
```

3.9.3 Sentencia for

El ciclo *for* tiene como función repetir un bloque de instrucciones hasta que se cumpla una condición definida.

La sintaxis es la siguiente:

```
for (inicialización; condición; actualización)
{
sentencias;
}
```

En el ciclo *for* existen tres parámetros:

- Es la *inicialización* determina cual es el valor de partida
- Es la *condición* que determina si se debe continuar con el ciclo o no.
- La *actualización* es una expresión que varía según van pasando las iteraciones del ciclo.

Ejemplo:

```
for ($i = 1; $i <= 5; $i++)
{
     echo "El numero es: $i";
}
```

3.10 PASO DE VARIABLES DE UN FORMULARIO HTML A PHP

El paso de variables de un formulario HTML a una aplicación PHP puede ser como, ya se vio anteriormente, por los métodos GET o POST.

Para capturar los datos que se necesiten en la aplicación PHP se debe generar primero un formulario HTML, en el cual cada elemento debe tener el nombre de la variable PHP a la que se le va a asignar el valor, por ejemplo la caja de texto "nombre" que va a capturar el

nombre del usuario, etc. También se le indica el método de petición y a qué aplicación va a mandar los datos capturados.

Teniendo el formulario hecho, cuando se llene éste, enviará los datos capturados a la aplicación PHP y ésta automáticamente los convertirá en variables de PHP con el mismo nombre antecedido por "$".

3.10.1 Ejemplo de paso de variables de HTML a PHP

A continuación se muestra un ejemplo del paso de variables de HTML a PHP. Este ejemplo pedirá datos del usuario y calculará su edad.

Para empezar se generará una página HTML llamada "pasoaPHP.htm", la cual contendrá un formulario con los campos: nombre, año de nacimiento (a_nacimiento) y año actual (a_actual).

El codigo de "pasoaPHP.htm" se muestra en la siguiente página:

```
<html>
<head>
<title>Paso de variabled HTML-PHP</title>
</head>
<body>
<form name="form1" method="get" action="calcula_edad.php">
<center>
<strong>Ejemplo de paso de variables de un formulario HTML a una pagina
PHP</strong>
<br>
<hr>
    <strong>Calcula tu edad!!!</strong>
<br>
    &iquest;Cual es tu nombre?
    <input name="nombre" type="text" id="nombre">
<br>
    &iquest;En que a&ntilde;o naciste?
    <input name="a_nacimiento" type="text" id="a_nacimiento">
<br>
    &iquest;En que a&ntilde;o estamos?
    <input name="a_actual" type="text" id="a_actual">
<br>
    <input type="submit" value="Calcula">
    <input name="Reset" type="reset" value="Borrar">
</center>
</form>
</body>
</html>
```

En el código anterior aparece la palabra *¿* que indica un símbolo "¿" al igual que *ñ* el cual indica una "ñ".

Esto en el navegador se ve como en la imagen de la siguiente página:

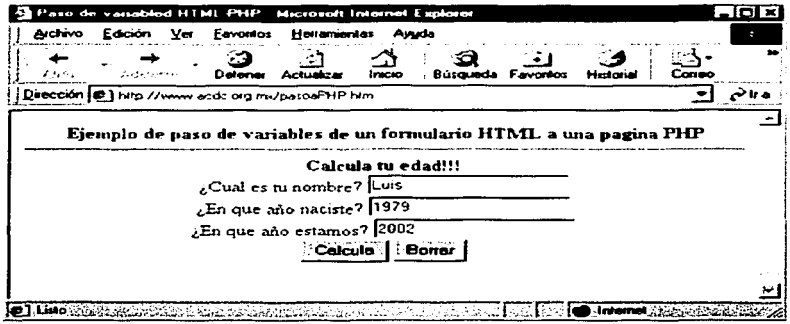

Figura 20. Formulario "pasoaPHP.htm" [20]

Los datos serán enviados a la aplicación PHP llamada "calcula_edad.php", la cual tomará los valores y los reconocerá como variables PHP. La sintaxis de esta es la siguiente:

```
<html>
<head><title>Calcula Edad</title></head>
<body>
<?
echo $nombre,"<BR>";
echo $a_nacimiento,"<BR>";
echo $a_actual,"<BR>";
$edad = $a_actual - $a_nacimiento;
?>
<center>
  <strong>
<? echo $nombre," tienes: ",$edad," a&ntilde;os de edad" ?>
</strong>
</center>
</body>
</html>
```

[20] *Figura realizada por el autor.*

Lo primero que hace, la aplicación es desplegar los valores obtenidos en "pasoaPHP.htm", luego procesa en una operación de resta los valores $a_actual y $a_nacimiento, guardando el resultado en la variable $edad. Por último, despliega $nombre y $edad. Esto en el navegador se verá de la siguiente forma:

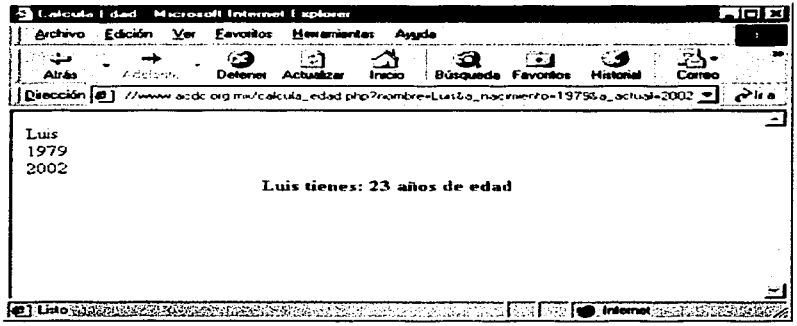

Figura 21. Aplicación "calcula_edad.php" [21]

[21] *Figura realizada por el autor.*

CAPITULO IV

PHP Y LAS BASES DE DATOS

4.1 ¿QUÉ ES UNA BASE DE DATOS?

Una base de datos es una serie de datos organizados y relacionados entre sí, los cuales son recolectados y explotados por los sistemas de información de una empresa o negocio en particular.

4.1.1 El sistema manejador de bases de datos (DBMS)

Es un conjunto de programas que se encargan de manejar la creación, y todos los accesos a las bases de datos. Se compone de un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consulta.

Una de las ventajas de este sistema es que puede ser invocado desde programas de aplicación que pertenecen a sistemas transaccionales escritos en algún lenguaje de alto nivel, para la creación o actualización de las bases de datos, o bien para efectos de consulta a través de lenguajes propios que tienen las bases de datos o lenguajes de cuarta generación.

4.1.2 Modelos de base datos

Existen fundamentalmente tres alternativas disponibles para diseñar las bases de datos: el modelo jerárquico, el modelo de red y el modelo relacional.

Modelo jerárquico

Puede representar dos tipos de relaciones entre los datos: relaciones de uno a uno y relaciones de uno a muchos.

Modelo de red

Este modelo permite la representación de muchos a muchos, de tal forma que cualquier registro dentro de la base de datos puede tener varias ocurrencias superiores a él. El modelo de red evita redundancia en la información, a través de la incorporación de un tipo de registro denominado el conector.

Modelo relacional

Este modelo se está empleando con más frecuencia en la práctica, debido a la ventajas que ofrece sobre los dos modelos anteriores, entre ellas, el rápido entendimiento por parte de usuarios que no tienen conocimientos profundos sobre sistemas de bases de datos.

4.2 ¿QUÉ ES SQL?

SQL acrónimo de Lenguaje Estructurado de Consultas (*Structured Query Language*) es un lenguaje empleado para crear, manipular, examinar, y manejar bases de datos relaciónales.

Proporciona una serie de sentencias estándar que permiten realizar las tareas antes descritas. SQL fue estandarizado según las normas ANSI (*American National Standards Institute*) en 1992, eliminando de alguna forma la incompatibilidad de los productos de los distintos fabricantes de bases de datos como Oracle, Sybase, Microsoft, Informix, etc. Esto quiere decir que una misma sentencia permite manipular los datos recogidos en cualquier base de datos que soporte el estándar ANSI, con independencia del tipo de base de datos.

La mayoría de los programas de base de datos soportan el estándar *SQL-92*, y adicionalmente proporcionan extensiones al mismo, aunque éstas ya no están estandarizadas y son propias de cada fabricante.

4.2.1 Reglas sintácticas del SQL

SQL tiene su propia sintaxis que hay que tener en cuenta, pues a veces puede ocurrir que sin producirse ningún problema en la compilación, al tratar de ejecutar una sentencia se produzca algún error debido a una incorrecta sintaxis en la sentencia.

Por tanto, será necesario seguir las siguientes normas:

- SQL no es sensible a los espacios en blanco. Los retornos de carro, tabuladores espacios en blanco no tienen ningún significado especial. Las palabras clave y comandos están delimitados por comas (,) y cuando sea necesario, debe emplearse el paréntesis para agruparlos.
- Si se van a realizar múltiples consultas a un mismo tiempo, se debe utilizar el punto y coma (;) para separar cada una de las consultas. Además, todas las sentencias SQL deben finalizar con el carácter punto y coma (;).
- Las consultas son insensibles a mayúsculas y minúsculas. Sin embargo, los valores almacenados en las bases de datos sí que son sensibles a las mismas, por lo que habrá que tener cuidado al introducir valores, efectuar comparaciones, etc.
- Para caracteres reservados como % o _ pueden y deben emplearse los *caracteres escape*. Así por ejemplo en lugar de % habrá de emplearse \%.
- A la hora de introducir un una cadena de caracteres, ésta deberá ir encerrada entre comillas simples, ya que de lo contrario se producirán errores en la ejecución.

4.3 SINTAXIS DE MySQL

A continuación se mostrará la sintaxis de MySQL el cual es un manejador de base de datos basado en SQL. Este manejador tiene ligeras diferencias con respecto al estándar de ANSI SQL-92, las cuales se mencionarán en su momento.

Para los ejemplos siguientes se utilizarán la siguiente base de datos para hacer más claro el uso de la sintaxis de MySQL.

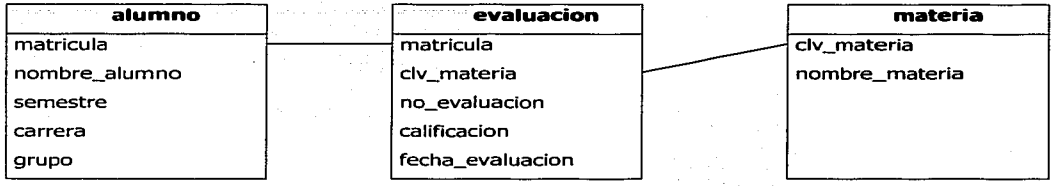

Figura 22. Tablas de ejemplo [22]

4.3.1 Creación de tablas

Para la creación de tablas se emplea la sentencia CREATE TABLE. La sintaxis es la siguiente:

```
CREATE TABLE nombre de tabla
(
Campo tipo_dato [NOT NULL | NULL] [DEFAULT valor] [AUTO INCREMENT]
ó PRIMARY KEY [nombre_indice,...]
ó KEY [[nombre_indice,...]
ó INDEX [nombre_indice,...]
ó UNIQUE [nombre_indice,...]
ó [CONSTRAINT simbolo]
FOREIGN KEY (nombre_indice,...)[referencia] o CHECK (campo))
```

Donde a cada campo se le asigna un nombre, y un tipo de dato. De manera opcional si se desea que nunca esté vacío se le agrega NOT NULL o en caso contrario NULL.

[22] *Figura realizada por el autor.*

También existe el comando DEFAULT el cual asigna un valor predeterminado al campo. PRIMARY KEY indica si el campo es llave primaria.

En caso de querer generar un índice existe en comando INDEX, esto sirve para en caso de que la búsqueda de datos sea lenta, para ello se necesita claves que indiquen el orden la cuales se generan con KEY. UNIQUE sirve para asegurar que en un campo tenga los valores nunca se repitan.

Las cláusulas FOREIGN KEY, CHECK, y REFERENCES en MySQL no tienen efecto únicamente se aceptan para facilitar la migración de otras base de datos SQL. Un ejemplo de esto podría ser:

```
create table alumno (
matricula char(15) not null primary key,
nombre_alumno char(50),
semestre char(15),
carrera char(30),
grupo char(5)
)
```

4.3.2 Recuperación de información

La sentencia SELECT es la que se utiliza cuando se quieren recuperar datos de la información almacenada en un conjunto de columnas. Las columnas pueden pertenecer a una o varias tablas y se puede indicar el criterio que deben seguir las filas de información que se extraigan.

La sintaxis de la sentencia es:

```
SELECT [ALL | DISTINCT |DISTINCTROW]
[tabla].campo
FROM tablas
WHERE condiciones
[ORDER BY <columna> [ASC | DESC][,<columna> [ASC | DESC]]...]
```

La selección contiene normalmente una lista campos separados por comas (,), o un asterisco (*) para seleccionarlas todas. Un ejemplo podría ser:

```
SELECT * FROM alumno;
```

Que devolvería el contenido completo de la tabla "alumno". Si solamente se quiere conocer los datos del alumno cuya matricula es lfzv1979, la consulta sería:

```
SELECT * FROM alumno WHERE matricula='lfzv1979';
```

Se quiere ahora saber cuántos alumnos en la carrera "Ingeniería en Computación". Para ordenar la lista resultante por el nombre del alumno, se usaría la directiva ORDER BY:

```
SELECT * FROM alumno
WHERE carrera = 'Ingeniería en Computación'
ORDER BY nombre_alumno;
```

Puede especificarse que esta ordenación es realizada de forma ascendente con la cláusula ASC o descendente con DESC. Así, si se quiere ordenar de forma ascendente:

```
SELECT * FROM alumno
WHERE carrera = 'Ingeniería en Computación'
ORDER BY nombre_alumno ASC;
```

Si lo que se quiere, además de que la lista esté ordenada por el nombre del alumno, es ver solamente la matricula, se consultaría de la forma:

```
SELECT matricula FROM alumno
WHERE carrera = 'Ingenieria en Computacion'
ORDER BY nombre_alumno;
```

Si se quieren resultados de dos tablas a la vez, tampoco hay problema en ello, tal como se muestra en la siguiente sentencia:

```
SELECT alumno.*, materia.* FROM alumno, materia;
```

Obsérvese que se especifica el nombre de la tabla y con el asterisco se indica que muestre todos sus campos.

Se puede utilizar el carácter % como comodín para especificar términos de consulta que empiecen por un determinado valor. Así, para recuperar todos los datos de los alumnos cuyo nombre comienza por "J":

```
SELECT * FROM alumno WHERE nombre_alumno LIKE 'J%' ;
```

Donde LIKE indica que busque los registros del campo nombre_alumno en los cuales comiencen con "J". De igual manera se pueden excluir resultados de una consulta con la cláusula NOT LIKE. Esto sería:

```
SELECT * FROM alumno WHERE nombre_alumno NOT LIKE 'J%' ;
```

Además se pueden emplear las cláusulas lógicas AND, OR y NOT. Por ejemplo si se quiere ver que alumnos cursan el noveno semestre de la carrera de derecho sería:

```
SELECT nombre_alumnos FROM alumno WHERE carrera='Derecho' AND semestre ='noveno';
```

El orden de los operadores lógicos es el siguiente:

1. NOT
2. AND
3. OR

Por último, la cláusula DISTINCT elimina cualquier fila duplicada que pueda obtenerse como resultado de una consulta a varias tablas relacionadas entre sí.

4.3.3 Almacenar información

La sentencia INSERT se utiliza cuando se quieren insertar filas de información en una tabla. La sintaxis de la sentencia es:

```
INSERT INTO nombre tabla
[(nombre campo,...)]
VALUES (valor,...)
```

Por ejemplo, en la tabla "alumno" se podría ingresar un registro nuevo con la siguiente información:

```
INSERT INTO alumno
VALUES ( 'jgnel458', 'Juan Gutiérrez Najera', 'septimo', 'Relaciones
Internacionales', 'A');
```

4.3.4 Eliminación de datos

La sentencia DELETE es la que se emplea cuando se quieren eliminar filas de las tablas, sus sintaxis es:

```
DELETE FROM nombre tabla WHERE condicion busqueda
```

Si no se especifica la cláusula WHERE, se eliminará el contenido de la tabla completamente, sin eliminar la tabla, por ejemplo:

```
DELETE FROM alumno;
```

DELETE vaciará completamente la tabla, dejándola sin ningún dato en las columnas, es decir, esencialmente lo que hace es borrar todas las columnas de la tabla.

Especificando la cláusula WHERE, se puede introducir un criterio de selección para el borrado, por ejemplo:

```
DELETE FROM alumno WHERE matricula='dsf123';
```

4.3.5 Actualización de datos

Para actualizar filas ya existentes en las columnas, se utiliza la sentencia UPDATE, cuya sintaxis es la siguiente:

```
UPDATE nombre tabla SET nombre campo= ( valor | NULL ), nombre campo= ( valor |
NULL ),...
WHERE condicion de busqueda
```

Este comando permite cambiar uno o más campos existentes en una fila. Por ejemplo, para cambiar el nombre de un alumno en la tabla de alumno, se haría:

```
UPDATE alumno SET nombre_alumno = 'Gorge Perez' WHERE matricula='lsk123';
```

Esto actualizará el nombre que actualmente tiene el registro con matricula "lsk123".

4.4 ACCESO A BASE DE DATOS EN PHP

El acceso a base de datos es una de las características más importantes de PHP, La generación de contenidos dinámicos en páginas web normalmente se basa en la información contenida en base de datos que se accede en función de los usuarios la soliciten, ésta es devuelta como un documento HTML.

Prácticamente se puede consultar cualquier motor de base de datos SQL, bien en modo nativo o mediante ODBC para plataformas Microsoft.

PHP soporta:

MySQL, PosgreSQL, mSQL, Informix, Interbase, SQL Server, Oracle , Sybase o trabajar con fichero DBF.

En general se requieren las siguientes funciones para el acceso de base de datos para todos los motores de DB.

- Una función para conectarse a la base de datos.
- Una función para realizar una consulta.
- Una función para acceder a los resultados de una consulta.
- Otras funciones auxiliares e informativas.

En este caso se trabajará con MySQL, el cual es de los más utilizados actualmente debido a su simplicidad y rapidez.

4.4.1 Comparativa de DBMS en la creación de websites dinámicos con PHP

DBASE

PHP tiene funciones para el acceso a datos en formato DBF, formato de Dbase. Los archivos Dbase son simples ficheros secuenciales con registros de longitud fija. Una vez creado el archivo la definición de la base de datos es fija.

Las funciones para Dbase no soportan índices o campos tipo memo, tampoco soportan el bloqueo de registros.

Se recomienda no utilizar archivos Dbase como base de datos sino elegir cualquier servidor SQL.

Informix

Existen funciones para este potente DBMS, aunque algunas de sus funcionalidades todavía no se pueden usar, ya que se encuentran en construcción las funciones para ello.

También se requiere de configurar el intérprete de PHP para que reconozca el modulo de funciones para Informix.

Algo en su contra podría ser el costo de licencias y la necesidad de dominarlo, ya que es un manejador muy robusto pero también un poco complicado.

Interbase

Motor desarrollado por Borland muy robusto, pero también como el anterior con desventajas en cuanto a costo de licencias y la documentación, la cual se da en la misma casa de software.

Microsoft SQL Server

Puede ser una buena opción para los que lo manejan, aunque en la web, no es muy conveniente, ya que además de requerir servidores muy grandes, también es grande el costo en cuanto a licencias de uso, en comparación con servidores Linux, en los cuales el licenciamiento es casi nulo.

Oracle

Uno de los manejadores de base de datos más conocidos, es robusto y caro, en cuanto a documentación existe mucha. PHP incluye funciones para Oracle.

Se debe configurar el intérprete de PHP para que acepte el módulo de Oracle.

PostgreSQL

Este manejador es *open source* lo cual lo hace gratuito y abierto a mejoras.

Es pionero de muchos conceptos relacionales y orientados a objetos que manejan ya algunos manejadores comerciales.

Soporta el llamado de llaves foráneas, las operaciones con transacciones, y la anidación de la sentencia SELECT, lo cual lo hace un manejador muy robusto, aunque ligeramente lento en entornos web.

MySQL

Este manejador es de los más usados para la Web, por la disponibilidad de licencias gratis, ya que es *software libre*, esto hace que se tenga acceso al código fuente, funciona en las principales plataformas, y tiene un gran rendimiento con PHP.

Para MySQL no hay que configurarle nada al interprete de PHP, ya que lo contempla como nativo. Su velocidad se debe a su sencillez en cuanto a código, con la desventaja que tiene algunas limitantes, en éste como el hecho de no soportar las llaves foráneas, transacciones, ni sentencias SELECT anidadas.

4.4.2 Conexión a base de datos

Para generar una conexión en general se necesita la siguiente información:

- Dirección del servidor de DB.
- Usuario de la base de datos.
- Clave de la base de datos.
- Nombre de la base de datos.
- Puerto de la conexión (en algunos casos).

Como resultado de estos datos debemos obtener un identificador de conexión que se utilizará posteriormente para realizar las operaciones de consulta.

Existen dos tipos de conexiones, las persistentes y las no persistentes, la única diferencia entre estas es que las primeras no terminan con el programa, y una posterior solicitud de conexión verifica si hay alguna abierta de las mismas características, y si es así la utiliza.

4.4.3 Conexión con MySQL

La forma de conexión con una base de datos MySQL es de la siguiente manera:

En primer lugar se establece la conexión con el servidor MySQL y a continuación se selecciona una base de datos.

```
$con = mysql_connect("servidor","usuario","password");
mysql_select_db("base de datos",$con);
```

Donde en la variable "$con" se guardara un identificador de la conexión, el cual se usará en otras funciones que utilicen la conexión.

El parámetro "servidor" es la dirección IP del servidor MySQL, en caso de que sea local se le asigna el valor "localhost", "usuario" es el nombre de usuario y "password" es la contraseña del usuario en el servidor MySQL.

Algo que es importante mencionar del servidor MySQL es la seguridad referente a acceso de usuarios, ya que para poder acceder al él, éste tiene que dar de alta al usuario registrando el *host* del cual el usuario se va a conectar. Esto le da la seguridad al servidor de que es el usuario autorizado.

Se puede ser utilizada la función *die()* con la conexión para terminar el programa si no se consigue ésta.

```
$dbs = mysql_select_db("bd",$con) OR die("No puedo conectar con $base en $servidor");
```

Las conexiones persistentes con MySQL se realizan con:

```
$con = mysql_pconnect("servidor","usuario","password");
```

4.4.3.1 *Consulta con MySQL*

Ya establecida una conexión con el servidor MySQL se necesita realizar consultas.

Las funciones que realizan las consultas toman como parámetros una cadena de caracteres que define la consulta SQL y el identificador de conexión.

Las funciones de consulta devuelven un identificador de cursor en el que está almacenado el resultado de la consulta, el cual se utilizará para obtener los datos de las tablas.

La función para ejecutar consultas MySQL es *mysql_query()* y su sintaxis es:

```
$consulta = "SELECT * FROM TABLA WHERE id > 100 " ;
$res = mysql_query($consulta);
```

4.4.3.2 *Obtener datos de una consulta*

Existen dos maneras de obtener la información de una consulta, por filas (tuplas) ó de manera individual por nombre del campo y numero de fila. A continuación las funciones.

a) Para recuperar la información de manera individual se utiliza la función *mysql_result()*:

```
$valor = mysql_result($res,$num_fila,$campo);
```

Donde "$res" es el identificador de cursor que arroja la función *mysql_query()* ,"$num_fila" indica el numero de fila del resultado que queremos recuperar, el numero de fila comienza por cero, "$campo" puede ser el numero de campo del resultado o el nombre del campo que queremos. El número de campo comienza por cero.

b) Para recuperar las filas obtenidas de una consulta como arreglos se utiliza la función *mysql_fetch_row()*:

```
$fila = mysql_fetch_array($res);
```

Esta función devuelve el resultado de una fila en forma de arreglo cada vez que se ejecuta, utilizando un puntero que se incrementa en uno cada vez que se ejecute la función.

4.4.3.3 *Número de filas obtenidas en una consulta*

La función para obtener el número de filas obtenidas por la consulta es:

```
$num_fila = mysql_num_rows($res);
```

4.4.3.4 *Número de campos obtenidos en una consulta*

La función para obtener el número de campos obtenidos por la consulta es:

```
$num_cols = mysql_num_fields($res);
```

4.4.3.5 *Relación entre el nombre de campo y número de columna*

Para desplegar el nombre de un campo determinado se utiliza la función *mysql_field_name()*. La sintaxis es:

```
$nombre_campo = mysql_field_name($res, $num_campo);
```

Donde "$num_campo" es el número de campo en la fila.

4.4.3.6 *Cierre de conexión y liberación de recursos*

Para liberar los recursos del cursor de consulta se utiliza la función *mysql_free_result()*. La cual es:

```
mysql_free_result($res);
```

donde "$res" es el cursor de resultado de una consulta.

Para cerrar una conexión con MySQL se utiliza la función *mysql_close()*. La sintaxis es:

```
mysql_close($con);
```

Donde "$con" es el identificador de la conexión.

4.4.3.7 *Recorrido de cursores*

Para obtener información de la base de datos por lo general se siguen los siguientes pasos:

Pasos	Función a usar
Conectarse al servidor MySQL.	mysql_connect()
Seleccionar la base de datos.	mysql_select_db()
Generar la consulta SQL.	Generar una cadena de caracteres
Ejecutar la consulta.	mysql_query()
Desplegar el resultado de el cursor.	mysql_result(), mysql_fetch_row() ó mysql_fetch_array()
Liberar los recursos del cursor.	mysql_free_result()
Cerrar la conexión MySQL.	mysql_close()

Figura 23. Pasos para el despliegue de información de una consulta [23]

Un ejemplo de lo anterior usando MYSQL_FETCH_ROW() se muestra a continuación:

[23] *Figura realizada por el autor.*

```
<html>
<head>
<title>Ejemplo consulta bd usando mysql_fetch_row()</title>
</head>
<body>
<?
$con = mysql_connect("localhost","ulsab_usr","ulsab1910");
mysql_select_db("consulta_calif",$con);
$query = "select * from alumno";
$result = mysql_query($query);
$num_camp = mysql_num_fields($result);
echo"<table border=1>";
while($datos = mysql_fetch_row($result))
{
echo"<tr>
<td>Matricula</td>
<td>Nombre</td>
<td>Semestre</td>
<td>Carrera</td>
<td>Grupo</td>
</tr><tr>";
for($a=0;$a<$num_camp;$a++){
echo "<td>",$datos[$a],"</td>";
}
echo "</tr>";
}
echo "</table>";
?>
</body>
</html>
```

El código anterior da como resultado lo que se muestra en la figura de la siguiente página:

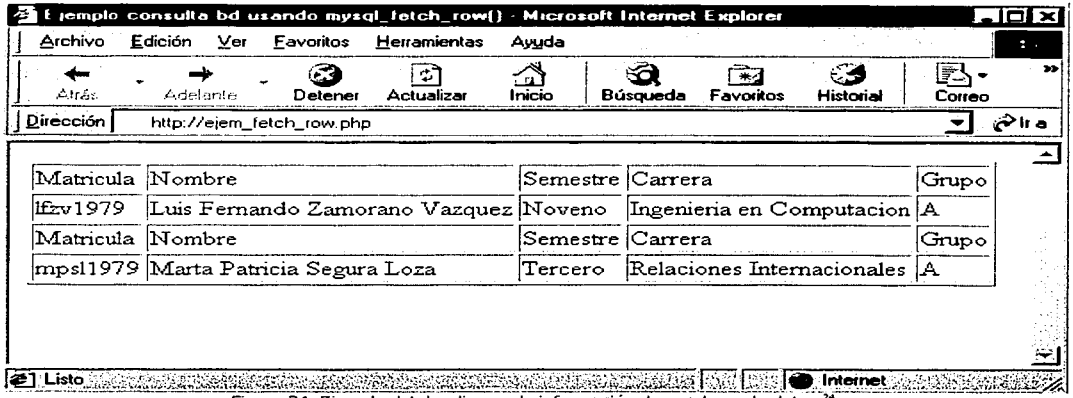

Figura 24. Ejemplo del despliegue de información de una base de datos [24]

[24] *Figura realizada por el autor.*

4.4.3.8 *Otras funciones*

Existen otras funciones para trabajar con la base de datos MySQL. A continuación se pondrá la lista de las funciones disponibles para MySQL.

mysql_affected_rows	Devuelve el número de filas afectadas de la última operación MySQL.
mysql_change_user	Cambia el usuario conectado en la conexión activa.
mysql_close	Finaliza una conexión con MySQL.
mysql_connect	Abre una conexión a un servidor MySQL.
mysql_create_db	Crea una base de datos MySQL.
mysql_data_seek	Mueve el puntero interno.
mysql_db_query	Envía una consulta MySQL al servidor.
mysql_drop_db	Borra una base de datos MySQL.
mysql_errno	Devuelve el número del mensaje de error de la última operación MySQL.
mysql_error	Devuelve el texto del mensaje de error de la última operación MySQL.
mysql_fetch_array	Extrae la fila de resultado como una matriz asociativa.
mysql_fetch_field	Extrae la información de una columna y la devuelve como un objeto.
mysql_fetch_lengths	Devuelve la longitud de cada salida en un resultado.
mysql_fetch_object	Extrae una fila de resultado como un objeto.
mysql_fetch_row	Devuelve una fila de resultado como arreglo.
mysql_field_name	Devuelve el nombre del campo especificado en un resultado.
mysql_field_seek	Asigna el puntero del resultado al offset del campo especificado.
mysql_field_table	Devuelve el nombre de la tabla donde esta el campo especificado.
mysql_field_type	Devuelve el tipo del campo especificado en un resultado.
mysql_field_flags	Devuelve los flags asociados con el campo especificado en un resultado.
mysql_field_len	Devuelve la longitud del campo especificado.
mysql_free_result	Libera la memoria del resultado.
mysql_insert_id	Devuelve el identificador generado en la última llamada a insert.
mysql_list_fields	Lista los campos del resultado MySQL.
mysql_list_dbs	Lista las bases de datos disponibles en el servidor MySQL.
mysql_list_tables	Lista las tablas en una base de datos MySQL.
mysql_num_fields	Devuelve el numero de campos de un resultado.
mysql_num_rows	Devuelve el numero de filas de un resultado.
mysql_pconnect	Abre una conexión persistente al servidor MySQL.
mysql_query	Envía una sentencia SQL a MySQL
mysql_result	Devuelve datos del resultado de una consulta.
mysql_select_db	Selecciona un base de datos MySQL.
mysql_tablename	Devuelve el nombre de la tabla de un campo.

Figura 25. Funciones PHP para MySQL.[25]

[25] *Basado en FÁBREGA, PEDRO PABLO, **PHP4**, Madrid, Editorial Pearson Educación, 2000, p.p.271 y 272.*

CAPITULO V

CASO PRACTICO

5.1 ANTECEDENTES

En la actualidad el acceso a la información es algo de gran importancia en todos los sectores de la sociedad. La Web es una potente herramienta para divulgar información.

Con herramientas como PHP y MySQL, se pueden generar sitios Web de contenido dinámico y de interés para las organizaciones.

En este caso se va generar una aplicación web para la Universidad Lasallista Benavente (ULSAB) la cual es una institución educativa fundada el 18 de agosto de 1969, ésta actualmente cuenta con 4000 alumnos, ubicados en cuatro áreas principales primaria, secundaria, preparatoria y profesional.

El área profesional cuenta con las siguientes carreras:

Licenciatura en Derecho.

Licenciatura en Ciencias de la Comunicación.

Licenciatura en Relaciones Internacionales.

Licenciatura en Contaduría.

Ingeniería en Computación.

Licenciatura en Educación Preescolar.

Licenciatura en Educación Primaria.

Licenciatura en Informática.

Actualmente en la sección profesional surgió la necesidad de informar a los alumnos y familiares el estado académico actual de los mismos de una manera más ágil y rápida, ya que actualmente se realiza mandando la boleta de calificaciones por correo tradicional. De

esta manera es lento y en algunas veces inútil, ya que en ocasiones la correspondencia no llega a su destino por circunstancias ajenas.

También existen alumnos foráneos, para los que es más difícil hacerles llegar la información.
Para solucionar esto se desarrollará una aplicación Web para la consulta de calificaciones.

5.2 OBJETIVO GENERAL DEL PROYECTO

El objetivo general del proyecto es generar un servicio informativo para la comunidad estudiantil de la sección profesional, el cual les proporcione las calificaciones de sus evaluaciones de una manera rápida, eficaz, confidencial y segura.
Teniendo en cuenta que el acceso a Internet es cada vez mayor, la información se podrá consultar desde cualquier computadora que se encuentre conectada a Internet, con esto los familiares y alumnos podrán estar informados de una manera rápida y fácil.

5.3 OBJETIVOS PARTICULARES DEL PROYECTO

A continuación se describirán los objetivos del proyecto y las herramientas para llevarlos a cabo.

Ser accesible:
La consulta podrá realizarse en cualquier computadora conectada a Internet.

Mostrar información oportuna:

En cuanto la información se introduzca a la base de datos MySQL, el sitio web actualizará la información instantáneamente.

Ser confiable:

La información será directamente proporcionada por la institución.

Ser confidencial y personal

Para poder ingresar a la consulta se debe ingresar matricula de alumno y password, el alumno únicamente verá su información.

5.4 ESQUEMA GENERAL

Actualmente la universidad cuenta con un sitio web, en el cual se le incorporará la aplicación de consulta de calificaciones.

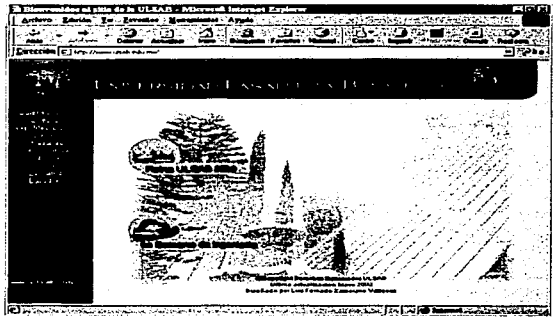

Figura 26. Pagina web de la ULSAB [26]

[26] *Figura realizada por el autor.*

En un panorama general el sitio web contará con tres elementos: páginas HTML estáticas con sus respectivos archivos gráficos, la aplicación PHP y una base de datos MySQL.

En la siguiente página se muestra un diagrama de la estructura general del sitio web de la ULSAB.

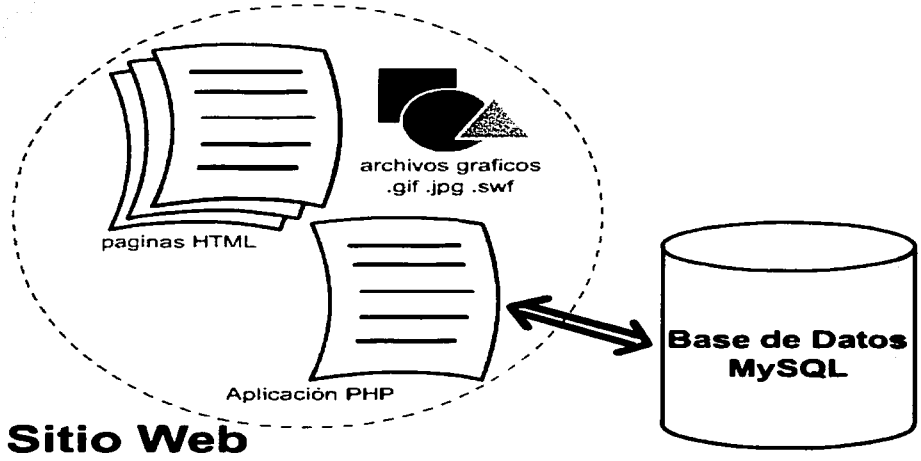

Figura 27. Diagrama general del sitio web [27]

A continuación se describirá cada uno de los elementos.

Páginas HTML estáticas: las páginas HTML contienen la información general del sitio, incluyendo gráficos y fotografías.

[27] *Figura realizada por el autor.*

Base de datos: la base de datos estará en un servidor MySQL y contendrá todos los datos de los alumnos en cuanto a sus evaluaciones y materias del semestre en curso.

La división de "Servicios Escolares" del plantel alimentará a la B.D. con toda la información previamente ordenada, validada y lista. Por lo cual la aplicación web se enfocará únicamente en la consulta de información.

Aplicación PHP: la aplicación PHP realizará una conexión con la base de datos MySQL, para luego hacer las consultas de información según el alumno y desplegar la información del mismo.

5.5 ENTIDADES DE LA BASE DE DATOS

Según los requerimientos y la estructura de la información existen cinco entidades que son: *alumno, materia, carrera, evaluación y cursando.*

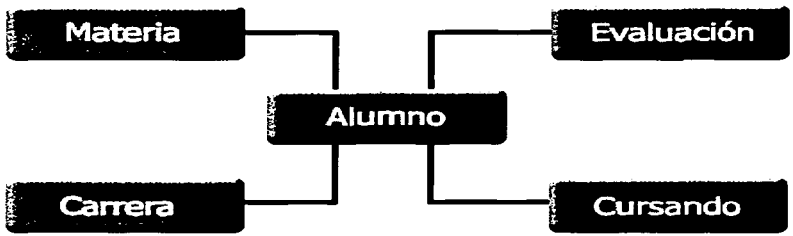

Figura 28. Entidades de la base de datos [28]

[28] *Figura realizada por el autor.*

Alumno

Contendrá la información general como lo es: matricula, nombre, clave de carrera, semestre que cursa, grupo y password.

Carrera

Contendrá las claves de carrera y sus nombres.

Materia

Almacenará: la clave de la materia, el nombre, la clave de la carrera, el grupo, y la clave del maestro asignado.

Evaluación

Almacenará: la matricula del alumno, la clave de la materia, el numero de evaluación, y la calificación.

Cursando

Contendrá: la matricula del alumno, la clave de la materia, el grupo, y la clave del la carrera.

5.6 DIAGRAMA ENTIDAD RELACIÓN

A continuación se mostrará el diagrama entidad relación de la base de datos.

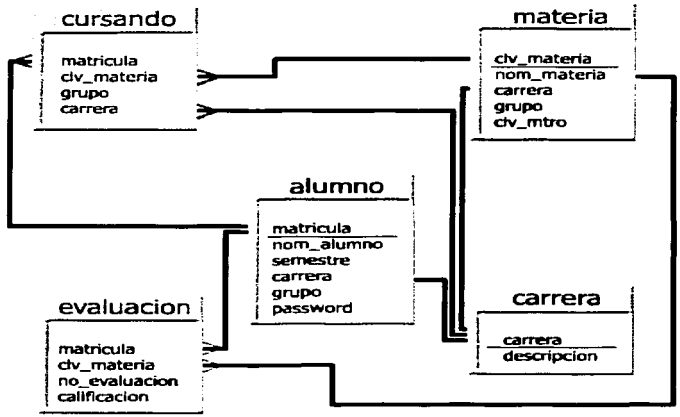

Figura 29. diagrama entidad relación [29]

5.7 CREACIÓN DE BASE DE DATOS consulta_calif EN MYSQL

Tomando en cuenta que ya existe un servidor MySQL instalado en la institución y que la mayoría de los servicios de hospedaje de sitios web en Internet proporcionan el servicio de Base de Datos MySQL no es necesario profundizar en la instalación de éste.

Tomando en cuenta la sintaxis antes vista y que MySQL no soporta llaves foráneas, esta es la sintaxis para crear la base de datos y sus tablas.

En la siguiente página se muestra el código en MySQL para la creación de la base de datos consulta_calif.

[29] *Figura realizada por el autor.*

```
create database consulta_calif;

create table carrera
(
carrera int(2) not null primary key,
descripcion char(50)
);

create table alumno
(
matricula int(9) not null primary key,
nom_alumno char(50),
semestre int(2),
carrera int(2) not null,
grupo char(1),
password char(6) not null
);

create table material
(
clv_materia int(4) not null primary key,
nom_materia char(30),
carrera int(2) not null,
grupo char(1),
clv_mtro int(8) not null
);

create table evaluacion
(
matricula int(9) not null,
clv_materia int(4) not null,
no_evaluacion int(1),
calificacion decimal(2,2)
);

create table cursando
(
matricula int(9) not null,
clv_materia int(4) not null,
grupo char(1),
carrera int(2) not null
);
```

5.8 APLICACIÓN

La aplicación cuenta con dos elementos: uno es un formulario HTML llamado *introduce.htm* el cual captura la matricula del alumno y su password, y envía los datos al segundo elemento que es la página PHP llamada *consulta.php*, la cual se conecta a la base de datos, verifica la existencia de la matricula y valida el password. Si existe y el password es el del

alumno, se despliega la información referente a éste, si no encuentra ni la matricula, ni el password manda un mensaje de error.

5.9 FORMULARIO introduce.htm

A continuación se muestra el código de *introduce.htm*, el cual es HTML puro.

```
<html>
<head>
<title>introduce tu matricula</title>
</head>
<body>
<center>
<form name="form1" method="get" action="consulta.php">
<hr size="5">
<p>
<font size="4">introduce tu matricula y password</font>
</p>
<p>
matricula.
<input name="matricula" type="text" size="15" maxlength="9">
<br>
password
<input name="password" type="password" id="password" size="15">
</p>
<input type="submit" name="submit" value="ok">
<input name="borrar" type="reset" id="borrar" value="borrar">
<hr size="5">
</form>
</center>
</body>
</html>
```

Esto da por resultado un pequeño formulario, el cual envía la información con el método GET a la pagina *consulta.php*. Este formulario contiene dos cajas de texto, llamadas *matricula* y *password*, la primera captura la matricula del alumno y la segunda el password, la ultima tiene la propiedad "password" la cual despliega asteriscos cuando se escribe en ella. Por último, contiene dos botones uno de envío y otro de limpieza.

Lo anterior se ve así:

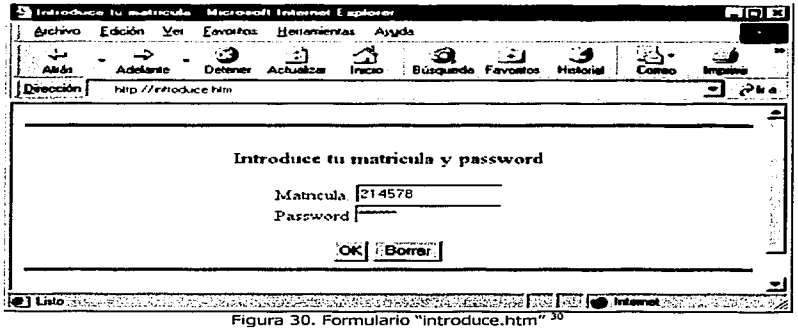

Figura 30. Formulario "introduce.htm" [30]

5.10 APLICACIÓN consulta.php

Enviados los datos llegan a *consulta.php*, la cual se conecta al servidor MySQL y busca la base de datos llamada *consulta_calif*.

Teniendo la conexión hecha, se procede a hacer la consulta para buscar la matricula y el password.

Si la matricula no existe o el password no coincide despliega el mensaje "no se encontró ningún registro con esa matricula, o su password es incorrecto, verifique su escritura". En caso de que la matrícula exista y el password también se procederá al despliegue de la información.

Ésta se desplegará en el orden descrito el la figura de la página siguiente:

[30] *Figura realizada por el autor.*

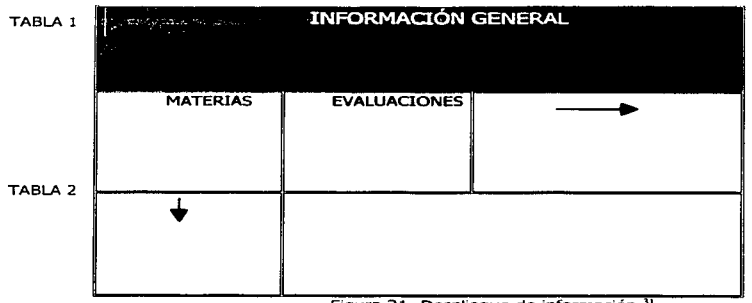

Figura 31. Despliegue de información [31]

El código de consulta_calif.php se muestra en la siguiente página:

[31] *Figura realizada por el autor.*

```
<html>
<head>
<title>consulta de calificaciones</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
</head>

<body>
<div align="center"><strong><em>consulta
de calificaciones ulsab</em></strong><br>
<br>
<?
$conex = mysql_connect("200.36.213.108","ulsab_usr","ulsab1910");
mysql_select_db("consulta_calif",$conex);

$res1 = mysql_query("select c.descripcion,a.matricula,a.nom_alumno,a.semestre,a.grupo from alumno
as a inner join carrera as c on a.carrera=c.carrera where matricula = '$matricula' and
password='$password' ");    $varaux = mysql_num_rows($res1);
if ($varaux == 0){
echo "<br><br>";
echo "<font size=3 face=arial, helvetica, sans-serif>no se encontro ningun registro con esa
matricula, o su password es incorrecto<br>verifique su escritura</font>";
mysql_close($conex);
}
else{
$datos = mysql_fetch_array($res1);
echo "<br><br>";
echo "<table width=75% border=1 bordercolor=#000000><tr><td colspan=4
align=center><b>$datos[0]<b></td></tr>";
echo "<tr><td align=right>matricula:</td><td>$datos[1]</td><td
align=right>nombre:</td><td>$datos[2]</td></tr>";
echo "<tr><td align=right>semestre:</td><td>$datos[3]</td><td
align=right>grupo:</td><td>$datos[4]</td></tr></table>";

$q2 = "select m.nom_materia, m.clv_materia from materia as m inner join cursando as c on
m.clv_materia = c.clv_materia where c.matricula = '$matricula' group by m.clv_materia";
            $res2 = mysql_query($q2);
            $num_mat = mysql_num_rows($res2);
            echo "<table width=75% border=1 cellspacing=0 bordercolor=#000000>";
            for ($a=0; $a<$num_mat ;$a++){
            echo "<tr>";
            $materias =mysql_fetch_array($res2);
            $clave_mat = $materias[1];
            echo "<td>$materias[0]</td>";
            $res3 = mysql_query("select no_evaluacion,calificacion from evaluacion where
clv_materia = '$clave_mat' and matricula = '$matricula' order by no_evaluacion");
            $num_eva =mysql_num_rows($res3);
             for ($b=0;$b<$num_eva;$b++){
                $evalu = mysql_fetch_array($res3);
                    echo "<td>";
                    echo "evaluacion: <b>",$evalu[0],"</b>";
                    echo "<br>";
                    echo "calif: <b>",$evalu[1],"</b>";
                    echo "</td>";
                    }
            echo "</tr>";
            }
            echo "</table>";
            mysql_close($conex);
}
?>
</div>
</body>
</html>
```

Por un principio como se platicó anteriormente, el script realiza una conexión con el servidor MySQL con la función *mysql_connect()* como se ve en la línea 12, dejando como identificador de conexión a la variable *$conex*, luego selecciona la base de datos llamada *consulta_calif* con la función *mysql_select_db()* en la línea 13.

En la línea 15 se ejecuta la primer consulta, la cual buscará los registros con los campos matricula y password de valor igual a las variables $matricula y $password que como anteriormente se vio provienen del formulario *introduce.htm.*

En la línea 17 a la variable $varaux se le asigna el número de registros arrojados por la consulta anterior que de encontrar la matricula y el password sería 1 y de no encontrar ningún registro seria 0. Esta variable se utiliza en la línea 18 donde está la condición if que en caso de que la consulta no arroje registros mostrara un mensaje de error. En caso contrario desplegará la información.

La información como se vio anteriormente, se desplegarán dos tablas, la de información general y la de materias y evaluaciones.

Información general

La primer tabla (línea 27) es la de información general, la cual se rellenará con el resultado de la primer consulta (línea 15) y contará con la carrera, matricula, nombre, semestre y grupo del alumno. Los datos serán desplegados usando la función *mysql_fetch_array()* (línea 25) que como ya se vio devuelve el resultado de la consulta por cada tupla en forma de arreglo, el cual se asigno a la variable $datos.

Materias y evaluaciones

La segunda tabla es la de materias y evaluaciones (línea 38), esta es un poco más compleja, ya que es dinámica. Básicamente crece por renglones según las materias que

curse el alumno y a su vez cada renglón se divide en celdas por cada evaluación que tenga de la materia.
Para controlar el número de renglones y qué va a contener cada uno se hace una consulta (línea 34), la cual arrojará las claves de las materias y sus nombres.

La función *mysql_num_rows()* (línea 37) arrojará el número de materias que lleva el alumno, este número será asignado a la variable $num_mat, la cual será usada en el ciclo for (línea 39 a 56) que controlará los renglones de la tabla.

Dentro de este ciclo, se ejecutará un mysql_fetch_array() para avanzar de renglón en el resultado de la segunda consulta (línea 34) y asignar los valores al arreglo $materias.

En cada renglón se desplegará el nombre de la materia (línea 43), y se asignará la clave de materia (línea 42) a una variable llamada $clave_mat. Esta servirá para realizar una tercer consulta (línea 44), la cual arrojará las evaluaciones de la respectiva materia. También existe otro ciclo for dentro del anterior, el cual va a generar las celdas de las evaluaciones correspondientes a la materia del renglón. Este ciclo for es controlado por la variable $num_eva, que sale de la función *mysql_num_rows()* en la línea 46. Dentro de este ultimo ciclo se ejecuta la función mysql_fetch_array(), para avanzar de tupla y desplegarla en un arreglo llamado $evalu (línea 48), también genera una celda en la cual mostrará el número de la evaluación y su calificación.

Por último, se cierra la conexión con MySQL con la función mysql_close() en la linea 58.

Un ejemplo de lo anterior se vería de la siguiente forma:

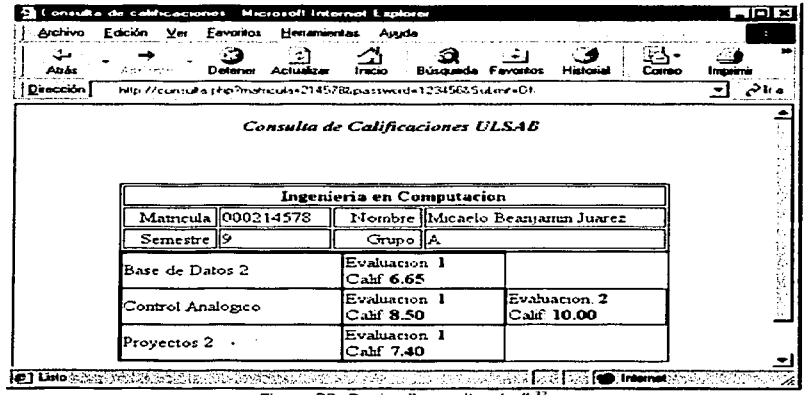

Figura 32. Pagina "consulta.php" [32]

Obsérvese que el alumno de ejemplo cursa tres materias, en las cuales en dos tiene una evaluación y en una tiene dos. En caso de que en todas las materia tuviera una evaluación sólo desplegaría una celda de evaluación.

5.11 NOTAS FINALES DEL CASO PRACTICO

Existiendo la necesidad de probar la aplicación antes de la implementación final, se buscaron las dos entidades esenciales para hacer funcionar la aplicación. Las cuales son un servidor web con soporte para PHP y un servidor MySQL, ambos con direcciones IP publicas. Teniendo en cuenta que los dos servicios también pueden estar en un solo servidor.

Las pruebas del proyecto fueron realizadas instalando el servidor MySQL en un servidor de las características siguientes: Doble procesador Pentium III a 900Hhz, 512 Mb en memoria

[32] *Figura realizada por el autor.*

RAM, utilizando el sistema operativo Microsoft Windows 2000 Advanced Server y contando con acceso a Internet dedicado con dirección IP publica.

En cuanto a la aplicación web esta se hospedo en un servicio de hospedaje con soporte para PHP de paga en un servidor remoto. En el cual se monto la aplicación vía FTP.

Teniendo ya una dirección IP publica tanto para el servidor web de la aplicación como para el servidor MySQL, en el este ultimo se dio de alta un usuario, con el nombre, contraseña y host, el cual es la dirección IP desde donde se van a realizar las peticiones o sea el servidor web de la aplicación PHP.

Ya generado un usuario en el servidor MySQL, se le agregaron los parámetros del usuario y la dirección IP del servidor MySQL a la conexión en el script PHP.

Los datos de prueba se introdujeron manualmente mediante instrucciones SQL en la base de datos.

Obteniendo con todo esto como resultado una conexión exitosa a la Base de Datos y el despliegue de información esperado.

CONCLUSIÓN

Desde sus orígenes, Internet ha ido evolucionando, así también sus servicios. En los últimos 10 años desde la creación del World Wide Web, esa evolución ha sido acelerada considerablemente gracias a la estandarización de tecnologías, y la ayuda colectiva de investigadores y programadores de todo el mundo.
También se ha generado una comunidad a favor de el software libre de código abierto, esto ultimo hace que el software esté en constante evolución, lo cual es favorable para el mundo de la informática.

PHP y MySQL son dos claros ejemplos de software libre, los cuales se han difundido en todo el mundo. La documentación de ambos están en aumento lo cual está haciendo cada vez más fácil iniciar en el desarrollo de aplicaciones web con estas herramientas.

PHP en mancuerna con MySQL son una muy buena opción en la creación de aplicaciones Web ya que son 100% compatibles entre sí y de fácil implementación. Además mostraron gran velocidad de respuesta, en la generación de páginas web dinámicas. Obteniendo aplicaciones web ligeras y practicas.

BIBLIOGRAFÍA.

FÁBREGA, Pedro Pablo, ***Php4 Serie Práctica,*** México, Prentice-Hall , 2000, 341 p.

GIL, Rubio, Fco. Javier, ***Creación de sitios web con PHP 4,*** Madrid, Mc-Graw-Hill Interamericana de España, 2001, 547 p.

McFEDRIES, Paul, ***Creando una página web con HTML,*** México, 2ª ed,1997, Prentice Hall Hispanoamericana , 328 p.

RAYA, José Luis, ***Redes Locales y TCP / IP ,*** España, Ra-Ma, 1995, 181 p.

RINEHART, Martín ***Desarrollo de Bases de Datos en Java*** España, Mc Graw Hill-Interamericana de España, 1998, 814 p.

OTRAS FUENTES

ÁLVAREZ, Miguel Ángel, "Programación de páginas web con PHP", ***Webmaster,*** año1, núm. 5.

BILKE, Petra, "PHP y MYSQL, páginas web dinámicas", ***PC-Cuadernos técnicos,*** año1, núm. 4.

Internet: http://www.php.net

Internet: http://www.desarrolloweb.com

Internet: http://www.mysql.org

Internet: http://www.webestilo.com

Internet: http://www.ciberteca.net/webmaster/php

Internet: http://www.mysql-hispano.org

Internet: http://www.interhelp.org/historia00.html

Internet: http://www.aunmas.com/future/internet_historia