UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
PLANTEL ARAGÓN

ACTIVE SERVER PAGE Y JAVASCRIPT COMO HERRAMIENTAS ÚTILES PARA LA GESTIÓN DE BASES DE DATOS EN INTERNET

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO EN COMPUTACIÓN

P R E S E N T A :

FORTINO GERARDO PEDRAZA GARCÍA

ASESOR:
ING. CÉSAR FRANCISCO GERMÁN ROSAS

SEPTIEMBRE 2003

Agradecimientos

A ***Dios nuestro Señor*** por iluminarme siempre en mi camino.

A ***mi Madre (Neini)*** por que siempre me proteges y me guías desde donde estás, por haberme encaminado en está hermosa carrera y por llevarme por el buen camino. MIL GRACIAS por haberme hecho un hombre de bien.

A ***mi Padre (Teiti)*** por apoyarme e impulsarme siempre en mis proyectos, por haberme dado la oportunidad con mucho esfuerzo de estudiar una carrera para que fuera un hombre de bien. Éste triunfo verdaderamente es tuyo. MIL GRACIAS.

A ***Patricia*** mi hermana, ***Alfonso*** su esposo y ***Melanie*** su hija por apoyarme mucho cuando Neini nos faltó y por todas las experiencias buenas y malas que hemos vivido juntos.

A ***Verónica*** mi hermana, ***Juan Carlos*** su esposo, ***Carlitos*** y ***Veritos*** sus hijos por estar siempre conmigo y apoyarme en mi carrera haciendo trabajos y programas juntos y sobre todo por la ayuda que me brindaron para hacer éste trabajo.

A ***Luchita*** mi suegra, ***Karla***, ***Fer***, ***Jonathan***, ***Luz*** y ***Nacho*** por dejarme ser parte de ustedes, por brindarme siempre su apoyo y sobre todo por enseñarme que la vida puede ser divertida y diferente.

A ***los Hermanos Zavala*** por brindarme su amistad y su apoyo y sobre todo por que siempre me acompañaron musicalmente para hacer éste trabajo. En especial a ***Eugenio*** e ***Isabel*** por sus consejos, a ***Javier*** por todo lo que me ha dado y a ***Angelita*** por brindarme un poco de su sabiduría y cariño. ***Carlos***, ***Magdalena*** y ***Miguelito*** por su entrañable amistad.

A ***Martha***, ***Marce***, ***Ale***, ***Juan Carlos***, ***Vero*** y ***Yola*** por su amistad y por todas esas cosas maravillosas que hemos compartido juntos.

A mi asesor de tesis el ***Ing. César Germán Rosas*** por su ayuda y consejos para hacer éste trabajo.

Y Finalmente al ser más importante de mi vida, mi querida y linda ***esposa Katy*** por estar siempre a mi lado y compartir su vida conmigo, por impulsarme, apoyarme, ayudarme y acompañarme siempre en la realización de éste trabajo. Y por haberme dado la maravillosa oportunidad de demostrarte cuando TE AMO.

Indice

Introducción

Hoy en día la utilización de la World Wide Web para presentar y acumular datos se ha desarrollado mucho más allá de la sencilla presentación de páginas, y ya están muy lejanos los días en que los diseñadores web que necesitaban presentar los elementos de una colección creaban una página independiente para cada uno de ellos, además de que dichas páginas eran difíciles de organizar inicialmente y era casi imposible mantenerlas actualizadas con el paso del tiempo.

Cada vez más el volumen y la estructura de los datos que se presentan en la web permite su almacenamiento y organización en bases de datos, además de que se considera que la gama de sitios que pueden beneficiarse de este enfoque es asombrosa. Sin embargo, es sorprendente que todavía vemos muy poco uso de lo que el web es capaz de hacer, por ejemplo: En proveer acceso interactivo a información sumamente importante, encuestas de mercadotecnia, levantado de pedidos y principalmente la actualización de datos a través del navegador y servicios de Web. Finalmente entre otras palabras las facilidades de las bases de datos pero accedidas remotamente. En nuestros días, la situación es lentamente cambiante y la mayor parte de las empresas prefieren que sus páginas sean más dinámicas con acceso a Bases de datos en vez de que sus sitios sean solo mas que folletos corporativos.

Algunas ventajas del uso de las bases de datos en Internet son:

- La base de datos es mucho más fácil de mantener que todas las páginas web individuales.
- El empleo de una base de datos facilita la búsqueda de los elementos deseados.
- Una base de datos facilita la presentación de los mismos datos de distintas formas: por categoría, por descripción, por edad o por cualquier otro campo de la base de datos.

El atractivo de difundir o recoger información mediante páginas web es universal, no importa si su empresa es grande o pequeña, si es algo personal, gubernamental, educativo o personal, o si tiene ánimo de lucro o no lo tiene, se aplican las mismas técnicas tanto si los elementos son antigüedades, coches usados, sellos raros, obras de arte, casas en venta o niños perdidos. Si se piensa crear una página web estática para cada uno de un montón de elementos resultaría tedioso, propenso a errores y difícil de organizar y mantener, es por eso que las páginas web con bases de datos son ideales para prevenir todas estas dificultades.

El presente trabajo, tratará de mostrar cómo se implementa una base de datos en Internet. Hoy en día existen diversas tecnologías que nos ayudan a la implementación de una base de datos para web, en nuestro caso nos basaremos en las tecnologías que Microsoft® posee para ello, es decir se implementará una base de datos de ejemplo en SQL Server 7.0® y para la gestión de la base de datos utilizaremos las Active Server Pages® (ASP's).

Generalmente cuando escribimos scripts para que nuestras páginas sean dinámicas o para la gestión de datos ya sea para que nos envíen información o para inscribirnos a alguna cuenta de correo, es mucho más sencillo llevar a cabo la validación de los datos que enviamos del lado del cliente y no esperar a que el servidor valide los datos que le enviamos, para ello aquí se explicará como podemos validar esos datos con ayuda de Javascript.

Finalmente para llevar a cabo todo esto, es necesario conocer ciertos conceptos relacionados con las bases de datos, la manipulación de información en Internet, los scripts, entre otras cosas, es por esto que el presente trabajo se divide de la siguiente manera, para la explicación de dichos conceptos:

- **Fundamentos de bases de datos**, proporciona los conocimientos de base claves sobre el diseño y utilización de bases de datos.
- **Javascript como lenguaje de validación de datos**, aquí conoceremos los fundamentos de Javascript que nos ayudarán a darnos cuenta de la potencialidad que posee para la validación de información.
- **Arquitectura Cliente/Servidor**, en esta parte se explicará como funciona esta arquitectura y su relación con Internet.
- **Introducción al Server-Side Scripting**, es una breve introducción al funcionamiento de los scripts (aplicaciones) que corren del lado del servidor.
- **Conceptos generales de ASP**, donde podremos conocer los fundamentos de esta tecnología de Microsoft, para la creación de server-side scripts.
- **Un caso práctico**, que nos ayudará a ejemplificar de manera práctica todos los conceptos antes mencionados.

1.Fundamentos de Bases de Datos.

1.1 ¿Que es una base de datos?

Una Base de Datos (cuya abreviación es BD, en inglés DB, *database*), es una entidad en la que es posible almacenar datos de una manera estructurada y con la menor redundancia posible. Estos datos pueden ser utilizados por ciertos programas y por diversos usuarios. La noción de una base de datos es generalmente comparada con la de una red, con el fin de que la información sea común, de ahí el nombre de base. En la ***imagen 1.1-1*** podemos observar la representación de un sistema básico de bases de datos, dónde los clientes hacen peticiones de información a un servidor que tiene una base de datos, como podemos observar el servidor es el intermediario entre la base de datos y los clientes.

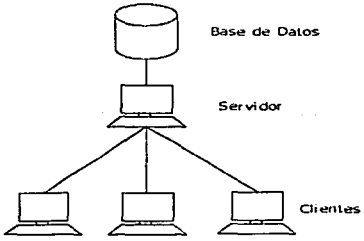

Imagen 1.1-1

1.2 ¿Utilidad de una base de datos?

Una base de datos permite agrupar datos dentro de una sola entidad o archivo y es por eso que los datos informáticos son cada vez más populares que estar almacenando cientos y cientos de carpetas con papeles en un archivero.

Una base de datos puede ser local, es decir utilizable solo por un usuario o bien puede estar en un servidor para poder ser accedida a través de la red por diversos usuarios.

Una de las principales ventajas de utilizar Bases de Datos, es la posibilidad de que varios usuarios puedan acceder a los datos casi de manera simultánea.

1.3 La gestión de Bases de Datos

A fin de poder controlar los datos de los usuarios, la necesidad de un sistema de gestión se hace presente. La gestión de una base de datos se lleva a cabo gracias a un sistema llamado SGBD (Sistema Gestor de Bases de Datos) o del inglés DBMS (Database management systeme). El DBMS es un conjunto de servicios (Softwares de Aplicación) que permiten administrar las bases de datos, es decir:

- Permiten el acceso a los datos de manera simple.
- Autorizar el acceso a la información a múltiples usuarios.
- Manipular los datos dentro de la base de datos (insertar, borrar, modificar, etc.).

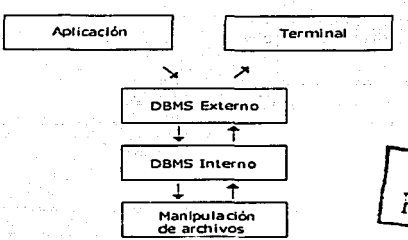

Imagen 1.3-1

Como observamos en la ***imágen 1.3.1*** el DBMS se puede componer de tres sub-sistemas, dichos sub-sistemas son:

- El sistema de gestión de archivos: que permite almacenar la información en un soporte físico de datos.
- El DBMS interno: Administra el orden de la información.
- El DBMS externo: Representa la interfase con el usuario.

1.4 Los principales DBMS

Los principales sistemas de gestión de bases de datos son los siguientes:

- Borland Paradox®
- Claris Filemaker®
- IBM DB2®
- Microsoft SQL Server®
- Microsoft Access®
- Microsoft FoxPro®
- Oracle®
- Sybase®
- MySQL®
- PostgreSQL®
- MSQL®
- SQL Server 11®

1.5 Los niveles de datos

Los niveles ANSI/SPARC

La arquitectura ANSI/SAPARC data de 1975, la cual define los niveles de abstracción para un sistema gestor de Bases de Datos:

- **Nivel Interno (o Físico):** Define la manera en la que los datos son almacenados y los métodos para acceder a ellos.
- **Nivel Conceptual:** Llamado también MCD (modelo conceptual de datos) o MLD (modelo lógico de datos). Define el orden de la información de la base de datos.
- **Nivel Externo:** Define las vistas de los usuarios.

1.6 Características de un DBMS

La arquitectura a 3 niveles definida por el standard ANSI/SPARC permite tener una independencia entre los datos y la forma de tratarlos. De una manera general un DBMS debe poseer las siguientes características:

Independencia física: El nivel físico puede ser modificado independientemente del nivel conceptual, esto quiere decir que todos los aspectos materiales de la base de datos no son vistos por los usuarios, es decir que sólo se trata de una estructura transparente de representación de la información.

Independencia lógica: El nivel conceptual debe ser capaz de modificarse sin tener que tocar al nivel físico, es decir, que el administrador de la base de datos pueda modificar la BD sin molestar a los usuarios.

Rapidez de acceso: El sistema debe ser capaz de responder a las peticiones de los usuarios lo más rápido posible, esto implica algoritmos de búsqueda que sean rápidos.

Administración centralizada: El DBMS debe permitir al administrador la manipulación de datos, inserción de elementos y la verificación de la integridad de la base de datos de manera centralizada.

Limitación de la redundancia: El DBMS debe ser capaz de evitar en la medida de los posible las informaciones redundantes, con el fin de evitar por un lado el desperdicio de espacio de memoria y por otro lado los errores.

Verificación de la integridad: Los datos deben ser coherentes entre ellos ya que los elementos que hacen referencia hacia otros deben estar siempre presentes.

Compartición de información: El DBMS debe permitir el acceso simultáneo a la base de datos por diversos usuarios.

Seguridad de la información: El DBMS debe tener mecanismos de administración de los derechos de acceso a los datos según el usuario que intente acceder a la información.

1.7 Los modelos de los DBMS

Los diferentes modelos de bases de datos

Las bases de datos aparecieron a finales de los años 60, en una época donde la necesidad de un sistema de gestión de información se hizo presente. Existen diversos modelos de DBMS que se diferencian principalmente según la representación de los datos que contienen:

El Modelo Jerárquico: Los datos son clasificados jerárquicamente, conforme a una arborescencia descendente. Este modelo utiliza apuntadores entre los diferentes registros de datos. Es considerado como el primer modelo de los DBMS.

En la ***Imagen 1.7-1*** podemos observar la representación gráfica de este modelo.

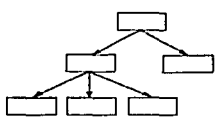

Imagen 1.7-1

El Modelo de Red: Como el modelo jerárquico éste modelo utiliza apuntadores entre los diversos registro de la BD. Sin embargo la estructura no es mas arborescente descendentemente, sino entre los elementos que están en un mismo nivel. En la ***Imagen 1.7-2*** podemos observar la representación gráfica de este modelo.

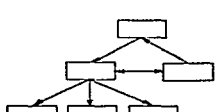

Imagen 1.7-2

El Modelo Relacional: Los datos son almacenados dentro de tablas de 2 dimensiones (renglones y columnas). La manipulación de estos datos se hace según la teoría matemática de relaciones. En la ***Imagen 1.7-3*** podemos observar la representación gráfica de este modelo.

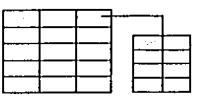

Imagen 1.7-3

El Modelo deductivo: Los datos son representados en forma de tabla, pero su manipulación se hace a través del cálculo de predicados.

El Modelo Orientado a Objetos: Los datos son almacenados en forma de objetos, es decir en estructuras llamadas clases que representan a los datos miembro. Los campos son instancias de estas clases. En la ***Imagen 1.7-4*** podemos observar la representación gráfica de este modelo.

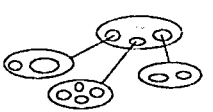

Imagen 1.7-4

Hoy en día las bases de datos relacionales son las bases de datos mas utilizadas (Según los propios desarrolladores de bases de datos consideran que tres cuartas partes de todas las bases de datos que existen en el mundo siguen el modelo relacional). Por lo que a continuación se presenta un apartado dedicado especialmente a este modelo de bases de datos.

1.8 Introducción al Modelo relacional

El modelo relacional está basado en la organización de los datos en forma de tablas. La manipulación de los datos se hace según el concepto matemático de relación de la teoría de conjuntos, es decir a través del álgebra relacional. El álgebra relacional fue inventada en 1970 por E.F Codd, director de investigación del centro IBM de San José. El álgebra relacional se compone de un conjunto de operaciones formales sobre las relaciones. Las operaciones relacionales permiten crear una nueva relación (tabla) a partir de operaciones elementales sobre otras tablas (por ejemplo la unión, intersección e incluso la diferencia).

La Teoría de conjuntos tiene dos nociones: La noción de dominio y la noción de producto cartesiano.

La noción de dominio

Un dominio es un conjunto finito o infinito de valores. Se puede representar por una lista de elementos o bien por una condición necesaria o suficiente de pertenencia.

El dominio de los boléanos: {0,1}

El dominio de los dedos de la mano: {pulgar, indice, medio, anular y meñique}

La noción de producto cartesiano

La manipulación de los datos (Selección de Valores) se hace según la noción matemática de producto cartesiano. El producto cartesiano de un conjunto de dominios D_i, denotados por $D_i*D_i*D_i*...*D_n$ es el conjunto de n-tuplas[1] $<V_1,V_2,...,V_n>$ tal que V_i pertenece a D_i

[1] Una tupla es un renglón de una tabla, por lo que un conjunto de tuplas no es mas que un conjunto de renglones.

El Modelado relacional

El modelado relacional permite representar las **relaciones** con la ayuda de tablas (de 2 dimensiones), en donde cada columna tiene un identificador que representa un dominio y un renglón de la tabla representa una entidad y cada uno de los casos representa uno de sus atributos.

Llamamos **atributos** al nombre de las columnas que representan los elementos que constituyen a una entidad. Un atributo (una columna) esta conformada por un nombre y por un dominio de definición, es decir por el conjunto de valores que puede tener.

Llamamos **tupla** a un renglón de la tabla.

La entidad automóvil puede ser representada por ejemplo de la siguiente manera:

- La marca.
- El modelo.
- La serie.
- La placa.

Marca	Modelo	Serie	Placa
Renault	18	RL	4698 SJ 45
Peugeot	309	Chorus	5647 ABV 82
Ford	Escort	Match	8562 EV 23

Donde:
Marca, Modelo, Serie, Placa son los atributos.
Y los renglones son las Tuplas.

La **cardinalidad** de una relación es el número de tuplas que la conforman. En el ejemplo anterior la cardinalidad es igual a 4.

La **llave principal** de una relación es el atributo o conjunto de atributos que permiten identificar de una manera única a una tupla. En el ejemplo anterior el número de la placa es una llave principal dado que con tan solo conocer el número de placa podemos saber todo a cerca del automóvil que posee dicha placa.

La **llave foránea** es una llave que hace referencia a una llave que pertenece a otra tabla.

La descripción de una relación (de una tabla) por su atributos (nombre y dominio) se llama *esquema de una relación*. Por lo que llamamos al termino *esquema de una base de datos relacional* al conjunto de relaciones que la componen.

La manipulación de los elementos de una tabla se hace gracias a las operaciones sobre los conjuntos. Dichas operaciones de base se conocen como:

- Operaciones unarias.
- Operaciones de conjuntos.

Operaciones de Base

Una operación de base se define por el hecho de que no puede ser realizada por combinación de otras operaciones. Existen 5 operaciones elementales que se pueden clasificar en 2 categorias:

Las operaciones unarias: Consisten en eliminar los renglones o columnas de la tabla.

Las operaciones de conjuntos: Consisten en efectuar una recuperación de datos entre diversas tablas.

Operaciones Unarias

Proyección

El operador proyección consiste en crear una tabla a partir de otra con las columnas especificadas en la proyección.

$Proj_{x1, x2,..., xn}(R)$

Dónde $x1, x2,..., xn$ representan las columnas a obtener.

Restricción (o Selección)

El operador restricción consiste en crear una tabla a partir de otra con los renglones con los que una columna cumpla con ciertas propiedades.

$Select_Q(R)$.

Dónde $_Q$ representa la condición a realizar.

Operaciones de Conjuntos

Unión

La unión de dos tablas es la creación de una tabla que contiene el conjunto de tuplas de una tabla u otra (o las dos tablas). Las 2 tablas deben ser de un mismo esquema, es decir que los atributos de la tabla (columnas) deben de ser los mismos. La unión entre 2 tablas R1 y R2 se denota por:

Union(R1,R2)

Diferencia

La diferencia entre 2 tablas es la tabla que contiene el conjunto de tuplas que contiene una tabla menos la segunda. De igual forma las tablas deben de ser del mismo esquema. La Diferencia entre 2 tablas se denota por:

Minus(R1,R2)

Producto Cartesiano

El producto cartesiano de 2 tablas es la tabla que contiene la concatenación del conjunto de tuplas de una linea de una tabla a las de la otra tabla y esto para cada línea. Las 2 tablas no necesariamente deben ser del mismo esquema. Un producto cartesiano se denota por la siguiente notación:

Product(R1,R2)

Operaciones Derivadas

A partir de las operaciones de base se pueden llevar a cabo numerosas operaciones con la combinación de estas.

Intersección

La intesección entre 2 tablas es la tabla que contiene el conjunto de tuplas que pertenecen a las 2 tablas. Las 2 tablas deben de ser del mismo esquema, es decir, que los atributos de la tabla (columnas) deben ser los mismos. Y se denota por:

Inter(R1,R2)

Este operador puede ser descrito a través de los operadores de base:

R1- (R1 – R2)

División

La división de 2 tablas es la tabla que contiene el conjunto de tuplas_que concatena a cada tupla de una de las tablas que abastece a las tuplas que pertenecen a la otra tabla. Y se representa a través de:

Div(R1,R2)

Juntura (Join)

Se llama *Juntura según la condición Q* al conjunto de tuplas provenientes del producto cartesiano de 2 relaciones que cumplen con cierta condición que se expresa a través de comparadores <> = >= =<.

Denotamos a la juntura de la siguiente manera:

$Join_{Q(R1,R2)}$

Se pueden definir diferentes tipos de juntura según la condición Q:

- **La juntura-equivalente** es un Join cuya condición es una igualdad entre 2 tablas.

- **La juntura natural** es una juntura-equivalente sobre los atributos del mismo nombre asociado a una proyección.

Operaciones de Cálculo

Las operaciones de calculo no son operaciones derivadas, por lo que pueden ser consideradas dentro de las operaciones de base sin llegar a ser de este tipo. Sin embargo permiten realizar operaciones muy útiles (Algunas veces operaciones que son necesarias) que con los otros operadores de base no se podrían.

Operador Suma

Este operador permite realizar la suma acumulada de los valores que el atributo X toma de una lista de atributos llamados **atributos de reagrupamiento.** Los valores de X deben de ser evidentemente numéricos. Este operador se denota por:

$Sum_{x1,x2,...,xn}(R,X)$

Operador de Conteo

Este operador permite contar el número de líneas por las cuales el atributo hace parte de una línea de atributos llamados *atributos de reagrupamiento.* Este operador cuenta el número de tuplas de la columna. Se denota por:

$Count_{x1,x2,..., xn}(R,X)$

1.9 Introducción a los Sublenguajes de Datos (DSL)

Muchos sistemas manejadores de base de datos tienen un SUBLENGUAJE DE DATOS (Data Sub Language) que está constituido en dos partes:

Lenguaje de Definición de Datos *(Data Definition Language DDL)*

Lenguaje de Manipulación de Datos *(Data Manipulation Language DML)*

El lenguaje de definición de datos está constituido por las instrucciones que permiten **crear/mantener:**

Las estructuras de almacenamiento de los archivos de datos. La estructura de las estrategias de acceso(Heap, Indice, Hash, etc.)

El lenguaje de manipulación está compuesto por las instrucciones que permiten realizar ALTAS, BAJAS, CAMBIOS y OBTENCION DE INFORMACION de los archivos de datos.

SQL

Dentro de los sublenguajes de datos más difundidos y de amplio uso tenemos el SQL (Structured Query Language).

DDL del SQL

Como ejemplo del DDL del SQL veremos cómo se crea una tabla. Supongamos que deseamos crear una base de datos simple(Compuesta por una sola tabla), para llevar el control de calificaciones de los exámenes de alumnos de una materia, suponiendo que los atributos importantes son:

MATRICULA, NOMBRE, PARCIAL 1, PARCIAL 2, PARCIAL 3, FINAL

La forma de crear esta tabla sería:

```
CREATE TABLE CALIF(MATRICULA INTEGER NOT NULL, NOMBRE CHAR(35),
PARCIAL1 SMALLINT, PARCIAL2 SMALLINT, PARCIAL3 SMALLINT, FINAL SMALLINT);
```

Para soportar el concepto de llave (No duplicados) es necesario crear un índice único sobre los atributos llave (que fueron especificados con NOT NULL):

```
CREATE UNIQUE INDEX CALIX ON CALIF (MATRICULA);
```

Supongamos que ya se cargaron datos, un ejemplo de contenido es dado en la tabla.

MATRICULA	NOMBRE	PARCIAL1	PARCIAL2	PARCIAL3	FINAL
331243	JUAN MARTINEZ	8	8	6	7
335467	PEDRO LOPEZ	10	10	6	9
337890	MARIA ALONSO	9	9	6	10
558967	FRIDA GUTIERREZ	7	8	7	10
578990	DIANA CAMACHO	6	7	8	9
654321	LUIS PEREZ	8	5	9	9
723445	FRANCIS RIQUELME	8	6	10	8

Tabla para control de calificaciones

DML del SQL

Veremos ahora ejemplos del DML:

Inserción

INSERT INTO CALIF VALUES (331232, 'LUIS PEREZ MARTINEZ', 1,1,1,1);

En este ejemplo lo que se hace es insertar un nuevo registro en la tabla CALIF con los valores siguientes: Matricula=331232, Nombre=LUIS PEREZ, Parcial1=1, Parcial2=1, Parcial3=2 y Final=1.

Borrado

DELETE FROM CALIF WHERE MATRICULA=331289;

En este ejemplo lo que se hace es borrar el registro de la tabla CALIF cuya matricula sea igual a 331289.

Actualización

UPDATE CALIF SET PARCIAL1=PARCIAL1+1;

En este ejemplo lo que se hace es actualizar el campo PARCIAL1 de todos los registros de la tabla CALIF.

Consulta

SELECT MATRICULA, NOMBRE FROM CALIF WHERE PARCIAL1=PARCIAL2

En este ejemplo lo que se hace es mostrar la matricula y el nombre de los registros de la tabla CALIF cuyo PARCIAL1 sea igual a PARCIAL2.

1.10 Ciclo de vida del desarrollo de un sistema de Base de Datos

El ciclo de vida clásico de un sistema de base de datos consiste en:

Análisis, Diseño, Desarrollo, Implementación y Mantenimiento.

A continuación veremos en que consiste cada una de estas etapas.

Análisis

Los resultados de esta etapa son:

- Obtención de un **Modelo Conceptual de Datos** que describe a la información utilizada dentro de la organización pero no en términos computacionales. Esta etapa de análisis de datos será considerada mas adelante. El modelo conceptual de datos provee un contexto del que se obtienen especificaciones detalladas de diseño.
- Obtención de un **Modelo Conceptual de procesos** que describe las funciones de la organización en términos de eventos (ventas, pagos, etc) y el proceso que se lleva a cabo para desarrollar o llevar a cabo dichos eventos.

Diseño

Los resultados de esta etapa son:

Obtención de un Modelo Lógico de Datos que es una descripción de los datos a ser almacenados en la Base de Datos, usando las convenciones prescritas por el DBMS a utilizar. Algunas veces esto se conoce o se hace referencia como un ESQUEMA.

Obtención de una Especificación de Sistema que describe a detalle lo que el sistema debe hacer. Ahora sí se hace referencia a procesos computacionales; pero probablemente en términos de mensajes de entrada (INPUT) y mensajes de salida (OUTPUT), cuando ocurre una descripción, error, el efecto de seleccionar un elemento del menú o cualquier comando dentro del sistema. Los módulos del programa son definidos en términos de pantallas.

Desarrollo

Aquí es donde se debe decidir a cerca del almacenamiento físico de los datos. Así como la elección del programa de desarrollo a utilizar, introducir el código a dicho programa, pruebas y depuración del sistema, generación de reportes y algunas otras salidas de información.

Implementación

Aquí es donde finalmente se obtiene el sistema final, es decir se pone en marcha el sistema para su uso normal, también se dice que aquí se pone en marcha el trabajo realizado en las tres etapas anteriores. También se lleva a cabo el entrenamiento del personal que utilizará el sistema, probablemente introducción de nuevos hábitos de trabajo, puede ser también que se trabaje a la par entre el nuevo sistema y tal vez alguno anterior ya antes realizado en caso de que nuestro nuevo sistema falle por alguna razón no contemplada.

Mantenimiento

Una vez que el sistema ha sido implementado, generalmente requiere de un mantenimiento para un funcionamiento optimo, además de corregir los errores no previstos o correcciones por cambios en los requerimientos y el simple mantenimiento de los datos tanto físicos como lógicos y el respaldo mismo de la información.

1.11 Componentes de un Sistema de Base de Datos

Involucra los siguientes componentes:

- Datos
- Hardware
- Software
- Usuarios

Datos

Los datos dentro de una base de datos están integrados y son compartidos:

INTEGRADOS Puesto que la base de datos es la unificación de varios archivos con redundancia parcial o totalmente eliminada.

COMPARTIDOS Esto implica que los datos pueden ser accedidos concurrentemente por diferentes usuarios.

Hardware

Consta básicamente de unidades de almacenamiento como discos duros, discos compactos, cintas magnéticas y las computadoras donde se encuentran estos dispositivos.

Software

Entre la base de datos física y los usuarios existe una capa de Software denominada SISTEMA MANEJADOR DE BASE DE DATOS(SMBD ó DBMS).

Todos los requerimientos de acceso a la base de datos son manejados por el SMBD.

Por otro lado también son parte del software, el conjunto de programas de nuestro sistema de información, así como los compiladores e interpretes que utilizamos para el desarrollo de nuestro sistema de información.

Usuarios

Hay 3 tipos de Usuarios:

- **Programador de Aplicaciones** Se encarga de escribir programas para el manejo de la Base de Datos usando un lenguaje de alto nivel.
- **Usuario Final** Es quien utiliza el sistema de información ya terminado, este tipo de usuario no necesariamente debe ser un experto en computación para poder utilizar dicho sistema.
- **Administrador de la base de datos (DBA)** Es el responsable de definir políticas de acceso a la Base de Datos.

1.12 Ventajas de Utilizar una Base de Datos

- **Compactas.** No se necesitan voluminosos archivos de papel.
- **Velocidad.** La velocidad de operación es mayor a la que se tiene con un sistema manual.
- **Menos Tedio**
- **Actualización.** La información se puede mantener más fácilmente actualizada.

1.13 Ventajas de Tener el Control Centralizado de una Base de Datos

Se puede reducir la redundancia. Se evita la inconsistencia. Los datos pueden ser compartidos. Se pueden reforzar los estándares. Se tiene el control del acceso. La integridad puede ser mantenida. Se pueden balancear requerimientos conflictivos.

Inconsistencia

Cuando dos instancias del mismo elemento no tienen valores iguales.

Si hay dos registros para el alumno con matricula 331540 (en diferentes archivos), los atributos iguales deben tener los mismos valores.

Falta de Integridad

Se da la falta de integridad cuando una instancia de un elemento tiene valores raros:

Que el número de horas trabajadas a la semana por un empleado sea de 400.

Independencia de Datos

Se dice que un sistema tiene INDEPENDENCIA DE DATOS:

Si los programas de aplicación no tienen que ser modificados al cambiar: la estructura de almacenamiento y/o la estrategia de acceso.

Dicho de otro modo es:

LA INMUNIDAD DE LAS APLICACIONES A LOS CAMBIOS EN LA ESTRUCTURA DE ALMACENAMIENTO Y/O LA ESTRATEGIA DE ACCESO.

1.14 Conceptos Adicionales

Campo Almacenado

Es la unidad de datos más pequeña que se encuentra almacenada.

Registro Almacenado

Es una colección de campos almacenados que están relacionados.

Archivo Almacenado

Es el conjunto de todas las ocurrencias de un registro almacenado.

Representación de Datos Numéricos

Se pueden almacenar como:

- Un String de Caracteres.
- Un Decimal Empacado.
- En Binario.

Representación de Datos Caracter

Se Pueden almacenar en ASCII, EBCDIC, etc.

Manejo de Objetos

Se pueden manejar como campos objeto que pueden ser Gráficas, Sonido, Hojas de Calculo, Textos, etc.

Codificación de Datos

Resulta útil en ocasiones almacenar los datos en forma codificada. En lugar de almacenar los nombres de los colores podríamos codificarlos para ahorrar espacio y facilidad de uso de acuerdo a la tabla:

Color	Número
Negro	0
Café	1
Rojo	2
Naranja	3
Amarillo	4
Verde	5
Azul	6
Violeta	7
Gris	8
Blanco	9

Tabla de Codificación de Colores

Materialización de Datos

Existen campos virtuales, debido a que no tienen equivalencia con un campo almacenado, y para poder ser accesados deben ser calculados. Un campo virtual llamado PROMEDIO que depende de otros datos, de forma que para ser accesado debe ser calculado primero.

2. JavaScript como lenguaje de validación de datos

2.1 El lenguaje JavaScript

JavaScript es un lenguaje de scripts[2] compacto basado en objetos (y no orientado a objetos). Originariamente era denominado *LiveScript*, y fue desarrollado por Netscape para su navegador *Netscape Navigator 2.0*. Fue éste el primer cliente en incorporarlo. Se ejecuta sobre 16 plataformas diferentes, incluyendo los entornos de Microsoft e incluso el MS Explorer lo incorpora en su versión 3.0 .

JavaScript permite la realización de aplicaciones de propósito general a través de Internet y aunque no está diseñado para el desarrollo de grandes aplicaciones es suficiente para la implementación de aplicaciones para Internet completas o interfaces WWW hacia otras más complejas.

Por ejemplo, una aplicación escrita en JavaScript puede ser incrustada en un documento HTML proporcionando un mecanismo para la detección y tratamiento de eventos, como clicks del ratón o validación de entradas realizadas en *formularios*.

Sin existir comunicación a través de la red una página HTML con JavaScript incrustado puede interpretar, y alertar al usuario con una ventana de diálogo, de que las entradas de los formularios no son válidas. O bien realizar algún tipo de acción como ejecutar un archivo de sonido, un applet de Java, etc. En otras palabras si habláramos en términos de la arquitectura Cliente/Servidor, JavaScript se ejecuta del lado del cliente. En el **capitulo 3** de este trabajo se abordará todo lo relacionado con esta arquitectura, para su mejor entendimiento.

[2] Un script es un conjunto de comandos que pueden ser ejecutados sin la interacción del usuario. También se se puede decir que un script es una lista de acciones o tareas a realizar.

2.2 JavaScript y Java

Las diferencias entre Java y JavaScript son notables pero también sus similitudes.

En primer lugar Java es un lenguaje de programación mientras que JavaScript es un lenguaje de scripts (como su nombre indica). Éste último es más sencillo de entender y usar que Java si no se tienen conocimientos previos de metodología de programación orientada a objetos. La mayor sencillez de JavaScript hace que sea interesante aprender éste último lenguaje como paso previo a adentrarse en el mundo de Java.

JavaScript es mucho más modesto pero precisamente por ello es más sencillo. Se basa en un modelo de instanciación de objetos muy simple para el que no es necesario tener conocimiento de conceptos tales como herencia y jerarquías.

Soporta un sistema en tiempo de ejecución basado en un pequeño número de tipos de datos (numérico, Boolean, y string) en el que ni siquiera es necesario declarar el tipo de variables. Sin embargo Java exige una gran rigidez en el tipo de datos utilizados y dispone de una amplia variedad de tipos básicos predefinidos, operadores y estructuras de control.

En Java uno de los principales bloques de programación son las clases a las que se asocian funciones específicas. Para utilizarlas es necesario instanciarlas en objetos. Los requerimientos de Java para declarar dichas clases, diseñar sus funciones, y encapsular tipos hacen que la programación en este lenguaje sea mucho más compleja que la realizada con JavaScript.

Otra diferencia importante es que Java es un lenguaje lo bastante potente como para desarrollar aplicaciones en cualquier ámbito. No es un lenguaje para programar en Internet, sino que se trata de un lenguaje de propósito general, con el cual se puede escribir desde un applet para una página Web (ésto es una pequeña aplicación escrita con un determinado formato que se ejecuta en un trozo de un documento HTML) hasta una aplicación que no tenga ninguna clase de conexión a Internet.

Los requerimientos también son diferentes; para programar en JavaScript sólo es necesario un editor de texto mientras que para programar en Java se necesita un compilador específico.

La complejidad de Java es semejante a la de un programa en C++ mientras que la de JavaScript es cercana a la de uno en dBase, Clipper o sh.

Por otra parte, la sintaxis de ambos lenguajes es muy similar sobre todo en lo que a estructuras de control de flujo se refiere. Existen además mecanismos de comunicación entre Java y JavaScript.

En definitiva, la principal ventaja de JavaScript es su simplicidad y su menor demanda de requisitos.

2.3 Relación entre JavaScript y Java:

JavaScript	Java
Interpretado (no compilado) en cliente.	Compilado en servidor antes de la ejecución en el cliente.
Basado en eventos y objetos. Usan objetos, pero no clases ni herencia.	Programación orientado a objetos. Los applets constan de clases objeto con herencia.
Código integrado en el código HTML.	Applets diferenciados del código HTML (accesibles desde las páginas HTML).
No es necesario declarar el tipo de las variables.	Necesario declarar los tipos.
Enlazado dinámico. Los objetos referenciados deben existir en tiempo de ejecución (lenguaje interpretado).	Enlazados estáticos. Los objetos referenciados deben existir en tiempo de compilación (lenguaje compilado).

2.4 JavaScript y CGI

CGI (the ***C**ommon **G**ateway **I**nterface*) es una interfaz entre programas de aplicación y servicios de información. Es decir, son un conjunto de reglas a cumplir tanto por parte del servidor como por parte del programa, pero se deja libertad al programador a la hora de escoger el lenguaje que considere mas adecuado para programar la aplicación. Un programa CGI puede ser escrito en cualquier lenguaje como: C/C++ , Fortran, PERL ,TCL, etc.

En JavaScript no existen restricciones a cumplir en el Servidor hasta el punto que ni siquiera es necesario que éste exista.

Por otra parte y al contrario que CGI, JavaScript únicamente depende del cliente y no del sistema operativo, sólo necesita un browser(Navegador) capaz de interpretarlo. Cualquier persona puede desarrollar aplicaciones escritas en JavaScript del mismo modo que realiza páginas HTML.

Esto no ocurre con aplicaciones CGI que necesitan la existencia de un servidor WWW para ser ejecutadas.

Con JavaScript todo el código es trasladado al cliente y no se necesita la comunicación a través de la red cada vez que se produce un evento, como se requería en CGI.

Por otro lado, JavaScript no es un lenguaje válido para desarrollar aplicaciones concurrentes y/o de acceso compartido.

2.5 Comparación entre JavaScript y CGI:

JavaScript	CGI
Es un lenguaje de programación	Es una interfaz. Da libertad de elección del lenguaje
No requiere un servidor de WWW	Exige la presencia de un servidor WWW
No requiere una red, funciona en local.	Requiere comunicación en red. No funciona en local
La aplicación reside en el cliente	La aplicación reside en el servidor
Requiere un cliente especial	Sirve cualquier cliente, por simple que sea
Pensado para aplicaciones monousuario	Permite desarrollo de aplicaciones distribuidas, acceso concurrente y/o compartido
Tiempo de desarrollo muy breve	Tiempo de desarrollo medio

2.6 JavaScript y HTML

Los programas en JavaSecript aparecen incrustados en los propios documentos HTML como si de HTML se tratara. Pueden integrarse de dos formas:

- Como programas propiamente dichos, combinando funciones y sentencias, con el mismo aspecto que tendría el código de cualquier otro lenguaje.
- Introduciendo manejadores de eventos JavaScript en etiquetas HTML.

En los siguientes temas se presentarán los dos mecanismos:

2.7 La etiqueta script

La manera más convencional en que aparece JavaScript en un documento es en forma de programa. Podemos empezar por mostrar unos breves scripts y ver como son implementados dentro de documentos HTML. Empezaremos con un pequeño programa que muestra un texto en un documento HTML.

```
<html>
<head>
   ¡Mi primer JavaScript!
</head>
<body>
   <br>
     Este es un documento HTML normal
   <br>
        <script language="JavaScript">
            document.write("Esto es JavaScript!")
        </script>
   <br>
   En HTML otra vez.
</body>
</html>
```

Este primer programa se limita a escribir en pantalla un determinado texto para lo que se emplea el código *document.write*. En este código, *document* es un objeto creado por el sistema que hace referencia al propio documento y *write* es uno de los métodos que proporciona para interactuar con él. El resultado de cargar este documento en un browser (navegador) que interprete JavaScript será la aparición de los dos textos, el escrito en JavaScript y el escrito en HTML, sin que el usuario sea consciente del proceso.

El resultado seria:

```
Este es un document HTML normal.
Esto es JavaScript!
En HTML otra vez
```

Este script no es muy útil pero sirve para mostrar el uso de la etiqueta <*SCRIPT*>. Puedes usar estas etiquetas en cualquier lugar del documento, tanto en la cabecera como en el cuerpo, aunque si se declaran funciones es más aconsejable hacerlo en la cabecera.

La etiqueta *<SCRIPT>* es una extensión de HTML en la que se encierra el texto que compone el código del programa JavaScript correspondiente de la manera siguiente:

```
<SCRIPT>
        Sentencias JavaScript...
</SCRIPT>
```

De esta manera el navegador que "entienda" JavaScript reconocerá el texto encerrado entre estas etiquetas como código JavaScript y no lo mostrará en la pantalla del cliente. Una cuestión importante a considerar es el mantenimiento de la compatibilidad con navegadores anteriores. Cualquier browser ignora las etiquetas desconocidas, por lo tanto, aquellos que no soporten JavaScript ignorarán el comienzo y el final del código del programa (encerrado entre las etiquetas *<SCRIPT>* y *</SCRIPT>*). Para que el resto del código también sea ignorado y no aparezca en la pantalla del cliente, lo encerraremos entre los símbolos de comentario HTML, <!-- y -->.

Los navegadores que, por el contrario si lo soporten, interpretarán el código encerrado entre las etiquetas SCRIPT e ignorará el principio de la línea en el script que comienza con la doble slash (//) o bien el encerrado entre "/*" y "*/", que son los símbolos de comentarios en este lenguaje. Un documento puede tener varias etiquetas SCRIPT, y cada una de ellas incluir sentencias JavaScript diferentes.

Si queremos dar la posibilidad de ejecutar un código alternativo a los browsers que no interpretan JavaScript, debemos utilizar las etiquetas *<NOSCRIPT></NOSCRIPT>*

Por ejemplo:

```
<SCRIPT>
<!-- Ocultación a browsers antiguos
document.write("Si ves esto, tu browser interpreta JavaScript")
// Fin de la ocultación -->
</SCRIPT>
</HEAD>
<BODY>
<NOSCRIPT>
    Si ves esto, tu browser no incorpora la etiqueta
</NOSCRIPT>
</BODY>
</HTML>
```

Con vistas a un uso futuro, esta etiqueta admite un parámetro opcional LANGUAGE que indica el lenguaje de script que se ha incrustado en el documento así como la versión de JavaScript.

<SCRIPT LANGUAGE="Versión de JavaScript";>
Sentencies JavaScript...
</SCRIPT;>

Versión de JavaScript especifica la versión de JavaScript en la que está escrito el código, y puede ser:

- *<SCRIPT LANGUAGE="JavaScript"> especifica JavaScript para Navigator 2.0.*
- *<SCRIPT LANGUAGE="JavaScript1.1"> especifica JavaScript para Navigator 3.0.*

Las sentencias encerradas entre las etiquetas son ignoradas si el browser que las lee no tiene el nivel de JavaScript especificado en el atributo *LANGUAGE*; por ejemplo:

- Navigator 2.0 ejecuta el código escrito con la etiqueta *<SCRIPT LANGUAGE="JavaScript">* e ignora el código escrito en la etiqueta *<SCRIPT LANGUAGE ="JavaScript1.1">*.
- Navigator 3.0 ejecuta el código escrito entre las etiquetas *<SCRIPT LANGUAGE="JavaScript"> o <SCRIPT LANGUAGE="JavaScript1.1">*.

Si el atributo *LANGUAGE* es omitido, Navigator 2.0 asume *LANGUAGE="JavaScript"* y *Navigator 3.0 asume LANGUAGE="JavaScript1.1"*

Puede usar el atributo *LANGUAGE* para escribir scripts que contengan código para Navigator 3.0 (con nuevas funciones que no existían en versiones anteriores) y que no provoquen un error ejecutándose bajo Navigator 2.0.

Ejemplo. Este ejemplo muestra como usar dos versiones diferentes de JavaScript una para JavaScript 1.0 y otro para JavaScript 1.1. El documento cargado por defecto es para JavaScript 1.0. Si el usuario utiliza Navigator 3.0, utilizará la función location.replace("..") implementada únicamente en esta versión de JavaScript.

```
<SCRIPT LANGUAGE="JavaScript1.1">
        location.replace("mipagina.html") //Sustituye la página actual por
        "mipagina.html"
</SCRIPT>
```

En el Netscape 3.0 se añade un nuevo parámetro opcional a la etiqueta <SCRIPT>, SRC. El objeto del mismo es el de actuar como un *include*, incluyendo en el documento un código JavaScript que pueda ser compartido por diversos documentos, es decir, posibilitar el uso de librerías. Se recomienda que éstas tengan extensión ".js". La sintaxis asociada será:

El atributo SRC debe especificar una URL[3], relativa o absoluta. Por ejemplo:

```
<SCRIPT SRC="libreria.js"></SCRIPT>
<SCRIPT SRC="http://home.netscape.com/functions/libreria.js">
```

Esta librería no puede contener código HTML, únicamente definiciones de funciones JavaScript.

2.8 Funciones en JavaScript

Las funciones son unos de los bloques fundamentales en JavaScript. Una función es un procedimiento escrito en JavaScript, es decir, un conjunto de sentencias que realizan una determinada tarea.

Una función consta de las siguientes partes básicas:

- Un nombre de función.
- Los parámetros pasados a la función separados por comas y entre paréntesis.
- Marcar el inicio y el final de la función entre llaves ({})

Es importante entender la diferencia entre definir una función y llamarla.

[3] Abreviación de las siglas en inglés de *Uniform Resource Locator que es un localizador uniforme de recursos en Internet. La primera parte de un url indica que protocolo usa y la segunda parte especifica la dirección IP o el nombre de dominio dónde se encuentra localizado dicho recurso.*

Definir una función es simplemente especificar su nombre y definir que acciones realizará en el momento en que sea invocada. Para ello se emplea la palabra reservada *function*.

```
Function nombre_de _la_función([parámetro1, parámetro2,...])
{
.....
return <valor_retorno>
}
```

Llamando a esta función realizará las acciones especificadas con los parámetros indicados.

```
Nombre_de_la_función(a,b);
```

Las funciones pueden definirse en cualquier parte del documento pero es aconsejable declararlas en la cabecera; de esta forma se garantiza que todas las funciones se cargen antes se que el usuario tenga la oportunidad de realizar ninguna acción en la que llame a una de ellas.

El siguiente ejemplo define una función en la cabecera (HEAD) de un documento y la llama en el cuerpo (BODY).

```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- Oculto el código a los browsers que no entiendan "JavaScript"
function cuadrado(numero) {
        return numero * numero
}
// Final de la ocultación-->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT>
document.write("La función retorna ", cuadrado(5), ".")
</SCRIPT>
</BODY>
```


La función cuadrado tiene un argumento, llamado numero.

La función consta de una sentencia return numero * numero que indica que ha de retornar el cuadrado del número pasado por parámetro.

En el cuerpo del documento hacemos una llamada a la función definida mediante la sentencia:

```
cuadrado(5)
```

La función ejecuta la sentencia *numero * numero* como *5 * 5* y retorna el valor 25, el script muestra el siguiente resultado en pantalla:

La función retorna 25.

2.9 Manejadores de eventos

La mayoría de las acciones de programa (al tratar con una aplicación de WWW) deben ser activadas por eventos. Los eventos son acciones que ocurren como resultado de alguna acción realizada por el usuario. Un click de ratón, la focalización de un campo en un formulario, modificar un campo de texto o mover el cursor son ejemplos de eventos. La segunda forma de incrustación de JavaScript en documentos HTML consiste en la definición de manejadores de eventos en las etiquetas.
La sintaxis general es:

<ETIQUETA manejador_evento = "Código JavaScript">

Donde ETIQUETA es cualquier etiqueta HTML que pueda relacionarse con un evento y *manejador_evento*, el evento en sí.

Cada evento es reconocido por ciertos objetos, etiquetas HTML, como son:

- **Focalización, desfocalización y edición**: *campos de texto, textareas* y *selections.*
- **Clicks**: *buttons,radio buttons, checkboxes, submit buttons, reset buttons*, y *enlaces.*
- **Selección**: *text fields, textareas.*
- **Señalización**: *enlaces.*

A partir de la versión 3.0 de Netscape, *onBlur* y *onFocus* se aplican también a ventanas y framesets.

El nombre de un manejador de eventos es el nombre del evento, precedido por "on.". Por ejemplo, el manejador de eventos para *focus* en *onFocus*.

La relación entre eventos y nombres de manejadores, así como su descripción, se puede observar en la tabla adjunta:

Evento	Manejador	Descripción
click	*onClick*	El usuario pulsa sobre un elemento de un formulario o un enlace
señalización	*onMouseOver*	El usuario coloca el ratón sobre una determinada área
carga	*onLoad*	El usuario carga la página
descarga	*onUnload*	El usuario sale de la página
focalización	*onFocus*	El usuario selecciona un elemento de un formulario como entrada
desfocalización	*onBlur*	El usuario quita la selección de un elemento de un formulario como entrada
edición	*onChange*	El usuario cambia el valor de un elemento text, textarea o select de un formulario
selección	*onSelect*	El usuario selecciona el campo de entrada de un formulario
Interrupción	*onAbort*	El usuario interrumpe la carga de una imagen
Error	*onError*	La carga de un documento o imagen produce un error.
Des-señalización	*onMouseOut*	El usuario saca el ratón de un área (imagemap) o enlace.
Inicialización	*onReset*	El usuario pulsa el botón de reset en un form.

Ejemplo. En el siguiente ejemplo definimos un campo de texto en el que entraremos un número, *x*. Al hacer click sobre el botón obtendremos el resultado (x^2) en el segundo campo de texto.

Al mismo tiempo, aparecerá un mensaje en la barra de status que indicará el número introducido así como el resultado obtenido tras aplicarle el cuadrado. Para realizar estas acciones utilizaremos la función definida más arriba como *cuadrado*.

Cada vez que el usuario haga un click sobre el botón *Cuadrado* (para determinarlo utilizaremos el manejador de eventos *onClick*) se llamará a la función del mismo nombre, pasándole como parámetro el valor introducido en el primer campo de texto (*x*).

Inmediatamente será mostrado en la zona de status un mensaje del tipo: *"Has entrado el número x y el resultado es* x^2*"* mediante la sentencia *window.status*,donde window es un objeto creado por el sistema que hace referencia a la ventana actual y *status* no es más que una de sus propiedades que especifica el mensaje a mostrar en la barra de estado.

El resultado es el siguiente:

Entra un número: [ ]

Resultado: [ ]

Y el código asociado:

```
<FORM>
Entra un número:
        <INPUT TYPE="text" NAME="entrada"
        SIZE=15>
        <INPUT TYPE="button" VALUE="Cuadrado"
        onClick="cuadrado(entrada.value)">
<BR>
Resultado:
        <INPUT TYPE="text" NAME="resultat" SIZE=15
        VALUE = 0>
</FORM>
```

Las modificaciones realizadas en la función cuadrado son las siguientes:

```
function cuadrado(numero){
        document.form1.resultat.value = numero * numero;
        window.status='Has entrado el número ' + numero + ' y el
        resultado es ' +document.form1.resultat.value;
}
```

Mediante los manejadores de eventos es posible dotar a los documentos HTML de una gran interactividad.

2.10 Validación de la información de un Formulario

Una de las utilidades más importantes que ofrece JavaScript es la posibilidad de realizar validaciones inmediatas dela información introducida en un formulario. Aunque existe la alternativa de utilizar un programa CGI que realice la misma función, poder llevar a cabo este proceso dentro de la máquina del cliente, ahorra un gran número de conexiones innecesarias
al servidor.

Validaciones alfabéticas

Un tipo elemental de información que aparece en muchos formularios es el tipo alfabético compuesto por los caracteres alfabéticos del idioma en particular con el que se trabaje.

Validaciones numéricas

Frecuentemente la información que hay que proporcionar a un elemento Text de un formulario tiene que ser numérica. Principalmente se distinguen 3 tipos de formatos numéricos:

- Número natural formado por caracteres numéricos.
- Número entero formado por un signo inicial opcional (+,-) seguido de caracteres numéricos.
- Número real formado por un signo inicial opcional (+, -), seguido de caracteres numéricos y, opcionalmente, seguido de la coma decimal y una serie de caracteres numéricos.

Validaciones de fechas y horas

Este tipo de datos aparece con menos frecuencia en los formularios, pero es necesario tener a disposición funciones de validación por si se da el caso. Las fechas y las horas se han agrupado dentro de la misma categoría, ya que las comprobaciones que hay que realizar son similares. En ambos casos, se trata de tres grupos de caracteres numéricos que cumplen una serie de restricciones de rango entre los que se intercala un separador específico.

A continuación veremos un pequeño ejemplo de la validación de la información introducida por el usuario en un formulario:

```
<script language="JavaScript">
<!--

function validaciones(){  //Función para la validación de los datos de la forma
        var forma = document.forms[0]; //Obtención de los datos de la forma

        //Validación del Campo Nombre

        if(forma.nombre.value == ""){
                alert("El nombre es requerido");
                forma.nombre.focus(); //Si no existe el nombre nos regresa
                 al campo para capturarlo
                return;
        }


} - ->
</script>

Función para la Validación de Fechas.

function validaFechas(fecha){
        var s = new String(fecha);
        var diag = s.indexOf("/",0);
        var diag2 = s.lastIndexOf("/");
        var mes = s.substring(0,diag);
        var dia = s.substring((diag + 1),diag2);
        var anio = s.substring((diag2+1),s.length);
        if(diag == diag2 ||diag < 1 || dia < 1 || dia > 31 || mes < 1 || mes > 12 ||
anio<1995 || anio > 2100 ||(mes == 2 && dia >29) || isNaN(dia) || isNaN(mes) ||
isNaN(anio)){
                alert("La fecha es incorrecta, el formato es: MM/DD/AAAA");
                        return 1;
        }else{
                return 2;
        }
}
//-->
</script>
```

3. Arquitectura Cliente/Servidor

Hoy en día numerosas aplicaciones funcionan según la arquitectura Cliente/Servidor, esto es que las máquinas **clientes** (máquinas que son parte de la red) se conectan a un **servidor** (una máquina poderosa en términos de capacidades) que le proporciona servicios al cliente. Estos servicios pueden ser programas que proveen datos, archivos, etc.

Los servicios son explotados por los programas llamados **Programas Clientes** que se ejecutan en las máquinas clientes. Los clientes se encargan principalmente de recuperar información del serivdor, por ejemplo un cliente FTP como WS-FTP® recupera archivos del servidor, un cliente de correo electrónico como Outlook® obtiene mensajes electrónicos del servidor.

3.1 Ventajas de la Arquitectura Cliente/Servidor

El modelo cliente/servidor es reconocido particularmente por las siguientes características:

- **Recursos centralizados:** Ya que el servidor es le centro de la red, ya que puede generar recursos comunes a todos los usuarios, por ejemplo una base de datos centralizada con el fin de evitar problemas de redundancia y contradicción.
- **Una mejor seguridad:** Por que el número de puntos de entrada que permiten el acceso a los datos, es menos importante.
- **Una administración a nivel servidor:** Los clientes no requieren de ser administrados.
- **Una red evolutiva:** Gracias a esta arquitectura se pueden suprimir o agregar clientes sin perturbar el funcionamiento de la red y sin grandes modificaciones.

3.2 Desventajas de la Arquitectura Cliente/Servidor

La arquitectura cliente/servidor también tiene algunas ligeras desventajas de las cuales podemos decir:

- **Un costo elevado:** Por la alta tecnología que debe tener un servidor.
- **Un eslabón débil:** Ya que toda esta arquitectura se concentra en el servidor y si este falla todo fallará. Otro punto importante es la red en la que estamos trabajando de igual forma si la red falla el cliente y el servidor también fallarán.

3.3 Funcionamiento de un sistema Cliente/Servidor

La ***imagen 3.3-1*** muestra como funciona un sistema Cliente/Servidor:

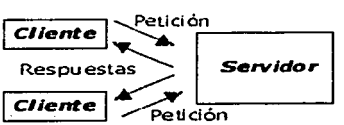

Imagen 3.3-1

- El cliente hace una petición al servidor gracias a su dirección[4] y puerto[5].
- El servidor recibe la petición y responde gracias a la dirección de la máquina Cliente.

[4] Un nombre o conjunto de bits que identifica un componente en una red. En una red LAN por ejemplo cada nodo tiene una dirección única.
[5] En una red TCP/IP es un punto de entrada para una conexión lógica. El número de puerto identifica que tipo de puerto es, por ejemplo el puerto 80 es usado para el servicio de http y el puerto 20 para FTP.

3.4 Proceso de un Cliente

El Cliente es un proceso (programa) que envía un mensaje al servidor (programa), en espera de que el servidor lleve a cabo una tarea (servicio). Los programas cliente usualmente se encargan de la interfase de usuario, validación de datos que provienen del usuario. El proceso Cliente es conocido también como el Front-End de la aplicación que el usuario ve y con el que interactúa, además de que es considerado una interfaz entre el usuario y el resto de la aplicación. El proceso cliente también se encarga del manejo de los recursos locales tales como el monitor, teclado y periféricos. Uno de los elementos clave en un cliente es la interfaz gráfica de usuario(graphical user interface GUI).

3.5 Proceso de Servidor

Un proceso de Servidor (programa) lleva a cabo las tareas que le han sido pedidas por el Cliente. El Servidor generalmente recibe peticiones del cliente, peticiones de acceso a datos de una Base de Datos y actualizaciones de los mismos, manejo de la integridad de los datos. Algunas veces el servidor realiza transacciones comunes o complejas de datos. Los procesos del servidor pueden correr en otra máquina en la red, es decir que el servidor puede ser el servidor principal o puede ser un servidor de archivos o en algunos casos alguna otra máquina pude proveer de servicios de aplicación. El Servidor trabaja como un sistema que provee de software encargado de administrar y compartir los recursos tales como bases de datos, impresoras, enlaces de comunicación o algunos otros procesos. Finalmente se puede decir que el Servidor es conocido también como el Back-End.

3.6 Arquitectura a 2 Niveles

La arquitectura de 2 niveles es cuando se dice que el Cliente habla directamente con el servidor, esta arquitectura se usa generalmente cuando se tienen ambientes de trabajo pequeños, es decir, menos de 50 usuarios.

Un error común en un ambiente Cliente/Servidor es modelar o hacer prototipos de una aplicación en una arquitectura de 2 niveles sin contemplar que ese ambiente crecerá y tendrá más usuarios, esto conlleva a que el resultado sea un sistema ineficiente. Por lo que se recomienda siempre en pensar en una arquitectura de 3 niveles.

La arquitectura a 2 niveles se caracteriza por que el cliente hace una petición de un recurso y el servidor lo atiende directamente. Esto significa que el servidor no hace llamadas a otra aplicación a fin de proveer el servicio. En la ***imagen 3.6-1*** podemos observar la representación gráfica de esta arquitectura a 2 niveles.

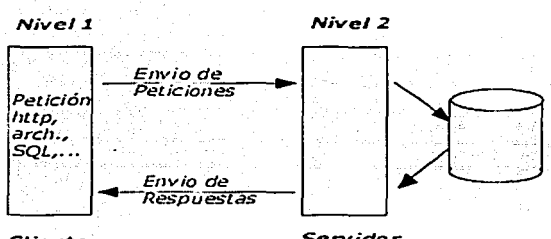

Imagen 3.6-1

3.7 Arquitectura a 3 Niveles

Esta arquitectura introduce un servidor (o un agente) entre el cliente y el servidor. La tarea de este agente puede ser variable por ejemplo: puede proveer servicios de translación.

Partes de la arquitectura a 3 niveles:

1. **El Cliente:** El que hace peticiones de los recursos.
2. **El Servidor de Aplicación:** Llamado también Middleware, el servidor da respuesta a la petición pero haciendo llamadas a otro servidor.
3. **El Servidor Secundario**: Generalmente un servidor de base de datos, que provee de algún servicio al primer servidor.

En la ***imagen 3.7-1*** podemos observar la representación gráfica de esta arquitectura a 3 niveles.

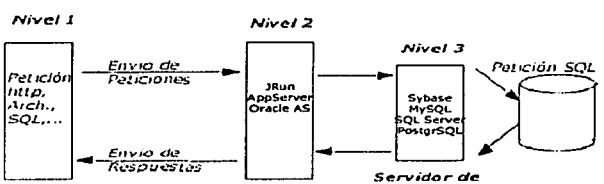

Imagen 3.7-1

3.8 Comparación entre los tipos de arquitecturas

La arquitectura a 2 niveles es una arquitectura en la que el servidor es polivalente, es decir, que es capaz de proveer directamente al conjunto de peticiones de los clientes. Mientras que en la arquitectura a 3 niveles las aplicaciones a nivel servidor son descentralizadas, es decir, cada servidor esta especializado en una tarea por ejemplo, servidor web, servidor de bases de datos. Por lo que la arquitectura a 3 niveles permite:

- Una mejor flexibilidad.
- Una mayor seguridad (La seguridad puede ser definida por cada servicio).
- Un mejor performance (ya que las tareas son repartidas).

3.9 La Arquitectura Multi-Nivel

En la arquitectura a 3 niveles, cada servidor (nivel 1 y 2) hace una tarea (servicio) especifica. Así que un servidor puede usar los servicios de uno o varios servidores con el fin de llevar acabo su propio servicio:

Por consecuencia la arquitectura a 3 niveles es potencialmente una arquitectura a N niveles. En la ***imagen 3.9-1*** podemos observar la representación gráfica de esta arquitectura.

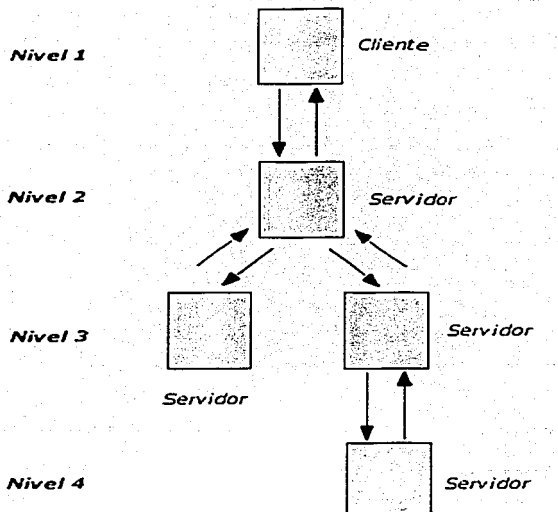

Imagen 3.9-1

3.10 Middleware

La conectividad permite a las aplicaciones la transparencia entre los programas y procesos sin importar donde se encuentren. El elemento clave de la conectividad es el Sistema Operativo de Red (Network Operating System: NOS). El NOS provee de servicios como enrutamiento, distribución, mensajería, archivos e impresoras y servicios de manipulación de la red. Los NOS proveen de diversos protocolos para un servicio especifico. Estos protocolos se pueden dividir en 3 grupos: Medios, Transporte y cliente-servidor. Algunos ejemplos de NOS son: UNIX®, Novell®, Microsoft® Lan Manager y Windows NT®

Los protocolos de medios determinan el tipo de conexiones físicas usadas en una red (Algunos ejemplos de protocolos de medios son Ethernet, Token Ring, Fiber Distributed Data Interface (FDDI), Coaxial y par Trenzado).

El protocolo de Transporte provee de mecanismos para mover los paquetes de datos del cliente hacia el servidor (algunos ejemplos pueden ser de Novell IPX/SPX, de Apple AppleTalk, Transmission Control Protocol / Internet Protocol (TCP/IP), Open Systems Interconnection (OSI) y Government Open Systems Interconnection Profile (GOSIP). Una vez que la conexión física se ha establecido y se ha elegido los protocolos de transporte, un protocolo cliente-servidor es requerido antes que el usuario pueda accesar a los servicios de red. Un protocolo cliente-servidor dicta la manera en la que los clientes hacen la petición de información y de servicios hacia un servidor y también como el servidor contesta a estas peticiones, algunos ejemplos de estos protocolos son NetBIOS, RPC, Advanced Program-to-Program Communication (APPC), Named Pipes, Sockets, Transport Level Interface (TLI) and Sequenced Packet Exchange (SPX)).

3.11 Procesamiento Cooperativo

El procesamiento cooperativo requiere 2 o mas procesadores para completar una transacción simple. El procesamiento cooperativo se relaciona con los procesos distribuidos y cliente/servidor. Esta es una forma de procesamiento de información distribuida donde 2 o mas procesos son requeridos para completar alguna transacción.

Usualmente estos programas interactúan y se ejecutan concurrentemente en diferentes procesadores. El procesamiento cooperativo puede también considerarse como un tipo de procesamiento cliente-servidor si la comunicación entre los procesadores se lleva a cabo a través de un mensaje entre las arquitecturas.

3.12 Procesamiento Distribuido

Es la distribución de aplicaciones a través de múltiples plataformas de procesamiento. El procesamiento distribuido implica que procesamiento deberá ocurrir en más de un procesador en orden en que una transacción sea completada. En otras palabras, el procesamiento es distribuido entre 2 o más computadoras y los procesos no son corridos al mismo tiempo, sino a diferentes tiempos.

Por ejemplo: cada proceso se lleva a cabo como parte de una aplicación en una secuencia lógica. A menudo los datos usados en un ambiente de procesamiento distribuido es también distribuido entre diversas plataformas.

3.13 La Intranet

La explosión de la World Wide Web es gracias a la aceptación de un transporte común de información (TCP/IP), un servidor estándar (HTTP), el lenguaje de marcas (HTML). Muchas empresas se han dado cuenta que estas tecnologías pueden ser usadas para aplicaciones bajo la Arquitectura Cliente/Servidor dentro de sus empresas de igual forma que como se usan en Internet. De esto ha surgido el concepto de Intranet que no es mas que el uso de las tecnologías de Internet en implementaciones internas de aplicaciones cliente/servidor.

Una de las ventajas clave en las Intranets es que el problema de la manipulación de código ha sido reducido enormemente. Gracias a la existencia de un browser común en cada estación de trabajo, todos los cambios en la interfaz de usuario pueden hacerse cambiando código en el servidor de HTTP. Esto es mucho más cómodo que cambiar o actualizar el código del cliente en 2,000 estaciones de trabajo, por ejemplo.

Una segunda ventaja muy importante es que si la empresa en cuestión ya usa Internet no se necesitan códigos adicionales que tengan que tener alguna licencia o que se necesiten instalar algunas cosas en los clientes de las estaciones de trabajo. El usuario puede tener acceso tanto a la información interna como a la externa ya que se encuentran integradas.

Una desventaja que ha ido desapareciendo rápidamente es la posibilidad de integrar código personalizado a los clientes, pero ahora con la liberación de diversas herramientas para generar código como Java y Javascript esta limitación a casi desaparecido.

En la ***imagen 3.13-1***, podemos observar la representación de una INTRANET.

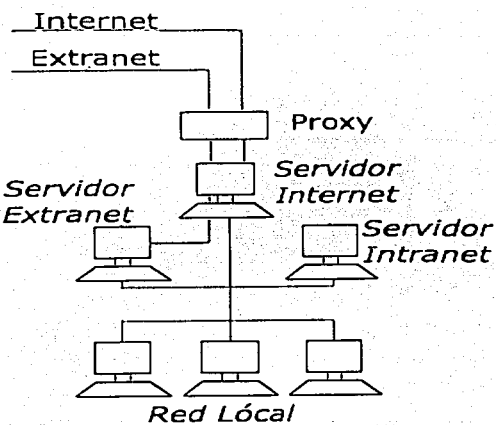

Imagen 3.13-1

3.14 Características de la Arquitectura Cliente/Servidor

1) Combinación del Cliente o Front- End que interactúa con el usuario y el Servidor o Back-End que interactúa con los recursos compartidos. El proceso del Cliente contiene la solución lógica que provee la interfaz entre el usuario y el resto de las aplicaciones del sistema. El proceso del Servidor actúa como un software que administra los recursos compartidos como bases de datos, impresoras, modems y procesadores de alto poder.

2) Las tareas del Cliente y las tareas del Servidor tienen diferentes requerimientos en cuanto al proceso de computo de cada uno, por ejemplo la velocidad del procesador, la memoria, la velocidad de los discos y su capacidad y los dispositivos de entrada/salida.

3) El ambiente es típicamente heterogéneo. El hardware y el sistema operativo del Cliente y el Servidor no son usualmente el mismo. Los procesos del Cliente y el Servidor se comunican a través de un conjunto bien definido de interfaces de aplicación de programa (application program interfaces API's) y RPC's[6].

4) Una importante característica de los sistemas Cliente/Servidor es la escalabilidad. Es decir pueden escalarse horizontal y verticalmente. Escalarlo horizontalmente significa poder poner o remover estaciones de trabajo según las necesidades que se presenten. Escalarlo verticalmente significa poder migrarlos a un servidor mas potente y mas grande o a múltiples servidores.

[6] Abreviación de las siglas inglés *de remote procedure call*, que es un tipo de protocolo que permite a un programa en una computadora ejecutar un programa en un servidor.

3.15 Tipos de Servidores

La forma más simple de un servidor son los servidores de disco y servidores de archivos. Con un Servidor de archivos, el cliente hace peticiones de archivos o peticiones para almacenamiento de archivos sobre la red hacia el servidor de archivos. Esta forma de servicios de información y datos requieren un gran ancho de banda y pueden hacer que la red sea mas lenta. Tradicionalmente en una red LAN se permite que los usuarios compartan recursos como archivos de datos y dispositivos periféricos, estos servidores son mejor conocidos como NFS's (Networked File Server).

Algunos de los tipos de servidores más importantes, son los Servidores de Bases de Datos, Servidores de Transacciones y los Servidores de Aplicaciones.

En los Servidores de Bases de Datos, los Clientes hacen peticiones de SQL (Structured Query Language) como mensajes al servidor y los resultados del Query son regresados al cliente. El código que procesa la petición SQL y los Datos se encuentran en el servidor. Sybase®, MySQL®, PostgreSQL®, Microsoft® SQL Server® son ejemplos de servidores de bases de datos.

En los Servidores de Transacciones los clientes invocan procedimientos remotos que se encuentran en el servidor que también contienen un manejador de bases de datos. Microsoft Transaction Server® y CICS de IBM® son ejemplos de servidores de transacciones.

Hay sentencias de procedimientos en el servidor que ejecutan una serie de sentencias SQL (Transacciones) que suceden o fallan todas como una unidad o conjunto. Las aplicaciones basadas en servidores de transacciones son llamadas procesamiento de transacciones en línea (on-line Transaction Processing (OLTP)) y son útiles cuando se requieren aplicaciones de misión critica en donde se requieren de 1 a 3 segundos de tiempo de respuesta, 100% de tiempo de procesamiento, extremo control en la seguridad e integridad de la base de datos.

Los Servidores de Aplicación no son necesariamente bases de datos centralizadas pero son usadas para servir a las necesidades de los usuarios. Oracle® Aplication Server®, Borland® AppServer® y JRun® de Macromedia® son ejemplos de servidores de aplicación.

3.16 Manejo Remoto de Datos

En el manejo remoto de datos, toda la aplicación se encuentra en el Cliente y el manejo de datos se encuentra en el servidor. El manejo remoto de datos es relativamente fácil de programar por que hay un solo programa de aplicación. El cliente se comunica con el servidor usando SQL, el servidor, por consiguiente, responde con datos que satisfacen al query enviado. Los RDBMS[7] que ofrecen manejo remoto de datos proveen una capa de software en el cliente que maneja la comunicación con el servidor DBMS[8]. Este estilo representa un servidor de bases de datos LAN o una especie de servidor de archivos.

Las estaciones de trabajo (workstations) soportan la presentación y función lógica; y la interfaz con el servidor de datos a través del lenguaje de manipulación de datos (Data Manipulation Language). El Manejo de Datos Distribuidos es una extensión del manejo remoto de datos y utiliza las facilidades de los DBMS's para acceder a los datos distribuidos de una manera transparente a los usuarios. En la ***imagen 1.16-1*** podemos observar la representación gráfica del manejo remoto de datos.

[7] Abreviación de las siglas en inglés de *relational database management system* que es un tipo de DBMS que almacena datos en forma de tablas relacionadas.
[8] DBMS es una colección de programas que permite almacenar, modificar y extraer información de una base de datos.

4. Introducción al Server-Side Scripting

Hoy en día la programación Server-Side[9] esta llegando a ser una herramienta muy importante. La programación Server-Side puede utilizarse para beneficiar a las páginas web haciendolas más dinámicas y útiles. También se puede utilizar para desarrollar intranets y ayudar al manejo de la información en una empresa, el manejo del inventario de clientes y proveedores. Cuando la programación Server-Side se usa en conjunción con HTML se conoce como server-side scripting, pero exactamente, ¿Qué es el server-side scritping?, antes de definirlo es necesario conocer algunos conceptos.

En primer lugar la palabra "server" (Servidor). El servidor no es mas que un programa que provee servicios a otras computadoras. La computadora en donde el servidor esta corriendo también es referido como servidor, y las computadoras que reciben los servicios del programa servidor se llaman clientes.

A continuación la palabra "script", un script es un tipo de programa que es escrito para aplicaciones basadas en el Web; entonces ¿Qué es un programa?. Un programa es un conjunto especifico de instrucciones que la computadora ejecuta.

Ahora se tienen los elementos necesarios para definir al server-side scripting. El server-side scripting es esencialmente la programación de aplicaciones de servidor. Estas aplicaciones, una vez que se han programado son, a menudo, benéficas para el servidor y los clientes. El server-side scripting posee grandes beneficios para las páginas de Internet porque permite hacer muchas mas cosas que la programación con HTML.

Algunos de los lenguajes de scripting son ASP (Active Server Pages) de Microsoft®, JSP (JavaServer Pages) de Sun Microsystems®, Perl® desarrollado por Larry Wall, PHP® desarrollado por Rasmus Lerdorf y ColdFusion de Macromedia. De todas estas opciones se debe considerar cuál es la más eficiente según los recursos que tengamos, es decir, qué tanta interacción se tenga con las Bases de Datos, el acceso a los sistemas de archivos del Sistema Operativo y si ese sistema operativo soporta la creación de páginas dinámicas. Finalmente las razones por las que se debe elegir uno sobre otro pueden ser muy subjetivas, mas bien todo depende de nosotros mismos, los conocimientos que poseamos y qué es lo que estamos tratando de desarrollar.

[9] Server-Side Significa "Del lado del Servidor", es decir programación del lado del servidor.

También hay que tomar en cuenta qué clase de programadores o desarrolladores somos, por ejemplo hay quienes no soportan la programación entre brakets < >, para esto se tiene perl o PHP. Si por el contrario nos gusta la idea de usar etiquetas como las de HTML aunque algunos digan que esto no es programar, podemos optar por ColdFusion. Hay, claro, mucho mas aspectos que tomar en cuenta antes de decidir en que trabajar por ejemplo la rapidez y estabilidad, aunque medir la rapidez y la estabilidad depende mucho también de quien haga esto, por ejemplo los investigadores que trabajaron de Linux® contra NT®, daban la razón a Linux® y qué pasa cuando Microsoft® hace algo contra Linux®, pues lógicamente le dan la razón a su compañía.

Algunas personas piensan que para poder determinar cuales son unas de las mejores propuestas de desarrollo, lo mejor es preguntar a los expertos o profesionales de la programación WEB sobre que plataformas prefieren trabajar y por qué. Como resultado de estas encuestas se ha descubierto que la mayoría prefiere trabajar con ASP, tal vez por que hay muchisimo mas computadoras con un sistema operativo de Microsoft® que otros sistemas operativos.

Un pequeño ejemplo que nos puede ayudar a entender lo que es el server-side scripting es, un script CGI es una aplicación server-side por que corre en un servidor web. En contraparte los scripts hechos en Javascript son aplicaciones client-side por que se ejecutan en el propio navegador (el cliente). Los Applets de Java pueden ser server-side o client-side dependiendo de en que computadora (el servidor o el cliente) se ejecuten.

4.1 El Presente y el futuro de los Server-Side Scripting

No hay duda que el número de usuarios de Internet crece día con día cada vez más rápido, además de que cada día es más popular hacer negocios a través de Internet. Hoy en día muchos de los sitios web de las compañías y de sitios de negocios han sido desarrolados con server-side scripting.

Existen buenos augurios para el uso de la tecnología de los server-side scripting en el futuro, pero la cuestión es qué tipo de lenguaje de scripts se utilizará, solo el tiempo nos podrá responder esta pregunta. El debate de cual es el lenguaje mas efectivo puede ser llevado a cabo observando cual de estos se acopla mas a las necesidades de los desarrolladores para hacer los sistemas basados en web, ademas de cual se actualiza mejor, según la evolución de las tecnologías de web.

Por el momento, hoy se le dan diversos usos a la tecnología de los server-side scripting, a parte de los beneficios que ofrece a las aplicaciones de Internet también se pueden utilizar en Intranets.

Los Server-Side Scripting ofrecen un potencial increíble a las áreas de e-comercio, negocios y otros tipo de sitios web.

Muchos de los sitios que poseen servicios en línea (online) utilizan server-side scripting para hacer ventas en línea que hacen que sus sitios sean mas novedosos y útiles. Además de que son sitios simples de usar. Los scripts también pueden ser utilizados en sitios de entretenimiento para hacer Chats (conversación en línea) y juegos en línea.

A continuación veremos las principales tecnologías para el desarrollo de aplicaciones Server-Side:

4.2 ASP

Cuando se tiene que decidir con cuál lenguaje de scripts trabajar como PHP®, ASP®, ColdFusion® o algún otro, lo más importante es considerar quién es el que va a trabajar en nuestro sitio web ahora y en el futuro. No sería una buena decisión utilizar PHP® si las personas que se van a encargar de nuestro sitio no están familiarizados con Unix®, Perl® y C®. Esto es, quién va a dar mantenimiento al sitio si el experto en Unix® deja de trabajar en nuestra empresa.

Las Active Server Pages (ASP's) creadas por Microsoft® pueden ser escritas en VBScript, un lenguaje de scripts fácil que usa una sintaxis similar a la de Visual Basic. Si en la compañía existen varias personas que ya conocen Visual Basic el sitio de la empresa puede tener mantenimiento ahora y en el futuro. Por otro lado, si se tiene a gente con alguna otra formación como JavaScript o Perl esto será muy útil ya que se pueden usar estas tecnologías para escribir las ASP's. Sin embargo, con la extensiva integración de capacidades de la tecnología .NET® de Microsoft®, se pueden crear páginas ASP de forma muy sencilla con las librerías de dicha tecnología.

El beneficio más grande de ASP es la posibilidad de utilizar los objetos COM[10]. De la misma forma en que ASP es fácil de usar, los objetos COM también lo son. Con una sola línea de código es posible crear una instancia de un objeto COM, por lo que a partir de esto es posible utilizar al objeto cada que sea requerido llamando a sus métodos y propiedades.

De esto surgen 2 grandes beneficios: primero, se pueden utilizar los beneficios de los objetos COM que se usan en Visual Basic o Visual C++ en nuestras páginas ASP y segundo se pueden crear nuestros propios objetos COM para usarlos en nuestras páginas ASP.

[10] Abreviación de las siglas en inglés de Component Object Model, que es un modelo de desarrollo de código binario desarrollado por Microsoft® que permite a los programadores crear otros objetos a partir de estos.

Haciendo uso de los objetos COM ya desarrollados, los desarrolladores pueden reducir la generación de código en las aplicaciones. Por ejemplo, cuando se instala IIS (Internet Information Server de Microsoft, que es un servidor web) y ASP, un conjunto de útiles objetos COM son agregados al Servidor Web incluyendo al "ad rotator". El "ad rotator" hace ciclos aleatorios en un banner (cartel) de avisos.

Un desarrollador de ASP's inmediatamente después de instalar IIS puede crear una página contendiendo dos líneas de código que despliegue aleatoriamente una serie de banners. Esto, por ejemplo, es imposible de hacer en PHP, cuando se acaba de instalar.

Los beneficios de los objetos COM no pueden ser completamente apreciados hasta que se hace uso de los Actives Data Objects (ADO) una combinación de los objetos COM que son usados para el acceso a datos. Con ADO se puede utilizar en Visual Basic o Visual C++ de la misma forma que en una página ASP, se pueden hace conexiones a cualquier archivo de datos, transferir información desde una base de datos en SQL Server a una hoja de cálculo de Excel.

4.3 ColdFusion

Otra de las herramientas favoritas de los desarrolladores de aplicaciones para el WEB es ColdFusion® creado por Allaire Corporation of Cambridge que posteriormente se fusionó a la empresa Macromedia, ColdFusion® incluye un servidor web y un conjunto de herramientas diseñadas para integrar bases de datos y páginas web. Un ejemplo, puede ser, que en una página web un usuario pueda introducir un código postal que se envía al servidor, y éste haga una consulta(query) a la base de datos para obtener información acerca de cuales son los teatros o cines que se encuentran en dicho código postal y posteriormente presente los resultados en una página HTML.

ColdFusion es muy simple y se basa en las sintaxis de etiquetas como en el HTML, aunque dichas etiquetas se escriben en el lenguaje Cold Fusion Markup Language (CFML) que simplifica la integración con bases de datos y evita el uso de lenguajes complejos como C++ u otros.

ColdFusion tiene un IDE[11] que puede ayudar a las empresas a obtener productividad rápidamente. Por otro lado no existe nada que se haga en ColdFusion que no se pueda hacer en ASP o PHP. ColdFusion ayuda a los desarrolladores a hacer el trabajo más rápido y eficiente.

La razón de esto es muy simple: a diferencia de otras herramientas, ColdFusion no fue remendado como otras tecnologías lo han hecho, sino que ha sido desarrollado desde cero. Los creadores de ColdFusion se dedicaron a observar cuales eran los principales problemas que presentaban los diversos paradigmas y trataron de ver la forma de resolverlos.

Por ejemplo, el siguiente código hecho con ASP se conecta a una base de datos en SQL Server, mostrando algunos datos en el navegador.

```
<%

        Set OBJdbConnection =Server.CreateObject("ADODB.Connection")
         OBJdbConnection.Open "nba_membership"
        SQLQuery = "Select id, business FROM Directory"
        Set rsCustomers = OBJdbConnection.Execute(SQLQuery)

        Do Until rsCustomers.EOF
                Response.Write (rsCustomers("ID") & " " &
        rsCustomers("Business"))
                rsCustomers.MoveNext
        Loop

%>
```

El mismo Código en ColdFusion.

```
<cfquery name="rsCustomers" datasource="nba_membership">
        select id, business from directory
</cfquery>

<cfoutput query="rsCustomers">#id# #business#</cfoutput>
```


[11] Abreviación de la palabra en inglés de Integrated Development Environment, es decir, un entorno de programación integrado a una aplicación.

Ambos códigos hacen lo mismo, pero el código de ColdFusion es mucho más elegante que el código de ASP. Como ya hemos mencionado el lenguaje nativo de ColdFusion es el CFML (ColdFusion Markup Languge) el cual usa pocos comandos. No es necesario ser un genio para saber lo que hacen las etiquetas <cfquery> y <cfoutput>, mientras que su equivalente en ASP es un poco más complicado entender a primera vista.

Este concepto es una de las razones por lo que algunos programadores prefieren a ColdFusion. Prácticamente podemos decir que con ColdFusion existe mucha transparencia de cómo se hacen las cosas, si se quiere hacer un Query (Consulta), si se quieren ver algunos valores, etc. Con este lenguaje no existe misterio alguno de cómo hacer algo.

Hoy en día existen en el mercado diversas herramientas con las cuales podemos desarrollar sitios con ColdFusion de una manera rápida y sencilla; por ejemplo, el ColdFusion Studio trata de automatizar la generación de código tanto como le sea posible.

4.4 JSP

Ser uno de los últimos lenguajes de scripts no es necesariamente una cosa mala, especialmente si se puede beneficiar de los errores de otros. Este es el caso de las Java Server Pages (JSP). Mientras que las comunidades asociadas a otros lenguajes de scripts maduraban y pulían los errores de estos lenguajes los desarrolladores de Sun Microsystems® buscaban en los fundamentos de la arquitectura Cliente/Servidor y se preguntaban qué se podría hacer para mejorar a dicha arquitectura. Trabajando con el lenguaje JAVA, desarrollaron el Servlet API, que es un conjunto de clases que dan nuevas capacidades y funcionalidad al WEB-Serving.

JSP utiliza servlets para crear un lenguaje dinámico de scripts con contenido variable, generalmente de recuperación de información de bases de datos, incluido dentro de HTML. Conceptualmente es similar a ASP pero con una mayor ventaja: mientras que ASP usa VBScript o Jscript, JSP utiliza todo el potencial de JAVA.

Los Servlets han logrado ventaja en el área de la persistencia. Cuando un servlet es cargado en la memoria del servidor, este se almacena como un objeto JAVA muy compacto.

Cuando un servidor que utiliza Servlets recibe una petición ya no se lanzan interpretes o se instancian variables (después de la primera vez). Esto significa que los servlets son muy eficaces, es decir, que los servlets permanecen como marinos siempre preparados, listos para servir. También, además de su eficiencia en este sentido los servlets se integran firmemente con el servidor, que permite interacciones más sofisticadas con el servidor que como se haría con un CGI.

Cuando una máquina cliente hace la petición de una página JSP por primera vez, el servidor automáticamente, construye, compila y comienza automáticamente en background un Java Servlet que genera una página HTML. Esa página HTML es enviada al Navegador del cliente. Cada vez que la página JSP es accesada, el Servlet se almacena en la memoria del servidor como Java Byte code, listo para accesar a una base de datos y crear la página HTML que el cliente pide. Con una página ASP el código necesita ser interpretado cada vez que el cliente hace la petición de la página, lo cual hace que el proceso de generación de la página sea más tardado.

JSP brinda todo el poder del lenguaje JAVA sobre todo en lo que se refiere a la arquitectura Cliente/Servidor. Portabilidad, multiproceso, grandes librerías de clases, código orientado a objetos, grandes características de seguridad, elegancia, todas estas son algunas de la ventajas de JAVA. Para una mayor referencia, observemos cómo las personas que usan ColdFusion han implantado aplicaciones de Sun® en sus servidores para así poder interactuar con JSP, Servlets y otras tecnologías de JAVA.

No necesariamente se necesita ser un programador sofisticado de JAVA para usar JSP. Si se combina JSP con JavaBeans todo será más fácil. Por otro lado si los desarrolladores saben un poco de JavaScript, con eso será suficiente para comenzar.

4.5 Perl

Perl es un lenguaje de alto nivel, creado por Larry Wall. Perl deriva del lenguaje de programación C y de los lenguajes sed y awk del shell de unix. Su nombre significa Lenguaje Practico de Extracción y Reportes (Practical Extraction and Report Language).

El proceso de archivos de Perl y fácil manipulación de textos hacen de Perl una herramienta muy útil para el manejo de utilidades de los sistemas, herramientas de software, manipulación de tareas de los sistemas, acceso a bases de datos, programación gráfica, networking y la programación para el Web.

Todas estas características hacen que Perl sea muy popular entre los administradores y los programadores de CGI's.

La Cultura pre-populista de internet y el profundo escepticismo de la sobrevivencia de Perl dado que es gratis y de libre distribución, Perl hoy en dia es soportado por sus propios usuarios. El corazón, la librería estándar, los módulos opcionales y la documentación han sido escritos por voluntarios, es decir por personas que desean contribuir con algo que es gratis y de código abierto.

Las diversas liberaciones de Perl incorporan una serie de arreglos a los errores de liberación (Bugs) además de nuevas funciones. Cada liberación de Perl es probada exhaustivamente antes de liberarla y cada año es liberada una nueva versión de Perl.

Perl posee la característica de que es muy fácil de aprender, sobre todo para aquellos que apenas inician. Perl es similar como ya se dijo a C, a los scripts de awk, a los scripts del shell de unix y aún similar a BASIC, por lo que aquellos que tengan una pequeña experiencia de programación en estos lenguajes ya están preparados para comenzar con Perl.

Finalmente ya que Perl es un lenguaje interpretado, se pueden escribir programas y probarlos sin el paso intermedio de la compilación, permitiendo de esta manera experimentar y probar rápida y fácilmente nuestros programas.

Perl es flexible y extensible suficientemente para hacer virtualmente cualquier tarea. Para algunas personas Perl es el que vino a remplazar al shell scripting.

Si se tiene una librería que provee un API[12], se puede hacer cualquier componente haciéndola disponible como cualquier otra función de Perl o variable usando una extensión de Perl escrita en C o C++ y dinámicamente ligada en el interprete principal de Perl. También se puede escribir el programa principal en C o C++ y después ligarlo al código de Perl sobre la marcha para crear una buena aplicación.

4.6 PHP

Al hablar sobre PHP no se puede dejar de comentar antes sobre el código abierto (Open Source). La filosofía del Open Source es en si una de las grandes razones detrás de la popularidad de PHP.

La mayoría de los artículos que hoy en día podemos leer en revistas y periódicos han sido escritos con muchos argumentos que se basan precisamente en la posición del Open Source ó Código Abierto en el moderno e-comercio en Internet. Basta con decir que Internet y el e-comercio no serían lo que hoy en día son si no fuera por el Open Source. Algunos ejemplos de programas que se rigen a través del código abierto son: el servidor web Mosaic® de la empresa NCSA (National Center for Supercomputing Applications), Apache®, Perl® y por su puesto el sistema operativo LINUX®.

Como ya hemos dicho la principal característica de PHP es su costo: CERO. Para mucha gente esto es esencial, porque por ejemplo en una vieja PC se puede correr Linux®, Free BSD, o cualquier otra versión de Linux® que se tenga. En dicha computadora se puede instalar Apache®, PHP®, MySQL® y algún editor de texto. ¿Cuál es el costo total de esto? simplemente el tiempo en que tardemos en instalar, compilar y poner a punto nuestra computadora. La pregunta es ¿Por qué pagar por algo? Cuando se puede conseguir algo mejor, funcional y sobretodo gratis.

[12] Abreviación de las siglas en inglés de Application Program Interface, es decir, una interfaz de programación de aplicaciones, que es un conjunto de rutinas, protocolos y herramientas para la construcción de programas de aplicación.

PHP fue desarrollado por Rasmus Lerdorf en 1994. PHP tiene la ventaja de parecerse a otros lenguajes en cuanto a su sintaxis, por ejemplo incluye C, Java, Perl y otros. Para mucha gente con experiencia previa en programación, esto significa que hacer su primera aplicación en PHP será una cuestión muy sencilla.

Gracias al módelo de código abierto de PHP podemos crear nuevas funciones y métodos que podremos utilizar y compartir con las diversas aplicaciones que hagamos en PHP. Como bien sabemos, ASP posee sus objetos COM y ColdFusion tiene sus etiquetas personalizables, pero no se compara con la rapidez con la que podemos agregar funciones al código abierto de PHP.

Algunas personas argumentan que los lenguajes de scripts sirven únicamente para desplegar información de una base de datos. Pero PHP no es solamente un lenguaje de bases de datos, sino que sirve para muchas otras cosas como la generación dinámica de gráficos, la comunicación con redes a través de diversos protocolos (IMAP[13], SNMP[14]) y XML[15]. Claro que Perl hace todo esto también pero algunos programadores consideran que Perl es más complejo de aprender además de que no es considerado tan robusto para soportar una aplicación web muy grande.

Hablando de Bases de Datos, PHP® puede interactuar con Sybase®, Oracle®, Informix®, Solid®, Postgres®, y aún con MSSQL®.

PHP® es considerado multiplataforma. Siguiendo con el modelo de hacer algo sin costo alguno, se puede desarrollar una aplicación completa usando PHP® desde una computadora con MS® WINDOWS® a modo de servidor de producción. Aunque algunas cosas no trabajan con la versión de Win32, se piensa en ese momento en las plataformas como Unix para resolver estos problemas.

[13] Abreviación de las siglas en inglés de Internet Message Access Protocol, que es un protocolo para la obtención de mensajes de correo electrónico.
[14] Abreviación de las siglas en inglés de Simple Network Management Protocol, que es un conjunto de protocolos para el manejo de redes complejas.
[15] Abreviación de las siglas en inglés de Extensible Markup Language, el cual permite a los diseñadores crear sus propias etiquetas personalizables permitiendo la definición, transmisión, validación e interpretación de datos entre aplicaciones y organizaciones.

4.7 Comparación entre ASP y JSP

Como se ha mencionado anteriormente, solamente a través del tiempo se podrá decir cuál de los lenguajes de scripts será el más popular en el futuro. El que se presume pueda ser el más popular, es aquel que tiene más ventajas que desventajas. Hagamos una comparación entre 2 de los lenguajes más populares de scripts ASP y JSP.

Las similitudes entre ASP y JSP se basan en las funciones que realizan y no necesariamente de cómo las realizan. Ambos lenguajes fueron creados para crear aplicaciones interactivas para el web, y ambas tecnologías utilizan la lógica de Server side para llevar a cabo esta tarea.

ASP y JSP pueden llevar a cabo una misma tarea, cada uno con una manera diferente de hacerlo. Aquí es donde lo mejor de los dos lenguajes puede determinarse; a través de las diferencias que tienen entre si. Algunas de las diferencias más importantes pueden ser la independencia de la plataforma y del servidor, el lenguaje de programación y la utilidad en el uso.

Primeramente, hablemos de la independencia de la plataforma y del servidor, ya que es aquí donde JSP toma la ventaja. JSP puede correr en cualquier servidor web, mientras que ASP puede correr solamente en Microsoft Internet Information Server IIS® o Personal Web Server PWS®. La plataforma de ASP debe de ser Microsoft® Windows®, mientras que JSP puede correr en cualquier plataforma de las más populares, incluyendo en las plataformas basadas en Microsoft®. Esto ofrece flexibilidad a los usuarios.

Desde la perspectiva de los programadores, ASP puede tener una gran ventaja, ya que es un lenguaje muy sencillo. ASP se basa en el lenguaje BASIC, mientras que JSP es una adaptación del Lenguaje C. BASIC es un lenguaje mucho más fácil de aprender que el Lenguaje C, por lo que ASP es más fácil de aprender que JSP.

Otra ventaja que es muy importante mencionar es la explotación de los recursos informáticos de las empresas, por ejemplo ASP y JSP son compatibles con la mayoría de los DBMS. La compatibilidad con las bases de datos es muy importante en los Server Side Scripting ya que es muy fácil de manejar los datos en forma tabular de un inventario o de una lista de clientes.

5. Introducción a las Active Server Pages

5.1 Características de las Active Server Pages

Las ASP's (Active Server Pages) son un estándar de Microsoft® que permite la creación de aplicaciones Web interactivas, es decir, cuyo contenido sea dinámico. Para que una página web ASP (reconocida por la exensión .asp) tenga un contenido diferente dependerá de ciertos parámetros (por ej. La información almacenada en una base de datos, las preferencias del usuario, etc.) mientras que una página web "clásica" (cuya extensión pude ser .htm ó .html) siempre mostrará la misma información.

ASP es en realidad un lenguaje de script (un lenguaje interpretado) que se ejecuta del lado del servidor (de igual forma que los scripts de PHP, CGI, ...) y no del lado del cliente (como los scripts hechos en JavaScript o los applets de JAVA que se ejecutan en el navegador de la persona que consulta el sitio en cuestión).

Las ASP's se integran a una página web en HTML con la ayuda de ciertas marcas especiales que permiten que el servidor Web sepa que el código entre estas marcas debe ser interpretado con el fin de reenviar el código HTML al navegador del cliente.

De esta forma es que las Active Server Page se encuentran en una arquitectura de 3 capas, esto quiere decir que un servidor Web que soporte las Active Server Pages puede servir de intermediario entre el navegador del cliente y una base de datos permitiendo un acceso transparente gracias a la tecnología **ADO** (Actives Data Object), que provee los elementos necesarios para la conexión de un sistema gestor de bases de datos y a la manipulación de datos gracias al lenguaje SQL.

La ***imagen 5.1-1*** nos muestra la representación gráfica de la arquitectura de trabajo de las ASP's.

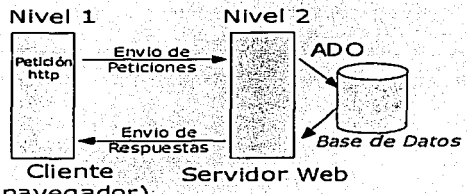

Imagen 5.1-1

Las ASP's fueron creadas básicamente para funcionar sobre el Servidor Web de Microsoft® llamado Microsoft® IIS® (Internet Information Server). Este servidor Web fue lanzado por Microsoft® en 1996, tiene la ventaja de que es gratuito y trabaja sobre Windows NT®, posteriormente se implantó también esta tecnología en Windows 98® sobre el servidor Personal Web Server PWS®.

Sin embargo, esta tecnología propia de Microsoft® se encuentra disponible hoy en día para otras plataformas, es decir, para otros servidores Web que no sean los de Microsoft®. En un principio el servidor Netscape® FastTrack® fue de los primeros en soportar las ASP's, otro servidor que da soporte a las ASP's es Apache® con el módulo Apache::ASP, que hace posible la creación de sitios Web utilizando la tecnología ASP bajo numerosas plataformas como (Unix®, Linux®, PowerPC®, etc.).

Las Active Server Pages pueden ser programadas en diferentes lenguajes de programación (Visual basic, Perl, Lenguaje C ++, Java, JavaScript, etc.) lo cual aumenta las posibilidades de utilizar el potencial que las ASP's ofrecen.

5.2 Interpretación del código ASP por el servidor

Un script ASP es un simple archivo de texto que contiene instrucciones escritas con los caracteres ASCII 7 bits (Caracteres no acentuados) que se incluyen dentro de un código HTML con la ayuda de marcas especiales. Este archivo debe tener la extensión ".asp" para poder ser interpretado por el servidor.

Los pasos que se llevan a cabo para la interpretación de una página ASP son:

- El servidor reconoce que se trata de un archivo ASP, gracias a su extensión.
- El servidor lee el archivo asp.
- Cuando el servidor encuentra una marca que le indica que las líneas siguientes son código ASP pasa a modo ASP, esto quiere decir que en vez de sólo leer las instrucciones, ahora las ejecuta.
- Cuando el servidor encuentra una instrucción, éste la transmite al interprete.
- El interprete ejecuta la instrucción, después envía las salidas temporales al interprete.
- Al final del script, el servidor transmite el resultado al cliente (el navegador).

El código ASP almacenado en el servidor nunca es visto directamente por el cliente ya que la página ASP que el cliente pide es interpretada como ya se dijo por el servidor y no por el cliente. De este modo ninguna modificación podrá hacerse por los navegadores.

5.3 Implantación del código ASP en el HTML

ASP se presenta de cierto modo como una extensión del HTML como los SSI (Server Side Include) que son comandos incrustados dentro del código HTML y que son interpretados por el servidor.

Con el fin de definir los scripts dentro del código HTML y que serán interpretados por el servidor, ASP define una nueva marca o tag HTML: **<% %>** .

Como ya se ha mencionado, al interior de estas marcas los scripts pueden ser escritos en diferentes lenguajes como:

- VBScript.
- Jscript.
- JavaScript.
- Perl.
- Java.
- C++
- etc.

Estos scripts, una vez que han sido interpretados por el servidor tendrán por efecto el código HTML enviado al navegador. Con esto volvemos a comprobar que el tratamiento de la información efectuada a nivel del servidor no son visibles en el código resultado o enviado.

5.4 Ejemplo básico de un Script ASP

Antes de comenzar debemos saber que elementos necesitamos para llevar a cabo nuestro ejemplo básico, los demás ejemplos que se presentan a lo largo de este capitulo y en general para llevar a cabo cualquier script ASP. No necesariamente se tiene que poseer una gran infraestructura de Hardware o Software para poder comenzar con las ASP's, a continuación, veremos como podemos comenzar con las ASP's utilizando nuestra propia computadora.

Primeramente necesitamos tener acceso a un servidor web capaz de soportar(interpretar) los scripts ASP, por ejemplo, el Personal Web Server® de Microsoft® que se puede instalar con Windows® 98® o el Internet Information Server® de Microsoft® que se puede instalar con Windows® XP®. Para una mayor referencia en el ***Apéndice A*** podemos observar los pasos para instalar estos servidores web en nuestra computadora.

A continuación necesitamos la ayuda de algún editor de textos para capturar nuestros scripts, este puede ser por ejemplo, el Notepad® o el Wordpad® de Windows®.

Finalmente necesitamos algún cliente de FTP (File Transfer Protocol) que nos permita enviar nuestros scripts al servidor, en nuestro caso, nuestra computadora. Un ejemplo de un cliente FTP puede ser, WS FTP®. Para una mayor referencia en el ***Apéndice B*** podemos observar los pasos para instalar este cliente de FTP, así de cómo configurarlo para conectarse a nuestro servidor web.

Para visualizar los resultados de nuestros scripts necesitamos únicamente cualquier navegador ya sea Microsoft® Explorer® o Netscape® Navigator®.

Para comenzar veremos un script ASP escrito en VBScript, recordemos que nuestro archivo debe tener la extensión ".asp":

Archivo: ejemploBasico.asp

```
<%@ LANGUAGE = "VBSCRIPT"%>
<HTML>
<HEAD>
<TITLE> Ejemplo de script ASP </TITLE>
</HEAD>
<BODY>
        <% FOR i = 1 to 10 %>
                Bienvenidos al ejemplo de Script ASP
        <% NEXT %>
</BODY>
</HTML>
```

Una vez que hemos capturado este ejemplo y que lo hemos transferido a nuestro servidor web, podemos observar el resultado en nuestro navegador introduciendo el siguiente URL:

http://servidor/ejemploBasico.asp

donde: ***servidor*** **es el nombre de nuestro servidor web.**

Este script tiene por resultado repetir 10 veces la cadena

"Bienvenidos al ejemplo de Script ASP".

Ahora veremos el mismo ejemplo, pero escrito en JavaScript:

```
<%@ LANGUAGE = "JAVASCRIPT"%>
<HTML>
<HEAD>
<TITLE> Ejemplo de script ASP </TITLE>
</HEAD>
<BODY>
        <% for (i=1;i<=10;i++) { %>
                Bienvenidos al ejemplo de Script ASP
        <% } %>
</BODY>
</HTML>
```

El comando <%@ *LANGUAGE* %> situado al inicio del archivo ".asp" permite definir el lenguaje de script a utilizar, es decir el lenguaje en el cual los scripts son escritos en nuestra página. La sintaxis de este comando es la siguiente:

```
<%@ LANGUAGE = "Lenguaje de Script"%>
```

Lenguaje de Script representa evidentemente el lenguaje de script que utilizará nuestro archivo, generalmente es VBScript, un lenguaje de script de Microsoft®.

5.5 Los Objetos de ASP

Las Active Server Pages se basan en objetos manipulados por el servidor que permiten la manipulación básica de los datos. Los 6 objetos de base son:

- **Application:** Representa el sitio. Permite la manipulación de las variables, constantes, etc. necesarios para el funcionamiento del sitio y de memorizar dichos valores.
- **ObjectContext:** Designa la transacción actual. Se encarga de administrar el desarrollo de las transacciones. Dichas transacciones son generadas por el MTS® (Microsoft Transaction Server)
- **Request:** Permite manipular la información que proviene del cliente. Es decir, permite la recuperación de los valores de los campos que se encuentran en el formulario que el usuario capturo y envio.
- **Response**: Representa el resultado a mostrar en el navegador. Es decir, se encarga de enviar las respuestas al cliente(Navegador).

- **Server:** Representa al Servidor, permite manipular o administrar los parámetros, así como de instanciar los objetos creados por el usuario.
- **Session:** Representa al usuario. Permite conservar los datos (preferencias generalmente) relativas al usuario de una página del sitio, es decir, permite conservar la información de una página a otra.

5.6 La Estructura de un objeto ASP

Los objetos ASP constituyen esencialmente el motor de scripts ASP, es decir, que los objetos ASP son los principales elementos que reagrupan las propiedades (valores) y los métodos útiles dentro de los scripts.

Un objeto ASP se constituye de tres tipos de entidades, que son:

Las colecciones: Una estructura de datos (una tabla ordenada) que contiene un conjunto de valores identificados por una clave. Cada objeto puede tener varias colecciones de variables.

Un valor de una colección de un objeto es accesible a través de la siguiente sintaxis:

```
objeto.colección("clave")
```

Las propiedades: Un valor específico directamente accesible. Para acceder a una propiedad de un objeto ASP se utiliza la siguiente sintaxis:

```
objeto.propiedad
```

Los métodos: Las funciones estándar asociadas a un objeto, permitiendo la manipulación de los valores pasados como argumento. La sintaxis de un método es la siguiente:

```
objeto.método(argumentos)
```

La ***imagen 5.6-1*** nos muestra la representación gráfica de un objeto ASP.

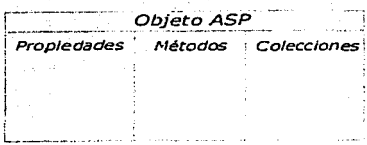

Imagen 5.6-1

La manipulación de las propiedades y métodos de los objetos internos permiten la recuperación de información en una petición, así como la creación y envío de la respuesta al navegador. La ***imagen 5.6-2*** nos muestra la representación gráfica del modelo de los objetos ASP dentro del esquema de la arquitectura cliente/servidor:

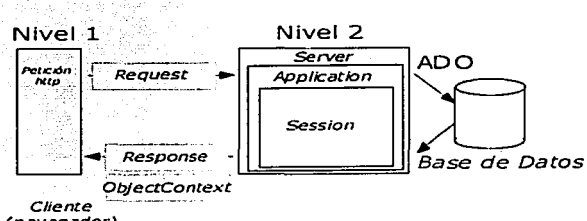

Imagen 5.6-2

5.7 Objeto Response

La tarea principal de un objeto *Response* es la de crear la respuesta que va a ser enviada al navegador, es decir, la página Web solicitada por el cliente.

El objeto *Response* permite en realidad manipular conjuntamente a la información que va a ser enviada al navegador.

El objeto *Response* contiene una sola colección y un gran número de propiedades y métodos:

Colecciones	Métodos	Propiedades
Cookies	Buffer	AddHeader
	CacheControl	appendToLog
	CharSet	BinaryWrite
	ContentType	Clear
	Expires	End
	ExpiresAbsolute	Flush
	isClientConnected	Redirect
	Status	Write
	PICS	

La mayoría de las propiedades y métodos del objeto *Response* corresponde a los campos de la respuesta enviada al navegador.

5.7.1 Envío de datos al navegador

Para enviar texto al navegador en un código ASP, se necesita usar la propiedad *write* del objeto *Response*, a continuación un pequeño ejemplo sobre la propiedad *write*:

```
<%@ LANGUAGE = "VBSCRIPT"%>
<HTML>
<HEAD>
<TITLE> Ejemplo de script ASP </TITLE>
</HEAD>
<BODY>
        <% Response.write(http://www.yahoo.com) %>
</BODY>
</HTML>
```


La Salida de este ejemplo sería la siguiente:

http://www.yahoo.com

Este script es totalmente inútil en una página HTML simple, ya que este resultado puede ser obtenido fácilmente sin tanta codificación.

Lo interesante de este método es poder mostrar el contenido o valor de las variables así como de cadenas, a continuación otro pequeño ejemplo de la propiedad *write* en donde se hace uso de variables:

```
<%@ LANGUAGE = "VBSCRIPT"%>
<HTML>
<HEAD>
<TITLE> Ejemplo de script ASP </TITLE>
</HEAD>
<BODY>
        <% For i=1 to 10
                Response.write(Cuenta regresiva: " & 10 – i & "<br>")

        <% Next %>
</BODY>
</HTML>
```

La salida del ejemplo anterior sería la siguiente:

Cuenta regresiva: 9
Cuenta regresiva: 8
Cuenta regresiva: 7
Cuenta regresiva: 6
Cuenta regresiva: 5
Cuenta regresiva: 4
Cuenta regresiva: 3
Cuenta regresiva: 2
Cuenta regresiva: 1
Cuenta regresiva: 0

5.7.2 Envío de Cookies al navegador

El objeto *Response* además de enviar una página web al navegador del cliente, también puede enviar cookies al navegador, una cookie es un mensaje enviado del servidor web hacia el navegador. El navegador almacena este mensaje en un archivo de texto en forma de pares nombre/valor. El mensaje almacenado en el cliente es enviado al servidor cada vez que el navegador haga una petición de una página.

La principal tarea de una cookie es la de identificar usuarios y la posibilidad de preparar páginas web personalizadas. Por ejemplo, cuando se accesa a una página web que envía cookies, ésta por lo general posee una forma donde se llenan datos como el nombre, dirección, etc. esta información se almacena en una cookie y se envía al navegador el cual almacena esta información para su uso posterior.

La siguiente vez que se accesa a la página web nuestro navegador envía la cookie al servidor web y este puede usar esta información para presentar una página personalizada con un mensaje de bienvenida con su nombre.

El objeto *Response* provee la colección **cookies** para efectuar las operaciones de envío de cookies al navegador.

El protocolo http permite especificar los valores de los cookies en las cabeceras http, el envío de cookies al navegador en la página ASP debe hacerse antes de cualquier envío en el cuerpo de la respuesta.

Por ejemplo para almacenar un valor en una cookie llamada CCMCookie, es suficiente con usar la siguiente sintaxis:

```
<%@ LANGUAGE = "VBSCRIPT"%>
<% response.cookies("CCMCookie") = "Valor" %>
```

Para almacenar varios valores asociados a los índices de una cookie llamada CCMCookie, es suficiente con usar las siguientes líneas:

```
<%@ LANGUAGE = "VBSCRIPT"%>
<% response.cookies("CCMCookie") ("Index1") = "Valor1" %>
<% response.cookies("CCMCookie") ("Index2") = "Valor2" %>
<% response.cookies("CCMCookie") ("Index3") = "Valor3" %>
```

En realidad la cookie creada anteriormente tendrá una duración limitada a la utilización del navegador, esto significa que será borrada cuando el navegador se cierre. Para remediar esto es necesario definir la propiedad *expires* que define la fecha límite de vida de una cookie.

```
<% response.cookies("CCMCookie").expires = #4/18/2003#%>
```

A demás de la propiedad *expires*, la colección Cookies posee otras propiedades:

- ***domain*** define el nombre del servidor por el que los valores de la cookie son accesibles.
- ***path*** define el camino en el servidor por el que los valores de la cookie son accesibles.
- ***secure*** permite indicar que la cookie no puede ser enviada mas que por una conexión segura (SSL, S-HTTP, ...).

5.8 El Objeto Request

La tarea del Objeto *Request* es la de obtener la información enviada del navegador hacia el servidor, es decir, la página web pedida por el cliente.

El Objeto *Request* tiene diversas colecciones, una propiedad y un método:

Colecciones	Métodos	Propiedades
ClientCertificates	TotalBytes	BinaryRead
Cookies		
Form		
QueryString		
ServerVariables		

La mayoría de las propiedades y de los métodos del Objeto *Request* corresponden a las funciones o propiedades para manipular los campos de la petición.

5.8.1 La recepción de Datos

Para entender como utilizar el Objeto *Request* es necesario saber la manera en la que los datos son enviados desde el navegador.

El envío de datos a un script se hace por medio de un formulario HTML.

Los formularios HTML se hacen gracias a la etiqueta *<FORM>* cuyo formulario contiene botones, campos, listas, opciones que tienen asociados valores, funciones y finalmente un botón que enviará toda esta información de manera conjunta a un script indicado según el atributo *Action* de la etiqueta *<FORM>* según el método *GET* o *POST* para el envío de datos. Cada elemento del formulario debe tener un nombre único de tal manera que el valor asociado a dicho elemento forme un par nombre/valor del tipo:

```
Nombre_del_elemento=valor
```

El conjunto de pares nombre/valor son separados por un ampersand (&), de este modo el envío de un formulario crea una cadena de la siguiente forma.

```
campo1=valor1&campo2=valor2&campo3=valor3
```

El envío de esta cadena será diferente según el método utilizado para el envío, existen 2 formas para el envío de un formulario que son el método *GET* y el método *POST*.

El método Get permite enviar a los elementos de un formulario a través del URL del script poniendo los pares nombre/valor al URL del script, separado por un signo de interrogación, quedando de la siguiente forma:

```
http://nombre_del_servidor/cgi-bin/script.cgi?campo1=valor1&campo2=valor2...
```

Sin embargo, la longitud de la cadena de un URL esta limitada a 255 caracteres, así que los datos que se encuentren fuera de este límite se perderán irremediablemente. Además que esto crea un URL sobrecargado en la barra del Navegador, además de se puede revelar información sensible como una clave de acceso.

El método POST es una buena alternativa al método *GET*. Este método codifica la información de igual forma que el método *GET*(codifica en el URL la información en pares nombre/valor) pero solo que POST envía los datos después de la cabecera http, dentro de un campo llamado Cuerpo de la petición. De este modo la cantidad de datos enviados no esta limitada.

5.8.2 La Colección QueryString

La colección *QueryString* permite recuperar el valor asociado a un campo, cuando los datos son enviados a través del método GET, según la siguiente sintaxis:

```
<%@ LANGUAGE = "VBSCRIPT" %>
<% Request.QueryString ("Campo") %>
```

A continuación veremos un ejemplo que nos ayudará a entender esta colección.

Formulario que envía los datos al Script:

```
<form name="form1" method="get" action="recepcionForm.asp">
        Forma de ejemplo para el envio de informacion a un archivo .ASP
        <br>
        Introduce tu Nombre:  <input name="nombre" type="text" id="nombre">
        <br>
        Introduce tu Telefono:<input name="telefono" type="text" id="telefono">
        <br>
        Sexo: <input type="radio" name="RadioSexo" value="Masculino">
        Masculino
        <input type="radio" name="RadioSexo" value="Femenino">
```

```
        Femenino

        <input type="submit" name="Submit" value="Enviar">
        <input type="reset" name="Submit2" value="Borrar">
</form>
```

La ***imagen 5.8.2-1*** nos muestra la salida del código anterior:

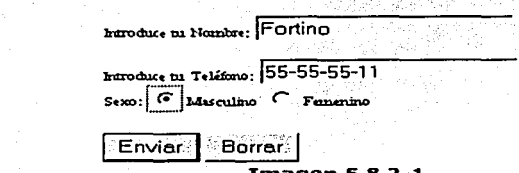

Imagen 5.8.2-1

Para mostrar todos los campos de un formulario, es posible utilizar la cláusula For Each de VBScript, que permite examinar o recorrer el conjunto de datos de una colección. A continuación veremos el script que muestra el conjunto de valores asociados a los campos del formulario después de que son enviados al servidor.

Archivo: recepcionForm.asp

```
<html>
<head>
<title>Recepción de Datos con QueryString</title>
</head>
<body>
        <%@LANGUAGE="vbscript"%>
        <%
        For Each Campo in (Request.QueryString)
                Response.write(Campo + " = ")
                Response.write(Request.QueryString(Campo))
                %>
                <br>
        <%Next%>
</body>
</html>
```

La salida del script anterior es:

```
Nombre = Fortino
Telefono = 55-55-55-11
RadioSexo = Masculino
```

5.8.3 La Colección FORM

Como ya habíamos dicho la colección *QueryString* permite recuperar de manera simple los datos enviados a un script ASP por medio de un URL (Con el método GET), en este caso la colección *Form* permite manipular los datos enviados por un formulario utilizado el método POST.

La colección *Form* permite recuperar el valor asociado a un campo según la siguiente sintaxis:

```
<%@ LANGUAGE="VBSCRIPT"%>
<% Request.Form("Campo") %>
```

A continuación veremos un ejemplo que nos ayudará a entender esta colección.

Formulario que envía los datos al Script:

```
<form name="form1" method="post" action="recepcionForm.asp">
```

Del formulario que usamos anteriormente, la línea anterior es la única que cambia, es decir, lo único que hay que modificar es el método que utilizamos para enviar los datos, en nuestro caso, es el método "Post".

A continuación veremos el mismo script que muestra el conjunto de valores asociados a los campos de un formulario pero ahora con la ayuda de la colección *Form*:

```
<html>
<head>
<title>Recepción de Datos con QueryString</title>
</head>
<body>
        <%@LANGUAGE="vbscript"%>
        <%
        For Each Campo in (Request.Form)
                Response.write(Campo + " = ")
                Response.write(Request.Form(Campo))
                %>
                <br>
        <%Next%>
</body>
</html>
```


La salida del script anterior es:

```
Nombre = Fortino
Telefono = 55-55-55-11
RadioSexo = Masculino
```

5.8.4 La Colección Cookies

La colección Cookies permite recuperar valores de una cookie, es decir, un archivo que se encuentra almacenado en alguna parte, por ejemplo el disco duro del cliente. Dicho archivo contiene los datos enviados por un servidor (una Cookie puede ser creada por el objeto Response).

El acceso a una cookie se hace de la siguiente manera:

```
<% LANGUAGE=VBSCRIPT"%>
<% For Each Elemento in Request.Cookies %>
<%
        Response.write(Elemento + " = ")
        Response.write(Request.Cookies(Element)) %>
<% BR %>
<% Next %>
```

5.8.5 La Colección ServerVariables

La colección *ServerVariables* del objeto *Request* contiene las cabeceras HTTP de la petición, algunas veces proporciona información muy útil para los visitantes.

La sintaxis para recuperar las cabeceras es la siguiente:

```
Request.ServerVariables("NOMBRE DE LA CABECERA")
```

Las principales cabeceras son las siguientes:

Nombre de la Cabecera	Descripción
ALL_HTTP	Método utilizado por el cliente (POST o GET)
CONTENT_TYPE	Tipo de contenido en el cuerpo de la petición (por ejemplo, text/html). Ver tipos MIME
METHOD	Tipo de método utilizado por el cliente (POST o GET)
REFERER	URL de liga a partir de la cual la petición a sido efectuada
REMOTE_ADDR	Dirección IP del cliente
HTTP_ACCEPT_LANGUAGE	Lenguaje entendido por el navegador (Inglés por default)
HTTP_USER_AGENT	Cadena proveedora de información sobre el cliente, como el nombre y la versión del navegador, el sistema operativo.

Cuando se hace la cuenta de cuantos visitantes han visitado cierto sitio, es también interesante guardar la dirección IP de los visitantes y también el poder contar el número de IP's almacenadas.

A continuación se presenta un código que muestra la IP del visitante:

```
<% LANGUAGE = "VBSCRIPT" %>
<%
        IP = Request.ServerVariables("REMOTE_ADDR")
        Response.write("Su dirección IP es: " + IP) %>
<% BR %>
```

5.9 El objeto Server

El objeto Server posee una única propiedad y cuatro métodos.

La propiedad *Server.ScriptTimeout*, especifica cuántos segundos puede ejecutarse una secuencia de comandos antes de que el servidor termine. El valor predeterminado de ScriptTimeout lo configura un administrador en el servidor, éste tiempo suele ser 90 segundos. El código ASP puede alargar (pero no acortar)este intervalo.

El método *Server.CreateObjet* crea cualquier tipo de objeto no incorporado que requiera una página ASP. Su sintaxis se presenta a continuación:

Set variabiable = Server.CreteObject("TipoObjeto")

Ejemplo:

Set miObjeto = Server.CreateObject("ADODB.Connection")

Aquí se crea un objeto de tipo Conexión a una BD (ADODB.Connection) que se asigna a la variable *miObjeto*.

5.10 El objeto Application

Los archivos ASP deben residir en una carpeta que sea ejecutable para que puedan ejecutarse. Y para que una carpeta sea ejecutable un administrador debe configurarla como tal. Una vez que una carpeta es ejecutable también lo serán de manera predeterminada todas sus subcarpetas.

En la jerga de ASP una aplicación (application) consta de todos los archivos ASP que existen dentro de un único árbol de carpetas ejecutable. El objeto *Application* proporciona una área donde todas estas páginas pueden intercambiar datos entre ellas, y un medio por el cual es posible programar el código para que se ejecute siempre que la aplicación empieza o termina.

5.11 El objeto Session

La primera vez que un visitante web entra en una aplicación (solicita una página ASP) el servidor web crea un objeto *Session* para el mismo. Estos objetos *Session* son persistentes, lo que significa que sigue existiendo hasta que deja de haber actividad por parte del visitante durante un intervalo especificado (generalmente 20 minutos). Mientras el objeto existe, éste proporciona una área de almacenamiento para los datos pasados desde la ejecución de una página hasta la siguiente, o de una página a otra.

Un *SessionID*(identificador de sesión) identifica de manera exclusiva cada objeto *Session*. El servidor web transmite el *SessionID* al navegador como una cookie y le indica que devuelva dicha cookie *SessionID* con todas las solicitudes que realice a la misma aplicación. El servidor web emplea la cookie SessionID con todas las solicitudes que realice a la misma aplicación. El servidor web emplea la cookie *SessionID* para asociar el mismo objeto *Session* con todas las peticiones del mismo visitante.

Si el visitante no envía una solicitud dentro de un período de tiempo determinado, el servidor web descarta el objeto Session, si el visitante regresa más adelante el servidor crea un nuevo objeto *Session* (vació).

5.12 ObjectContext Object

Este objeto se utiliza para abortar (Abort) o llevar a acabo con éxito (Commit) una transacción manejada por el Microsoft Transaction Server ® (MTS), que es inicializado por un script contenido en una página ASP.

Cuando una página ASP tiene la directiva: @TRANSACTION, esta página corre en una transacción cuyo proceso no termina hasta que la transacción termina con éxito o falla.

La sintaxis de este objeto es:

```
ObjectContext.método
```

El método *SetAbort* hace abortar la transacción. Esto causa que el MTS prevenga cualquier actualización a los recursos a los que se ha conectado (Bases de Datos) durante la primera fase de la transacción.

El método *SetComplete* no necesariamente significa que la transacción se complete. La transacción se completa únicamente si todos los componentes de la transacción se llevan a cabo.

6. Caso Práctico

6.1 Presentación

Hoy en día las posibles aplicaciones de bases de datos para el web son al menos tan numerosas como las bases de datos. Si toda nuestra información puede residir en una base de datos, podremos crear páginas web para introducir, mantener y distribuir dicha información a otras personas. Además de que se puede hacer esto sin tener que distribuir ni instalar ningún software en especial, lo único que hay que instalar es cualquier navegador web.

Es evidente que se pueden utilizar bases de datos para web para el desarrollo de aplicaciones de negocios tradicionales como introducción de pedidos, inventarios, facturación, etc. Por otro lado también aquellas páginas estáticas que son repetitivas, pueden ser realizadas con bases de datos.

El desarrollo de una página web o aplicación web que acceda a una base de datos no necesariamente tiene que ser muy difícil de llevarla a cabo. La mayor parte de los casos sencillos requieren unas cuantas líneas de código de las cuales muchas de estas líneas son HTML. Obviamente las páginas que requieren de mayor funcionalidad lógicamente requerirán muchas más líneas de código.

Ya que hemos estado hablando sobre ejemplos, vamos a crear uno para entender todos los conceptos que hemos estado platicando. Muchas personas creen que no hay nada mejor que un buen ejemplo en concreto para entender las cosas, en este caso para poder entender los conceptos clave de esta nueva tecnología de bases de datos para el web.

El ejemplo que se presentará, será la creación de una base de datos que contendrá información de alumnos que aspiran realizar estudios en Europa. Toda la gestión de esta base de datos lógicamente debe ser por internet. Supongamos que esta aplicación es de alguna agencia que se dedica a canalizar a los alumnos que desean estudiar en el extranjero.

La aplicación trabajará mas o menos de la siguiente manera:

- Algún aspirante que esta buscando informes para realizar sus estudios en el extranjero encuentra en la página de la agencia una forma de llenado de datos para que la agencia le envíe o lo contacte para darle mayores informes sobre las carreras, las escuelas y los paises donde el alumno puede hacer sus estudios.
- Otra opción es que el aspirante acuda directamente a la agencia a pedir informes, por lo que la misma forma se llena con los datos del aspirante.
- Algunas veces existen cambios en la opinión de los alumnos en cuanto a que, en que escuela estudiar o que carrera realizar e incluso en las fechas de partida o de llegada por lo que se necesita que la aplicación sea capaz de buscar a algún aspirante y editar su información.
- La aplicación debe ser capaz también de hacer una búsqueda muy sofisticada para así poder hacer diversos informes, por ejemplo: cuántos aspirantes hubo en el mes, cuántos van a cierta ciudad, qué alumnos parten en una fecha en especial, qué aspirantes llenaron su forma en Internet, cuáles acudieron al centro de información, etc.
- Finalmente la aplicación debe ser capaz de borrar a los alumnos que ya no estén interesados o que hayan cumplido ya cierto tiempo en la base de datos o simplemente por que ya llevaron a cabo sus estudios en el extranjero.

Los Datos que se deben recabar del aspirante son:

Datos Personales
Nombre, Apellidos, Teléfono de Casa, Algún otro teléfono, Celular, Fax, e-mail, Calle y Número, Colonia, C.P, Ciudad, Estado, Fecha de Contacto, Contacto (quien lleno la hoja de captura), Fecha de última actualización y como se entero del servicio.

Programa del país de estancia
Tipo de curso(verano, semestre, año académico, prácticas, otros), Fecha de inicio del programa, Fecha fin del programa, Costo del programa, Notas del programa.

Estancia del alumno
Ciudad de estancia, Escuela, Dirección de estancia, Teléfono estancia, Fecha de llegada, Fecha de Salida, relac (que son las personas que están muy interesadas en los programas de estudios en el extranjero.

6.2 Diseño de la Base de Datos

A continuación, veremos como queda el diseño de la base de datos conforme a los datos que se necesitan recabar. La base de datos de nuestro ejemplo la realizáremos en Microsoft® SQL Server® versión 7, que es el Servidor de Bases de datos que utilizaremos para nuestro ejemplo. Para una mayor referencia en el ***Apéndice C*** podemos encontrar los pasos para instalar éste servidor de Bases de Datos, así como los pasos para configurarlo para crear nuestra Base de Datos de ejemplo.

Cabe mencionar que nuestro ejemplo es a grandes rasgos una muestra de la gestión de una base de datos con ASP, cuya validación de información se hace con JavaScript, y no un ejemplo de cómo hacer el análisis y diseño de una base de datos ni nada por el estilo. Es tan solo un pequeño ejemplo ilustrativo de esta nueva tecnología de bases de datos en WEB.

6.2.1 Diagrama E-R de la Base de datos

En la ***imagen 6.2.1-1*** podemos observar el diagrama E-R que representa a nuestra base de datos, con sus tablas y registros respectivamente. Este diagrama se obtuvo gracias a la herramienta para diagramas que viene incluida en el Microsoft® SQL Server® versión 7.

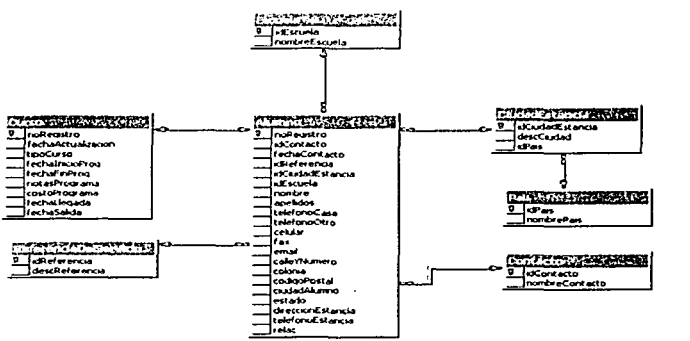

Imagen 6.2.1-1

A continuación se presenta una breve descripción de cada una de las tablas de nuestra base de datos de ejemplo:

Alumno: Es la tabla que contendrá los datos del alumno (Nombre, Teléfono, Dirección, etc.)
Curso: Es la tabla que contendrá los datos del curso al que el alumno desea inscribirse(Tipo de curso, Fecha de inicio del programa, etc.)
Escuela: Es una tabla que tiene almacenadas las posibles Escuelas donde el alumno puede realizar sus estudios.
ReferenciaDelServicio: Es un una tabla que tiene almacenados las referencias del servicio, es decir el famoso "¿Cómo te enteraste de nuestro servicio?" puede ser (Radio, TV, Embajada, etc.)
País: Es una tabla que contiene los países en donde se puede realizar un curso.

CiudadEstancia: Es una tabla que contiene las ciudades en donde se puede realizar un curso y como se puede observar esta depende del pais, para que no exista inconsistencia en la información, es decir, que se quiera estudiar en una ciudad en un pais inexistente o incoherente.
Contacto: Es una tabla que tiene los nombres de los contactos o asesores quienes proporcionan información a los alumnos aspirantes.

6.3 Creación de la Base de Datos en SQL Server

Antes de comenzar con la creación de nuestra base de datos, cabe mencionar, que lo que a continuación se presenta son únicamente los pasos para crear nuestra base de datos de ejemplo y no un manual de cómo utilizar SQL Server®.

Para la creación de la base de datos debemos abrir la aplicación ***"Administrador Corporativo"***, del conjunto de programas de la suite de SQL Server®. La ***imagen 6.3-1*** nos muestra como efectuar esto.

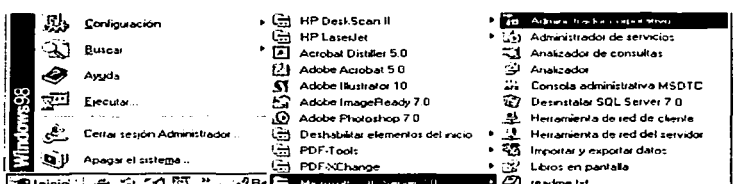

Imagen 6.3-1

Dentro del ***"Administrador corporativo"*** debemos realizar los siguientes pasos para la creación de la Base de Datos:

1) Debemos elegir el servidor en donde estará nuestra base de datos.
2) Elegir la opción Bases de datos del panel principal de SQL Server® y dar Click derecho para que aparezca la opción Nueva Base de datos. Aquí es donde ponemos todas las caracteristicas y propiedades de nuestra base de datos (ej. Tamaño, ruta de almacenamiento, etc.)
3) Ya que hemos creado nuestra base de datos, es necesario crear una cuenta de inicio de sesión, para poder acceder a nuestra base de datos que acabamos de crear. El procedimiento para crear un nuevo inicio de sesión, es similar a la creación de una nueva Base de datos. Del panel principal de SQL SERVER® se elige la opción inicio de sesión y ahí se crea el nuevo inicio de sesión.
4) También se debe crear una cuenta de usuario para cada inicio de sesión que hayamos creado. La creación de una cuenta es para tener mayor control de seguridad para el acceso a nuestra base de datos, es decir, que solo a través de una cuenta de usuario se pueda acceder a la base de datos. De igual forma del panel principal de SQL SERVER® se elige la opción cuenta de usuario y ahí se crea la nueva cuenta de usuario

Por el momento se ha creado la Base de Datos, pero falta todavía crear las tablas que conforman a nuestra base de datos. Para la creación de tablas debemos de abrir la aplicación llamada ***"Analizador de Consultas"*** de la suite de SQL Server®. La ***imagen 6.3-2*** nos muestra como efectuar esto.

Imagen 6.3-2

Dentro del ***"Analizador de Consultas"*** debemos realizar los siguientes pasos para la creación de las tablas de nuestra Base de Datos:

1) Proporcionar la cuenta que creamos anteriormente para entrar al servidor y a la base de datos.
2) Elegir la base de datos que creamos anteriormente para crear nuestras tablas.
3) Introducir el script de la *sección 6.3.1* para crear las tablas de nuestra base de datos. El script puede escribirse directamente en el ***"Analizador de Consultas"*** o se puede escribir en cualquier procesador de textos, guardándolo como un archivo con extensión ".sql" y después importarlo al ***"Analizador de Consultas"***.

6.3.1 Sript de Creación de la Base de Datos

El script de creación tiene la sintaxis que utiliza el Microsoft® SQL Server®, aún que no deja de ser lenguaje SQL. Recordemos que éste no es trabajo que enseñe como crear tablas con SQL, sino la gestión de bases de datos con las Active Server Pages.

El script para crear nuestra Base de Datos, es el siguiente:

```
CREATE TABLE Pais (
    idPais          smallint IDENTITY,
    nombrePais      varchar(40) NOT NULL,
    PRIMARY KEY (idPais)
)
go
```

```
CREATE TABLE Escuela (
    idEscuela          smallint IDENTITY,
    nombreEscuela      varchar(45) NOT NULL,
    PRIMARY KEY (idEscuela)
)
go


CREATE TABLE CiudadEstancia (
    idCiudadEstancia   smallint IDENTITY,
    deseCiudad         varchar(40) NOT NULL,
    idPais             smallint NOT NULL,
    PRIMARY KEY (idCiudadEstancia),
    FOREIGN KEY (idPais)
                REFERENCES Pais
)
go


CREATE TABLE ReferenciaDelServicio (
    idReferencia       smallint IDENTITY,
    deseReferencia     varchar(50) NOT NULL,
    PRIMARY KEY (idReferencia)
)
go


CREATE TABLE Contacto (
    idContacto         smallint IDENTITY,
    nombreContacto     varchar(35) NOT NULL,
    PRIMARY KEY (idContacto)
)
go


CREATE TABLE Alumno (
    noRegistro         int IDENTITY,
    idContacto         smallint NULL,
    fechaContacto      datetime NULL,
    idReferencia       smallint NULL,
    idCiudadEstancia   smallint NULL,
    idEscuela          smallint NULL,
    nombre             varchar(45) NOT NULL,
    apellidos          varchar(50) NOT NULL,
    telefonoCasa       char(20) NULL,
    telefonoOtro       char(20) NULL,
    celular            char(25) NULL,
    fax                char(20) NULL,
    email              varchar(65) NULL,
    calleYNumero       varchar(60) NULL,
    colonia            varchar(40) NULL,
    codigoPostal       char(15) NULL,
    ciudadAlumno       varchar(30) NULL,
    estado             varchar(35) NULL,
    direccionEstancia  varchar(80) NULL,
    telefonoEstancia   char(20) NULL,
    relac              char(2) NULL,
    PRIMARY KEY (noRegistro),
    FOREIGN KEY (idCiudadEstancia)
```

```
                REFERENCES CiudadEstancia,
    FOREIGN KEY (idEscuela)
                REFERENCES Escuela,
    FOREIGN KEY (idReferencia)
                REFERENCES ReferenciaDelServicio,
    FOREIGN KEY (idContacto)
                REFERENCES Contacto
)
go

CREATE TABLE Curso (
    noRegistro          int NOT NULL,
    fechaActualizacion  datetime NULL,
    tipoCurso           varchar(35) NULL,
    fechaInicioProg     datetime NULL,
    fechaFinProg        datetime NULL,
    notasPrograma       varchar(255) NULL,
    costoPrograma       varchar(11) NULL,
    fechaLlegada        datetime NULL,
    fechaSalida         datetime NULL,
    PRIMARY KEY (noRegistro),
    FOREIGN KEY (noRegistro)
                REFERENCES Alumno
)
go

SET IDENTITY_INSERT cuestionario ON
go
```


6.4 Creación del ODBC para la conexión a la base de datos.

Ya que hemos terminado con la creación de la base de datos, podremos ahora trabajar con ella. Para poder acceder a nuestra base de datos a través de páginas web, es necesario la creación de un ODBC. Un ODBC es un método estándar de acceso a bases de datos creado por Microsoft® con el que es posible acceder a los datos desde cualquier aplicación, en nuestro caso acceder a la base de datos desde páginas web. A continuación veremos la creación de nuestro ODBC:

1) Abrir el Panel de Control de Windows y elegir la opción Fuentes de Datos ODBC (32 bits).
2) Seleccionar la pestaña ***"DSN de Sistema"***. Seleccionamos esta opción por que queremos que nuestro ODBC funcione para nuestro sistema en general y no para un usuario en especial.
3) Seleccionar el botón Agregar e indicar el tipo de ODBC que necesitamos (Access®, SQL Server®, Oracle®, etc.), en nuestro caso necesitamos un ODBC para conectarnos a una base de datos en SQL Server®.
4)Crear el nuevo origen de datos, indicando el nombre de la base de datos, el servidor donde se encuentra dicha base de datos, el usuario y la clave (password).
5)Probar nuestro ODBC para verificar que está conectándose a la Base de Datos. Al término de la creación de un ODBC se presenta una opción que nos permite probar si el ODBC que acabamos de crear, funciona o no.

En la ***imagen 6.4-1*** podemos observar la pantalla donde creamos nuestro origen de datos ODBC.

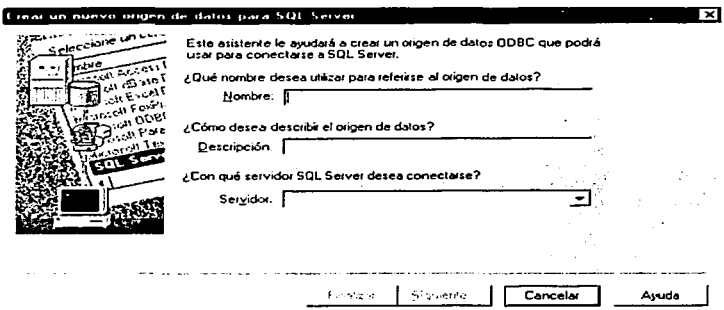

Imagen 6.4-1

6.5 Gestión de la Base de Datos

Como ya se mencionó en la sección **5.4 "Ejemplo básico de un Script ASP"**, antes que nada debemos estar concientes de los elementos que necesitamos para llevar a cabo nuestro ejemplo, recordemos que necesitamos un servidor para interpretar nuestros scripts ASP, un editor donde podamos escribir los scripts ASP, un cliente FTP para enviar nuestros scripts al servidor y un navegador para visualizar los resultados de nuestros scripts. El procedimiento para la instalación y configuración de algunos de estos elementos que necesitamos para trabajar, están descritos en los Apéndices que se encuentran al final del presente trabajo.

A continuación, en los siguientes subtemas se presentará una breve descripción de las tareas básicas para gestionar una base de datos (Inserción, Borrado, Actualización, etc. de registros de una base de datos) con Active Server Pages (ASP's) y la validación de los datos con JavaScript, conforme al ejemplo que se plateo al inicio de este capitulo.

Los Códigos completos de nuestro ejemplo se encuentran dentro del ***Apéndice D que está en el disco anexo a éste trabajo*** por lo que en cada subtema se presenta sólo la parte de código que es la representativa de ese subtema, por ejemplo, si se habla de conexión a la base de datos, sólo se presenta y describe brevemente el código ASP que hace posible conectarse a la base de datos.

Cuando se pensó en como se resolvería nuestro ejemplo, se descubrió que la información que se necesitaba almacenar en la base de datos tenía un comportamiento muy sencillo y casi siempre era el mismo, es decir, cuando un alumno llegaba a la agencia de información se capturaban sus datos y se registraban (almacenaban) en la base de datos, posteriormente se hacían búsquedas de alumnos que ya estaban registrados para actualizar sus datos o para contactar al alumno o simplemente para borrar al alumno de la base de datos.

Dado este comportamiento se pensó en que cada tarea se resolvería por separado, por lo que, se pensó en crear un archivo **".asp"** para cada tarea de nuestro ejemplo, es decir, un archivo para la captura, uno para el registro a la base de datos, otro para la búsqueda, otro para el borrado, etc. esto con el fin de simplificar la resolución de nuestro ejemplo, además de que, de esta forma se puede comprender mejor como las Active Server Pages nos pueden ayudar a gestionar la base de datos.

A continuación, se presenta la descripción de los archivos **".asp"** que se crearon para la gestión de la base de datos de ejemplo.

6.5.1 Captura de Aspirantes

La Captura de aspirantes es la primera tarea a resolver una vez que ya se creo la base de datos, la captura de aspirantes se encuentra en el **archivo "captura.asp"**. La Captura de aspirantes es la forma donde se ingresan los datos del aspirante para ser almacenados en la base de datos, para que posteriormente sean contactados por algún asesor.

A continuación, veremos como quedó el diseño de la pantalla de **"captura de aspirantes"** en el archivo ***"Captura.asp"***. La ***imagen 6.5.1-1*** nos muestra la primera parte que esta formada por la captura de los datos del alumno.

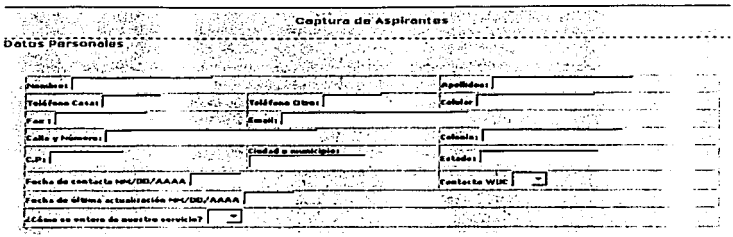

Imagen 6.5.1-1

Debemos recordar que cuando hacemos una forma para el envío de información al servidor, cada componente de dicha forma debe tener un nombre asociado a él. Por ejemplo, el componente asociado al nombre, se llama precisamente ***"nombre"***, el componente asociado al correo electrónico se llama ***"email"*** y así sucesivamente; esto facilitará muchas cosas, por ejemplo, si el componente se llama de igual forma que el campo en la base de datos no tendremos que estar buscando un nombre para el componente y otro para el campo en la base de datos y después no saber cual es cual o simplemente recordar que nombre es de la base y que nombre es de la forma. Además de que también nos facilitará las cosas en la validación de la información.

El siguiente paso es capturar los datos del curso al que el alumno desea inscribirse. En la ***imagen 6.5.1-2*** podemos observar como quedó el diseño de pantalla de esta parte de la hoja de captura. Cabe mencionar que la imagen **6.5.1-1** y la **imagen 6.5.1-2** son parte de una misma forma de envío, sólo que para fines ilustrativos se separó de esta manera. En la **imagen 6.5.1-2** se puede observar también que se presentan los botones de **"Registrar", "Limpiar Campos" y "Buscar alumnos"**, que son los botones que le dan funcionalidad a nuestra forma de envío, según la tarea que queramos hacer.

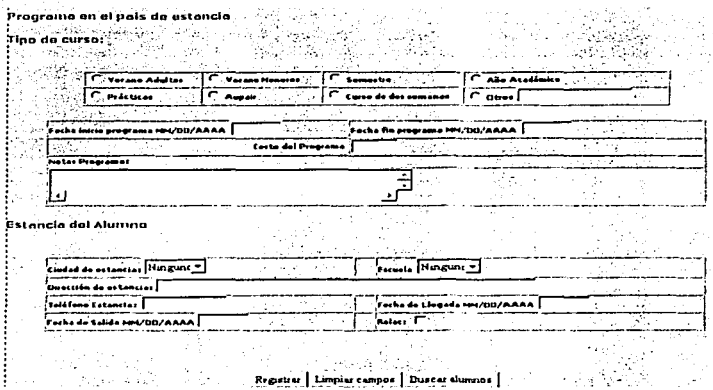

Imagen 6.5.1-2

Una vez que hemos capturado los datos del alumno, necesitamos validarlos antes de enviarlos al servidor, en el siguiente subtema veremos como llevar a cabo la validación de los datos.

6.5.2 Validación de la información con JavaScript

Es muy importante que la validación de los datos se haga antes de que la forma sea enviada al servidor, esto para ahorrar tiempo de procesamiento del servidor, imaginemos que la validación se hiciera como comúnmente se hacia, se mandaban todos los datos y el servidor se encargaba de validarlos y si existía algún error se notificaba al usuario, pero eso hacia perder tiempo. Hoy en día es mucho más sencillo validar los datos antes de enviarlos al servidor, es decir, la validación ahora es parte del cliente y no del servidor.

Cabe mencionar, que el botón **"Registrar"** no es un botón común y corriente que envía los datos de la forma al servidor, sino un botón que ejecuta una función en JavaScript que se encarga de validar la información y si toda la información es correcta, esta misma función envía los datos al servidor.

A continuación veremos como podría ser la validación de nuestra forma, sin antes mencionar que el siguiente script se encuentra dentro del mismo archivo **"Captura.asp"**:

```
<script language="JavaScript">
<!--

function validaciones(){ //Función para la validación de los datos de la forma
        var forma = document.forms[0]; //Obtención de los datos de la forma

        //Con la siguiente condición nos cercioramos de que exista el nombre en la
forma ya que este es un campo requerido, si no existe nos regresa al campo de
nombre para llenarlo.

        if(forma.nombre.value == ""){
                alert("El nombre es requerido");
                forma.nombre.focus();
                return;
        }

        //De la misma manera se validan los apellidos

        if(forma.apellidos.value == ""){
                alert("El apellido es requerido");
                forma.apellidos.focus();
                return;
        }
        //A continuación viene la validación de las fechas para checar que tengan
un formato valido y que sea a su vez una fecha existente, esta validación se auxilia
de otra función llamada validaFechas que más adelante se rebisará.

        if(forma.fechaContacto.value !=""){//Se revisa si se introdujo una fecha
                if(validaFechas(forma.fechaContacto.value,"de contacto")== 1){//Se
valida
                        forma.fechaContacto.focus();//Si hay algún error se regresa
al campo para reintroducir la fecha.
                        return;
                }
        }

//De igual manera se validan las siguientes fechas.

        if(forma.fechaActualizacion.value != ""){
                if(validaFechas(forma.fechaActualizacion.value,"de
actualizacion")==1){
                        forma.fechaActualizacion.focus();
                        return;
                }
        }
        if(forma.fechaInicioProg.value !=""){
                if(validaFechas(forma.fechaInicioProg.value,"inicio programa")
==1){
                        forma.fechaInicioProg.focus();
                        return;
                }
        }
        if(forma.fechaFinProg.value !=""){
                if(validaFechas(forma.fechaFinProg.value,"fin programa") == 1){
```

```
                        forma.fechaFinProg.focus();
                        return;
                }
        }
        if(forma.fechaLlegada.value !=""){
                if(validaFechas(forma.fechaLlegada.value,"de llegada") == 1){
                        forma.fechaLlegada.focus();
                        return;
                }
        }

        if(forma.fechaSalida.value !=""){
                if(validaFechas(forma.fechaSalida.value,"de salida") == 1){
                        forma.fechaSalida.focus();
                        return;
                }
        }

//Otro punto importante a validar, es el campo que contiene el correo electrónico,
que a continuación se valida.
        if(forma.email.value != ""){// Se revisa que el campo contenga información
                var correo = forma.email.value;
                if((correo.indexOf('@',0) == -1) || (correo.indexOf('.',0) == -1)){ //Se
checa que tenga un formato de mail.
                        alert("La direccion de correo tiene un formato invalido");
                        forma.email.focus();//Si hay algún error se regresa al
campo para recapturarlo.
                        return;
                }
        }

//si llega a este punto realiza el submit, es decir se envian los datos al archivo
registra.asp que es el encargado de meter los datos a la base de datos.
                forma.submit();

}
La siguiente función es la encargada de validar las fechas que introducimos, checa
que esten dentro de un rango de fechas, que se indroduzcan meses entre 1 y 12, la
posibilidad de un año biciesto que los dias no pasen mas de 31, etc.

function validaFechas(fecha, tipoFecha){
        var s = new String(fecha);
        var diag = s.indexOf('/',0);
        var diag2 = s.lastIndexOf('/');
        var mes = s.substring(0,diag);
        var dia = s.substring((diag + 1),diag2);
        var anio = s.substring((diag2+1),s.length);
        if(diag == diag2 ||diag < 1 || dia < 1 || dia > 31 || mes < 1 || mes > 12 ||
anio<1995 || anio > 2100 ||(mes == 2 && dia >29) || isNaN(dia) || isNaN(mes) ||
isNaN(anio)){
                alert("La fecha "+tipoFecha+" es incorrecta, el formato es:
MM/DD/AAAA");
                return 1;
        }else{
                return 2;
        }
}
//-->
</script>
```

6.5.3 Conexión a la Base de Datos con ASP a través del ODBC

Finalmente llegamos a la parte en que tenemos que conectarnos a la base de datos desde nuestra página asp con la ayuda del ODBC que hemos creado en la sección **"6.4 Creación del ODBC"**, a continuación revisáremos la conexión y la creación de algunos objetos necesarios para el trabajo con bases de datos, sin olvidar, que esta parte del código también esta dentro del archivo **"Captura.asp"**.

En nuestros scripts podemos usar o incluir otros archivos, es decir, incluir una especie de librerías que contienen código preescrito, o variables o constantes preconfiguradas, a continuación incluimos el archivo "adojavas.inc" que contiene algunos parámetros para el acceso a bases de datos.

```
<!-- #INCLUDE File="adojavas.inc" --> <%
```

Con la ayuda de los objetos ActiveX creamos un objeto *Connection* para conectarnos a la base de datos. Este objeto se asigna a una variable.

```
db = new ActiveXObject("ADODB.Connection");
```

Creamos una cadena con las características de la base de datos a la que queremos conectarnos, en nuestro caso, una BD en SQL Server con un ODBC llamado *"proyTesis"*, con un usuario llamado *"tesis"* y con password *"tesis"*. Esta cadena se asigna a la propiedad ConnectionString del objeto Connection que creamos anteriormente.

```
db.ConnectionString = "ODBC;Driver={SQL
Server};DSN=proyTesis;DATABASE=proyTesis;UID=tesis;PWD=tesis";
```

Abrimos la conexión hacia la base de datos.

```
db.open();
```

La referencia del servicio, el contacto, la escuela y la ciudad de estancia es información que se encuentra almacenada en la base de datos por lo que tenemos que crear objetos que almacenen temporalmente en la memoria dicha información.

Estos objetos son llamados *Recordsets*, que no son más que el resultado de un consulta, que se almacenan temporalmente en la memoria en forma de rejilla, en donde cada columna es un campo de nuestra consulta y cada renglón es un registro de nuestra consulta.

```
resRefServicio = new ActiveXObject("ADODB.RecordSet");
resContacto = new ActiveXObject("ADODB.RecordSet");
resEscuela = new ActiveXObject("ADODB.RecordSet");
resCiudadEstancia = new ActiveXObject("ADODB.RecordSet");
```

Asignamos una consulta a una cadena, según la información que deseamos obtener de la Base de Datos.

```
strQryRefServicio = "SELECT * FROM ReferenciaDelServicio";
strQryContacto = "SELECT * FROM Contacto";
strQryEscuela = "SELECT * FROM Escuela";
strQryCiudadEstancia = "SELECT idCiudadEstancia,deseCiudad,nombrePais
from CiudadEstancia, Pais where CiudadEstancia.idPais = Pais.idPais order by
nombrePais,deseCiudad";
```

Utilizamos el método "Open" del objeto Recordet para ejecutar las consultas, como podemos observar a este método le pasamos como parámetro la cadena que contiene la consulta que queremos realizar, el nombre de nuestro objeto connection,.

```
resRefServicio.Open(strQryRefServicio,db,adOpenStatic,adCmdText);
resContacto.Open(strQryContacto,db,adOpenStatic,adCmdText);
resEscuela.Open(strQryEscuela,db,adOpenStatic,adCmdText);

resCiudadEstancia.Open(strQryCiudadEstancia,db,adOpenStatic,adCmdText);
%>
```

6.5.4 Barrido de recordsets

Primeramente antes de continuar necesitamos saber qué son los ***recordsets***, hablando en términos de bases de datos, los recordsets son como una especie de rejilla cuyos renglones son cada registro resultado de una consulta y cada columna es un campo del registro. Por lo que es necesario para acceder a cada elemento un pequeño mecanismo que más adelante se presenta. Otro concepto que debemos conocer, es el término ***cursor***, que es una especie de apuntador hacia algún elemento de nuestro recordset, es decir, que a donde nuestro cursor este apuntando es el registro que podemos observar o manipular.

Como ya se ha dicho hay algunos valores que son obtenidos de la base de datos como la referencia del servicio, el contacto, la escuela y la ciudad de estancia. Posteriormente estos datos son almacenados en recordsets, para después desplegarlos en los "options" de nuestra forma de captura. El mecanismos del que hablábamos anteriormente los podemos observar a continuación:

```
while(!resEscuela.Eof){%>
    <option value="<%=resEscuela(0)%>"><%=resEscuela(1) %></option>
    <% resEscuela.MoveNext();
}
```

Como podemos observar, este mecanismo se auxilia de un ciclo While, esto es, por que necesitamos un ciclo que nos permita ir recorriendo cada registro de nuestro recordset. La condición para nuestro ciclo, utiliza el método "EoF" del recordset, el cual es un método que devuelve un valor verdadero o falso, según sea el caso, si se llega al último registro de nuestro recordset. También nos auxiliamos del método "MoveNext()" para cambiar nuestro cursor al siguiente elemento de nuestro recordset. Cada columna de un recordset se puede diferenciar a través de un índice, por ejemplo, la primera columna de nuestro recordset tiene el índice 0, la segunda tiene el índice 1, etc. Es por eso, que hacemos referencia a "resEscuela(0)" y a resEscuela(1) por que en la primera columna se encuentra el número de escuela y en la segunda se encuentra el nombre de dicha escuela.

6.5.5 Cerrado de la conexión a la Base de datos y de los recordsets

Cuando hemos terminado de trabajar con la base de datos es recomendable cerrar la conexión que hemos creado, así como cerrar los recordsets que se han utilizado, a continuación observaremos la forma de cerrar nuestra base de datos y los recordsets:

```
resRefServicio.close();
resContacto.close();
resEscuela.close();
resCiudadEstancia.close();
db.close();
```

Como podemos observar, tanto el Objeto "Connection" y el Objeto "Recordset", poseen un método llamado "close()" que nos permite terminar con la conexión y con los recordsets respectivamente.

Hasta este punto hemos trabajado dentro del archivo **"Captura.asp"**, muchas de las cosas que vimos se utilizan en los demás archivos "asp" de nuestro ejemplo práctico, como el barrido de recordsets, la validación de la información, la apertura de conexiones, el cierre de conexiones, etc.

6.5.6 Registro del Alumno en la BD en el archivo Registra.asp

El archivo **"Registra.asp"** es el encargado de registrar o almacenar la información de la forma de captura en nuestra base de datos. De igual manera se tienen que crear objetos para la conexión a la base de datos, además de crear objetos para la inserción de la información del aspirante en la Tabla Alumno y objetos para la inserción de la información del curso en la Tabla Curso, esto último recordando que en nuestra base de datos hay una tabla que almacena los datos del Alumnos (Tabla Alumno) y otra que almacena los datos del curso de dicho alumno (Tabla Curso).

A continuación se presentan las líneas de código más representativas del archivo **"Registra.asp"** que son las que hacen posible el registro de la información en la base de datos, recordemos que el código completo se encuentra en el ***Apéndice D que está en el disco anexo a éste trabajo*** para una mayor referencia.

El procedimiento para llevar a cabo el registro de información, es el siguiente:

Primero tenemos que crear objetos que nos ayuden a hacer el registro de la información, también necesitamos crear un objeto de tipo Recordset para saber cuál es el número o Identificador (ID) del último registro almacenado de la Tabla Alumno, es decir, saber cuál es el número que se le asignó al último alumno que se registró, esto es por que cuando se registra un alumno, la Tabla Alumno genera un identificador (ID) o número de registro automáticamente para identificar a ese alumno, mientras que la Tabla Curso toma al número de registro del último Alumno como identificador para registrar los datos del curso, esto para que concuerden los identificadores del Alumno y del Curso, es decir, que sean los mismos.

Una vez que se han creado estos objetos se comienza por crear las cadenas de inserción, es decir, los Querys para la inserción de la información a la base de datos. Estas cadenas son sentencias SQL (ej. INSERT INTO ALUMNO ...).

Posteriormente se comienzan a obtener los datos de la forma para agregarlos a nuestras cadenas de inserción, y una vez que han sido obtenida toda la información, se procede a insertar la información en la Tabla Alumno y en la Tabla Curso respectivamente.

A continuación se presenta el código representativo de este procedimiento de inserción:

```
//Creación de los objetos para la inserción

 cmdInsertaAlumno = new ActiveXObject("ADODB.Command");
 resObtenNewID = new ActiveXObject("ADODB.RecordSet");
 cmdInsertaCurso = new ActiveXObject("ADODB.Command");

//Necesitamos saber cuál es el ID del último alumno insertado para generar el
siguiente, por lo que creamos una cadena con este query.

 strQryObtenNewID ="SELECT MAX(noRegistro) from Alumno";

//Comenzamos a crear la cadena de nuestro Query de inserción.

strSQLAlumno = "INSERT INTO ALUMNO (idContacto, idReferencia,
idCiudadEstancia, idEscuela, nombre,apellidos, telefonoCasa, telefonoOtro, celular, fax,
email, calleYNumero, colonia, codigoPostal, ciudadAlumno, estado, fechaContacto,
direccionEstancia, telefonoEstancia,relac) VALUES (";

//Obtenemos los valores de los componentes de la forma para agregarlos a la
cadena de inserción

if(Request.Form.Item("nombre") != "") {  //Obtención del Nombre del Alumno
                strSQLAlumno = strSQLAlumno + "'" +
                filtraQuotesYComillas(Request.Form.Item("nombre")) + "',";
        } else {
                strSQLAlumno = strSQLAlumno + "'',";
        }

        if(Request.Form.Item("apellidos") != "") { //Obtención de los Apellidos
                strSQLAlumno = strSQLAlumno + "'" +
                filtraQuotesYComillas(Request.Form.Item("apellidos")) + "',";
        } else {
                strSQLAlumno = strSQLAlumno + "'',";
        }
// Y así sucesivamente con los demás valores de la forma, observemos que en esta
parte se utiliza una función llamada "filtraQuotesYComillas", que es una simple
función para eliminar las comillas y comas que a veces se agregan a los datos
cuando son enviados desde una forma. Los datos del curso son obtenidos y
almacenados en la cadena "strSQLCurso".

//Abrimos la base de datos.

db.open();

// Asignamos los querys de inserción a los objetos creados anteriormente y
llevamos a cabo la inserción.

// Inserción del Alumno en la Tabla Alumno

cmdInsertaAlumno.ActiveConnection = db;
cmdInsertaAlumno.CommandType = adCmdText;
cmdInsertaAlumno.CommandText = strSQLAlumno.toUpperCase();
cmdInsertaAlumno.Execute();
```

```
//Obtenemos el ID del último registro de la Tabla Alumno

resObtenNewID.Open(strQryObtenNewID,db,adOpenStatic,adLockOptimistic,adCmdT
ext);
newIDCurso = resObtenNewID(0)+"";
resObtenNewID.close();

strSQLCurso2 = "INSERT INTO CURSO (noRegistro,fechaActualizacion
tipoCurso,fechaInicioProg,fechaFinProg,costoPrograma,notasPrograma,fechaLlegada
fechaSalida) VALUES(";

//Inserción del Curso en la Tabla Curso.

strSQLCurso2 = strSQLCurso2 + newIDCurso +",";
strSQLCurso = strSQLCurso2 + strSQLCurso;
cmdInsertaCurso.ActiveConnection = db;
cmdInsertaCurso.CommandType = adCmdText;
cmdInsertaCurso.CommandText = strSQLCurso.toUpperCase();
cmdInsertaCurso.Execute();
db.close();

// Finaliza el proceso de inserción de la información.
```

6.5.7 Búsqueda de Aspirantes en el archivo Busqueda.asp

Una parte muy importante en una base de datos, es poder buscar ciertos registros almacenados en ella, nuestro ejemplo práctico presenta una parte que se encarga de buscar registros conforme a diversos criterios, el archivo **"Busqueda.asp"** es una simple forma que se encarga de recabar información para la creación de una cadena (query) de búsqueda. La forma se encarga de enviar los datos hacia el archivo **"Consulta.asp"** que es ahí donde se lleva a cabo la búsqueda. La ***imagen 6.5.7-1*** nos muestra como esta conformada esta forma de búsqueda.

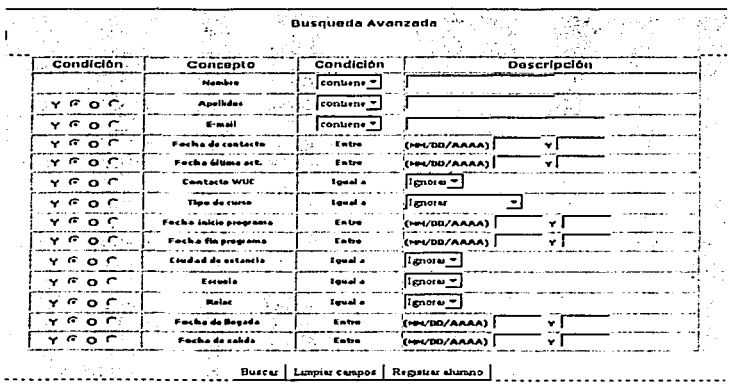

Imagen 6.5.7-1

Debemos recordar que es importante validar los datos antes de enviarlos al servidor, por lo que este archivo de búsqueda también tiene un script encargado de validar la información que va a ser enviada para formar la consulta de búsqueda, de hecho prácticamente es el mismo script que se uso en la forma de captura, solo con algunas pequeñas modificaciones simples que se pueden observar en el código completo del Archivo **"búsqueda.asp"**, que se encuentra en el ***Apéndice D que está en el disco anexo a éste trabajo.***

6.5.8 Formación de la Consulta en el Archivo consulta.asp

Recordemos que una consulta a una base de datos es la selección de ciertos registros que cumplen con ciertas condiciones.

Recordemos también que en SQL la sentencia para hacer una consulta es la palabra reservada **"SELECT"** cuya sintaxis básica es la siguiente: SELECT Campos FROM Tablas WHERE Condiciones.

En nuestro caso práctico, para llevar a cabo la consulta a la base de datos, debemos formar la cadena (query) de búsqueda con los datos que han sido enviados desde el archivo **"busqueda.asp"**, de hecho, con estos datos sólo formamos las condiciones de búsqueda.

A continuación se presenta la manera en que se va formando la cadena de búsqueda con los datos recibidos, la ejecución de la consulta y el despliegue de los registros que cumplen con la condición de búsqueda, recordando que el código completo de este archivo **"consulta.asp"** se encuentra en el ***Apéndice D que está en el disco anexo a éste trabajo.***

```
// Creamos los objetos para la conexión a la base de datos.
db = new ActiveXObject("ADODB.Connection");

// Utilizamos la propiedad "ConnectionString" del Objeto creado para la conexión
a la BD para indicarle que ODBC utilizamos, el nombre de la BD, el usuario y el
password de ese usuario en el SQL SERVER.

db.ConnectionString = "ODBC;Driver={SQL
Server};DSN=educeuropa;DATABASE=educeuropa;UID=educeuropa;PWD=izlj00zLPn
Y";
```

```
// Creamos un objeto Recordset para almacenar el resultado de nuestra consulta
de búsqueda.

resConsultaAlumno = new ActiveXObject("ADODB.RecordSet");

// Asignamos a la cadena "strSQL" el inicio de nuestra consulta de búsqueda.

 strSQL = "SELECT Alumno.noRegistro, Alumno.noRegistro, Alumno.nombre,
Alumno.apellidos, Curso.tipoCurso FROM Alumno, Curso WHERE
Alumno.noRegistro=Curso.noRegistro ";

// Variable que nos ayuda a saber si al inicio de nuestro query se pone un AND,
esto en caso de que cierto campo diferente al campo "nombre" sea el primer
elemento en el query de búsqueda, ya que no necesariamente se busca por el
nombre del alumno, que es el primer elemento en la forma de búsqueda del
archivo "busqueda.asp".

 var ponerAnd = true;

 // Aquí se empieza a obtener los valores de la forma del archivo "Busqueda.asp"

 //Se obtiene el Nombre.

if(Request.Form("nombre")+"." != ".") {
        ponerAnd = false; // Esto indica que "nombre" es 1er. elemento del query
        strSQL +="AND (Alumno.nombre LIKE '";
        if(Request.Form.Item("selNombre") == "contiene"){
                strSQL += "%" +
                filtraQuotesYComillas(Request.Form.Item("nombre")) +"%' ";
        }else{
                strSQL += filtraQuotesYComillas(Request.Form.Item("nombre")) +"'
                ";
        }
}

// Se obtienen los Apellidos
if(Request.Form("apellidos")+"." != ".") {

        // Si los "apellidos" son el primer criterio de búsqueda se ingresa a esta
        parte del if. Con los demás elementos de la forma (email, fecha de salida,
        contacto, ect.) se hace esta misma condición.
        if (ponerAnd){
                ponerAnd = false;
                strSQL += " AND (Alumno.apellidos LIKE '";
                if(Request.Form.Item("selApellidos") == "contiene"){
                        strSQL += "%" +
                        filtraQuotesYComillas(Request.Form.Item("apellidos"))
                        +"%' ";
                }else{
                        strSQL +=
                        filtraQuotesYComillas(Request.Form.Item("apellidos")) +"'
                        ";
                }
        }else{
        // Si hay más elementos en el query, antes que los apellidos, se ingresa a
        esta parte del if. De igual manera se validan los demás elementos de la
        forma (email, fecha de salida, contacto, etc.)
```

```
                    if(Request.Form.Item("rbApellidos") + "" == "AND"){
                            strSQL += " AND";
                    }else{
                            strSQL += " OR";
                    }
                    strSQL +=" Alumno.apellidos LIKE '";
                    if(Request.Form.Item("selApellidos") == "contiene"){
                            strSQL += "%" +
                            filtraQuotesYComillas(Request.Form.Item("apellidos"))
                            +"%' ";
                    }else{
                            strSQL +=
                            filtraQuotesYComillas(Request.Form.Item("apellidos")) +"'
                            ";
                    }
            }
    }

    // Y así sucesivamente con los demás elementos de la forma como el email, el
    contacto, la escuela, la ciudad de estancia, etc.

    //Al final siempre se concatena un paréntesis para completar la cadena (query) de
    búsqueda.

    if(!ponerAnd){
            strSQL+=")";
    }

    // Hasta este punto solo se han obtenido los datos para formar la cadena de
    búsqueda, ahora se procede a ejecutar la consulta y a desplegar el resultado de la
    búsqueda.

    // Se abre la base de datos

    <% db.open();

    // Se ejecuta la consulta

    resConsultaAlumno.Open(strSQL,db,adOpenStatic,adCmdText);

    // Se valida si se encontró algún registro según los criterios de nuestra consulta de
    búsqueda. Cuando no se encuentran registros, el cursor del Recordset se pone
    automáticamente al final de este, es decir, en Eof.

    if(resConsultaAlumno.Eof){%>
            La búsqueda no ha generado ningún resultado.
    <%
    }else{ %>

    // Si se encontraron registros que cumplen con la consulta de búsqueda, se procede
    a desplegarlos en una tabla.

            <table width="80%" border="1">
                    <tr>
                            <td> Nombre</td>
                            <td> Apellidos</td>
                            <td> Tipo de curso </td>
                            <td> Editar</td>
                            <td> Borrar</td>
```

```
</tr><%

// Se barre el Recordset para rellenar la tabla de resultados.
while(!resConsultaAlumno.Eof){%>
  <tr>
        <td><%=resConsultaAlumno("nombre")%></td>
        <td><%=resConsultaAlumno("apellidos")%></td>
        <td><%=resConsultaAlumno("tipoCurso")%></td>
    // Se pone una liga para editar al Alumno
        <td> <a
    href="Edicion.asp?noRegistro=<%=resConsultaAlumno("no
    Registro")%>"><b><font face="Verdana, Arial, Helvetica,
    sans-serif" color="#006600"
    size="2">Editar</font></b></a>
        </td>
    // Se pone una liga para borrar al Alumno
        <td>
            <a
            href="../borrado/Borrar.asp?noRegistro=<%=resCon
            sultaAlumno("noRegistro")%>"><b><font
            face="Verdana, Arial, Helvetica, sans-serif"
            color="#FF0000"
            size="2">Borrar</font></b></a></div>
        </td>
  </tr>
// Se pasa al siguiente elemento (Registro) del Recordset
<%
        resConsultaAlumno.MoveNext();
}//cierra while

// Se cierra el Recordset y la Base de Datos, recordemos que no debemos olvidar
cerrar los elementos u objetos que hayamos creado.

resConsultaAlumno.close()
db.close();%>
</table>
<%}%>
// Finaliza el proceso de despliegue de los registros que cumplen con la consulta de
búsqueda.
```

En la ***imagen 6.5.8-1*** podemos observar como queda el diseño de la tabla donde se despliegan los registros que cumplen con la consulta de búsqueda.

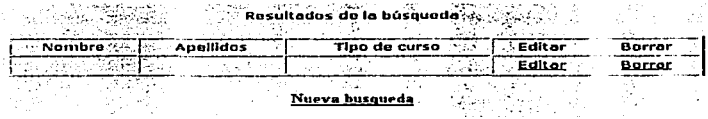

Imagen 6.5.8-1

Ya que se han obtenido los registros que cumplen con los criterios de búsqueda se pueden realizar dos cosas con dichos registros: Borrarlos o Actualizarlos. A continuación en los siguientes subtemas veremos como hacer estas dos operaciones.

6.5.9 Edición de Aspirantes en el archivo Edicion.asp

La página **"Edicion.asp"** es similar a la forma de captura solo que en ésta se despliegan los datos del alumno que ha sido seleccionado de la página **"Consulta.asp"** cuando se desea ***Actualizar*** los datos de dicho alumno.

La parte interesante de este archivo es la obtención de los valores del alumno seleccionado y después el llenado de los componentes de la forma con los datos obtenidos del alumno para su edición.

Una vez que se han editado los datos del alumno en la forma, se envían al archivo **"Actualiza.asp"** para su actualización en la base de datos. Como podemos observar este procedimiento es casi idéntico al que siguieron las páginas **"Captura.asp"** y **"Registra.asp"** de hecho se retomó el código de éstos y sólo se modificaron algunas cosas que a continuación se discutirán.

Primeramente en este subtema veremos la parte del código para el llenado de la forma con los datos del alumno elegido para editarlos y en el siguiente subtema discutiremos su actualización en la base de datos, sin olvidar que los códigos completos de estos archivos se encuentran en el ***Apéndice D que está en el disco anexo a éste trabajo.***

El Código para el llenado de la forma es:

```
//Se crean objetos para almacenar la información del Alumno

 resRefServicio = new ActiveXObject("ADODB.RecordSet");
 resContacto = new ActiveXObject("ADODB.RecordSet");
 resEscuela = new ActiveXObject("ADODB.RecordSet");
 resCiudadEstancia = new ActiveXObject("ADODB.RecordSet");
 resInfoAlumno = new ActiveXObject("ADODB.RecordSet");
 resInfoCurso = new ActiveXObject("ADODB.RecordSet");

//Querys para obtener cierta información del Alumno.

 strQryRefServicio = "SELECT * FROM ReferenciaDelServicio";
 strQryContacto = "SELECT * FROM Contacto";
```

```
  strQryEscuela = "SELECT * FROM Escuela";
  strQryCiudadEstancia = "SELECT idCiudadEstancia,deseCiudad,nombrePais from
CiudadEstancia, Pais where CiudadEstancia.idPais = Pais.idPais order by
nombrePais,descCiudad";

//Query para obtener los datos del Alumno
strQryInfoAlumno = "SELECT * FROM Alumno WHERE noRegistro =" +
Request("noRegistro");

//Query para obtener los datos del Curso del Alumno
  strQryInfoCurso = "SELECT noRegistro,convert(datetime,fechaActualizacion,113) as
fechaActualizacion,tipoCurso,convert(datetime,fechaInicioProg,113) as
fechaInicioProg,convert(datetime,fechaFinProg,113) as
fechaFinProg,costoPrograma,notasPrograma,convert(datetime,fechaLlegada,113) as
fechaLlegada,convert(datetime,fechaSalida,113) as fechaSalida FROM Curso WHERE
noRegistro = " + Request("noRegistro");

//Ejecución de los Querys

  resRefServicio.Open(strQryRefServicio,db,adOpenStatic,adCmdText);
  resContacto.Open(strQryContacto,db,adOpenStatic,adCmdText);
  resEscuela.Open(strQryEscuela,db,adOpenStatic,adCmdText);
  resCiudadEstancia.Open(strQryCiudadEstancia,db,adOpenStatic,adCmdText);
  resInfoAlumno.Open(strQryInfoAlumno,db,adOpenStatic,adCmdText);
  resInfoCurso.Open(strQryInfoCurso,db,adOpenStatic,adCmdText);

//Asignación de los datos del Alumno a Variables para desplegarlos posteriormente
en los componentes de la forma de edición.

  var rNoRegistro = resInfoAlumno("noRegistro");
  var rIdContacto=resInfoAlumno("idContacto");
  var rIdReferencia=resInfoAlumno("idReferencia");
  var rIdCiudadEstancia=resInfoAlumno("idCiudadEstancia");
  var rIdEscuela=resInfoAlumno("idEscuela");
  var rNombre=resInfoAlumno("nombre");
  var rApellidos=resInfoAlumno("apellidos");

//Así sucesivamente con los demás datos del alumno

//Asignación de los datos del Curso del Alumno a Variables para desplegarlos
posteriormente en los componentes de la forma de edición.

  var rFechaActualizacion=resInfoCurso("fechaActualizacion");
  var rTipoCurso=resInfoCurso("tipoCurso");
  var rFechaInicioProg=resInfoCurso("fechaInicioProg");
  var rFechaFinProg=resInfoCurso("fechaFinProg");
  var rCostoPrograma=resInfoCurso("costoPrograma");
  var rNotasPrograma=resInfoCurso("notasPrograma");
  var rFechaLlegada=resInfoCurso("fechaLlegada");
  var rFechaSalida=resInfoCurso("fechaSalida");

// Ahora se procede a llenar los componentes de la forma con los datos obtenidos

Nombre: <input type="text" name="nombre" size="30" value="<%= rNombre %>" >

Apellidos: <input type="text" name="apellidos"  size="30" value="<%= rApellidos
%>">
```

// Y así sucesivamente con los demás componentes de la forma.

// Para rellenar los menús desplegables se tiene que seguir un determinado procedimiento para determinar que elemento del menú es el que está elegido o mejor dicho cuál es el elemento que el Alumno eligió. A continuación un ejemplo con el menú de contactos, los demás menús trabajan bajo la misma ideología.

```
Contacto:
<select name="idContacto">
<%
while(!resContacto.Eof){
        var numContacto = resContacto(0);
        if(numContacto+"," == rIdContacto+","){ %>
                <option value="<%= numContacto %>"
                SELECTED><%=resContacto(1) %></option>
        <%}else{ %>
                <option value="<%= numContacto %>"><%=resContacto(1)
                %></option>
        <% }
        resContacto.MoveNext();
}%>
</select>
```


// De igual manera para rellenar los radio buttons, se tiene que seguir un procedimiento especial, para saber cuál de todas las opciones es la que esta seleccionada. A continuación un ejemplo con el tipo de curso, los demás radio buttons trabajan con la misma ideología.

```
<input type="radio" name="tipoCurso"
value="veranoAdultos"<%if(rTipoCurso+""=="VERANOADULTOS")Response.write(
" CHECKED");%>> Verano Adultos
```

// Una vez que se han rellenado los componentes se procede a enviar los datos al archivo "Actualiza.asp" para la actualización de los datos del alumno.
Pero antes de esto, al igual que el archivo "Captura.asp" el archivo "Edicion.asp" hace la validación de los datos, de hecho es la misma función de validación que usa el archivo "Captura.asp", esta función se encuentra al principio de nuestro archivo "Edicion asp", para una mayor referencia, consultar el Apéndice C, dónde se encuentra el código completo del archivo "Edicion.asp".

6.5.10 Actualización de Aspirantes en el archivo Actualiza.asp

Con los datos que le han sido enviados del archivo **"Edicion.asp"** el archivo **"Actualiza.asp"** se encarga de hacer la **actulización** de los datos del alumno, este archivo es muy similar al archivo **"Registra.asp"**, de hecho prácticamente lo único que cambia es la consulta que se hace a la base de datos, que en vez de ser un ***INSERT*** ahora es un ***UPDATE***, pero en cuanto a la formación de las consultas del Alumno y del Curso, son exactamente iguales.

A continuación se presentan algunas de las líneas de código que cambian, ya que algunas líneas sólo cambian en

cuanto al nombre de las variables, por ejemplo, en el archivo **"Registra.asp"** la cadena que contenía la consulta para insertar a un alumno se llamaba **"strSQLAlumno"** y en éste archivo de **"Actualiza.asp"**, esta cadena se llama **"strSQLUpdateAlumno"**, algunos de estos detalles como el anterior son los únicos cambios que se presentan a lo largo de éste archivo. Recordemos que el código completo de este archivo se encuentra dentro del ***Apéndice D que está en disco anexo a éste trabajo***:

```
// Así comienza la formación de la consulta de actualización del alumno.

strSQLUpdateAlumno = "UPDATE ALUMNO SET ";

//Así comienza la formación del query de actulización del curso.

strSQLUpdateCurso = "UPDATE CURSO SET ";

// Una vez que se han formado las consultas, la cadena que tiene la consulta,
termina de la siguiente manera. Recordemos que queremos Actualizar al alumno
que tiene cierto ID (identificador ó número de registro).

strSQLUpdateCurso = strSQLUpdateCurso + " WHERE
noRegistro="+Request.Form.Item("noRegistro");

// Finalmente, lo único que falta hacer es ejecutar las consultas para actualizar los
datos del alumno.
db.open();

cmdActualizaAlumno.ActiveConnection = db;
cmdActualizaAlumno.CommandType = adCmdText;
cmdActualizaAlumno.CommandText = strSQLUpdateAlumno.toUpperCase();
cmdActualizaAlumno.Execute();

cmdActualizaCurso.ActiveConnection = db;
cmdActualizaCurso.CommandType = adCmdText;
cmdActualizaCurso.CommandText = strSQLUpdateCurso.toUpperCase();
cmdActualizaCurso.Execute();
db.close();
```


6.5.11 Borrado de Aspirantes en el Archivo Borrar.asp

Después de que se eligió a un alumno para borrarlo en la página **"Consulta.asp"**, los datos de éste son enviados a la página **"Borrar.asp"** para visualizarlos y estar seguros que queremos borrar a ese alumno. El proceso de obtención de los datos y su despliegue en la página es idéntico al que se utilizó en la página **"Edicion.asp"**. Si ya estamos seguros de que queremos borrar a este alumno, se procede a enviar el ID ó número de registro de ese alumno al archivo **"Eliminar.asp"** para borrarlo de la base de datos.

En la ***imagen 6.5.11-1*** podemos observar como quedó el diseño de la página **"Borrar.asp"**.

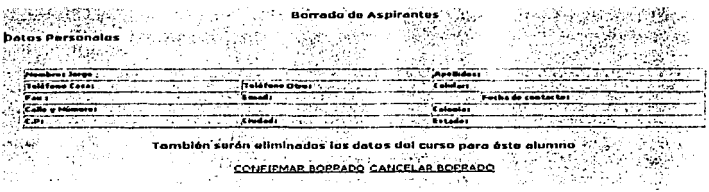

Imagen 6.5.11-1

A continuación se presenta parte del código del archivo **"Borrar.asp"**, recordando que el código completo se encuentra en el ***Apéndice D que está en el disco anexo a éste trabajo***:

```
<%
//Se crea un objeto para obtener los datos del alumno.

resInfoAlumno = new ActiveXObject("ADODB.RecordSet");

// Se crea una cadena con la consulta a la base de datos con la info del Alumno.
strQryInfoAlumno = "SELECT
nombre,apellidos,telefonoCasa,telefonoOtro,celular,fax,email,calleYNumero,colonia,co
digoPostal,ciudadAlumno,estado,fechaContacto FROM Alumno WHERE noRegistro
=" + Request("noRegistro");

// Se ejecuta la consulta.
resInfoAlumno.Open(strQryInfoAlumno,db,adOpenStatic,adCmdText);

//Asignación de los datos del alumno a variables para desplegarlos posteriormente.
  var rNombre=resInfoAlumno("nombre");
  var rApellidos=resInfoAlumno("apellidos");

// Y así sucesivamente se siguen asignando los datos del alumno a variables.
%>

// Finalmente se despliegan los datos del alumno.

Nombre: <%= rNombre %>
Apellidos: <%= rApellidos %>

// Si se decide borrar al alumno, se envía el número de registro del alumno al
archivo "Eliminar.asp" que es el que se encarga de realizar el borrado del alumno
```

de la base de datos. Observermos que el número de registro se envía en el URL como parámetro.

```
<a ref. = "Elimina.asp?noRegistro = <%Response.write(Request("noRegistro"))%>">
CONFIRMAR BORRADO</a>
```

6.5.12 Eliminación de Aspirantes en el Archivo Eliminar.asp

Dentro de este archivo es donde llevamos a cabo el borrado del alumno y del curso, este archivo es muy sencillo lo único que tenemos que hacer es formar una cadena para borrar los registros que contengan el número de registro que ha sido enviado desde el archivo **"Borrar.asp"**. A continuación veremos el proceso de borrado. Para una mayor referencia consultar el código completo en el ***Apéndice D que está en el disco anexo a éste trabajo***:

```
// Creamos un objeto para realizar el borrado del alumno y del curso
 cmdBorra = new ActiveXObject("ADODB.Command");

// Asignamos los querys de borrado a dos variables.

 strSQLBorraAlumno = "DELETE FROM Alumno WHERE noRegistro
="+Request("noRegistro");

 strSQLBorraCurso = "DELETE FROM Curso WHERE noRegistro
="+Request("noRegistro");

 db.open();

// Hacemos las asignaciones necesarias para proceder al borrado del alumno

 cmdBorra.ActiveConnection = db;
 cmdBorra.CommandType = adCmdText;
 cmdBorra.CommandText = strSQLBorraCurso;

// Eliminamos al curso en primer instancia ya que como depende del alumno
primero hay que borrar al curso.

 cmdBorra.Execute();

// Reasignamos el objeto para que ahora borre  al alumno

 cmdBorra.CommandText = strSQLBorraAlumno;

// Borramos al alumno

 cmdBorra.Execute();

// Cerramos la conexión a la base de datos

db.close();
%>
```

A lo largo de éste capitulo hemos visto cómo con la ayuda de Javascript y de las Active Server Pages (ASP's), podemos gestionar una base de datos en Internet, mostrando como se hacen las tareas más comunes en una base de datos, como son: La inserción, la actualización, la búsqueda y el borrado de registros. Como pudimos darnos cuenta no es muy complicado gestionar una base de datos en internet, cuando se tienen bien entendidos los conceptos de cómo gestionar una base de datos y de cómo trabajan las Active Server Pages y todos sus componentes.

La documentación que existe hoy en día sobre las bases de datos y sobre las Active Server Pages es muy extensa. Por lo que es imposible comentar en un solo trabajo todos los detalles y componentes de estas tecnologías. Por eso mismo en el presente trabajo era imposible comentar cada detalle que aparecía, por lo que sólo se comentó lo más relevante, significativo y sobre todo lo más productivo.

Conclusiones

JavaScript es un lenguaje de programación creado con el objetivo de integrarse en HTML y facilitar la creación de páginas interactivas sin necesidad de utilizar scripts de CGI.

El código de programa de JavaScript, llamado script, se introduce directamente en el documento HTML y no necesita ser compilado, es el propio navegador el que se encarga de traducir dicho código.

Gracias a JavaScript podemos desarrollar programas que se ejecuten directamente en el navegador (cliente) de manera que éste pueda efectuar determinadas operaciones o tomar decisiones sin necesidad de acceder al servidor. Por ejemplo, al desarrollar un programa que verifique una clave de acceso para poder acceder a una determinada página web, JavaScript debe comprobar la información dada por el usuario, verificar que sea correcta y actuar en consecuencia.

Una de las razones por las que vale la pena utilizar ASP y JavaScript es que facilitan la creación rápida de scripts.

Una de las mejores cualidades de las ASP's es la posibilidad de realizar accesos a bases de datos desde internet. A través de los objetos ADO podemos gestionar una base de datos de forma absoluta; desde sencillas selecciones hasta la creación de tablas.

La gran ventaja que nos ofrecen estos objetos es la posibilidad de utilizar SQL directamente sobre la base de datos. De esta forma no sólo tenemos las opciones que nos proporcionen los objetos ADO, sino toda la potencia de SQL.

Hemos visto que con los objetos ADO, podemos realizar las operaciones básicas sobre una base de datos: recorrido de una tabla, inserción, búsqueda, modificación y borrado de registros.

La tecnología ASP ha sido diseñada por Microsoft® para facilitar la creación de sitios web con una sencillez mayor que la empleada en la programación CGI.

En ASP todas las páginas web pueden ser diseñadas con editores de HTML, puesto que las instrucciones ejecutables y el código HTML están suficientemente delimitados. Así mismo, pueden utilizarse diversos lenguajes para la programación de la funcionalidad de nuestras páginas activas.

ASP permite compatibilizar la creación de páginas web activas en el cliente y en el servidor, pudiéndose así balancear la carga de proceso y de comunicaciones según los deseos del diseñador.

Hemos podido comprobar que con la conjunción de estas dos tecnologías podemos gestionar una base de datos de manera optima por una parte validando los datos que son enviados al servidor y por otro la gestión de nuestra base de datos.

A lo largo de éste trabajo pudimos darnos cuenta que la utilización de ASP's y JavaScript para gestionar una Base de Datos permite que la información pueda ser actualizada y modificada de una manera muy sencilla. Además de que es muy útil usar estas tecnologías cuando se desean publicar datos que varían temporalmente o requieren una actualización constante: listas de precios, catálogos, etc. Por lo que el costo de la actualización y mantenimiento de nuestra información es muy bajo y se logra un enorme beneficio al tener una información con una consistencia casi perfecta por llamarlo de alguna forma.

Las ASP's permiten construir sitios donde el usuario puede interactuar con el sitio, solicitando por ejemplo, listados parametrizados y se pueden actualizar en tiempo real los datos almacenados en la base de datos. Por ejemplo, un vendedor a través de una computadora conectada a Internet, puede desde la casa del cliente, cerrar una operación comercial, emitir la orden de pedido y descontar del stock la mercadería vendida. Todo esto con un bajo costo y enorme beneficio para la empresa.

Apéndice A
Instalación y configuración del Internet Information Server® y del Personal Web Server®

Instalación del Personal Web Server®

El Personal Web Server®, más conocido por sus siglas ***PWS***, es un servidor web personal que nos permite montar un servidor web en nuestra PC. De esta manera, podemos correr nuestras aplicaciones web directamente sin necesidad de subir el sitio a un servidor en línea (online).

El ***PWS*** se utiliza principalmente para correr las aplicaciones programadas en lenguaje ASP (Active Server Pages®), pero también es posible correr Perl® o Php® instalando y configurando los interpretes de cada uno de ellos.

El Personal Web Server® se encuentra en el CD de Instalación de Windows 98®, y si no se tiene se puede bajar desde el sitio de Microsoft®. Para proceder a la instalación nos dirigimos hacia el menú ***Inicio / Ejecutar*** y escribimos la ruta hacia el archivo de instalación, por ejemplo ***d:\add-ons\pws\instalar.exe*** (d: o la unidad de CD). Si por algún motivo aparece un error del Winsock, hay que instalar la versión 2 (Ws2setup.exe), disponible en el mismo directorio.

Aparecerá la clásica pantalla de bienvenida, en donde nos comenta en forma breve de que se trata el programa y las características que incluye. La ***imagen A-1*** nos muestra como se ve ésta pantalla.

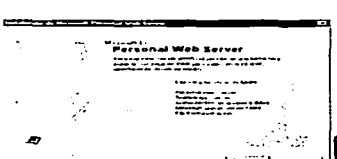

Imagen A-1

Pulsamos el botón ***Siguiente*** para continuar.

A continuación hay que seleccionar los componentes a instalar, se puede dejar las opciones predeterminadas (*instalación típica*) o escoger los componentes que queramos (*instalación personalizada*). Una vez finalizada la instalación, habrá que reiniciar el sistema.

Al entrar nuevamente, notaremos un nuevo icono en la barra de programas que indica que el Personal Web Server está funcionando. La ***imagen A-2*** nos muestra como se ve éste icono.

Imagen A-2

Administrador de Web Personal

Al hacer doble click sobre el icono del **PWS** en la barra de programas se abrirá una ventana con las características del programa. Si vamos a **Avanzada** podemos configurar algunos aspectos de nuestro servidor, como pueden ser los diferentes permisos en los directorios, los documentos predeterminados, permitir la exploración de directorios, entre otros. La ***imagen A-3*** nos muestra como se ve ésta pantalla.

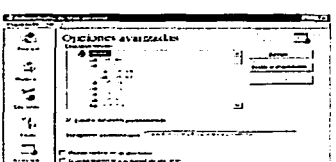

Imagen A-3

Comprobación de una correcta instalación

Una vez instalado el servidor, podemos abrir cualquier navegador e ingresar la dirección: **http://localhost** o **http://127.0.0.1** o también **http://nombre_de_la_maquina**. Las tres opciones funcionarán correctamente. Tendrá que aparecer la página de *"Bienvenidos a Microsoft Personal Web Server 4.0"*.

Ahora solo resta ubicar las páginas dentro de **c:\inetpub\wwwroot** (o en la ubicación que hayamos especificado durante la instalación).

El Personal Web Server 4.0 viene con soporte incorporado de las páginas ASP, por lo que no necesitamos realizar ninguna configuración adicional.

Instalación de IIS en Windows XP Profesional

Internet Information Server® (IIS) es el servidor de páginas web avanzado de la plataforma Windows®. Se distribuye gratuitamente junto con las versiones de Windows® basadas en NT®, como pueden ser Windows 2000 Profesional® o Windows 2000 Server®, así como Windows XP®.

Las normas de instalación que a continuación veremos son aplicables, a nivel general, a las que podemos encontrarnos en las distintas versiones de los sistemas operativos comentados antes.

En Windows 95®, Windows 98®, las versiones Home de Windows XP® y ME de Windows 2000®, no se admite la instalación de IIS®. En su lugar podemos instalar el Personal Web Server®, del que ya hemos visto como se instala.

IIS® se puede encontrar en el propio CD de instalación de Windows XP Profesional®. Lo primero que tenemos que hacer es acceder a la opción de ***"Instalar componentes opcionales de Windows"*** para poder cargarlo en nuestro sistema. Para ello tenemos dos opciones:

1) Insertar el CD de instalación de Windows y en la ventana de autoarranque que se muestra, seleccionar la opción que pone ***"Instalar componentes opcionales de Windows"***. La ***imagen A-4*** nos muestra como se ve esta pantalla.

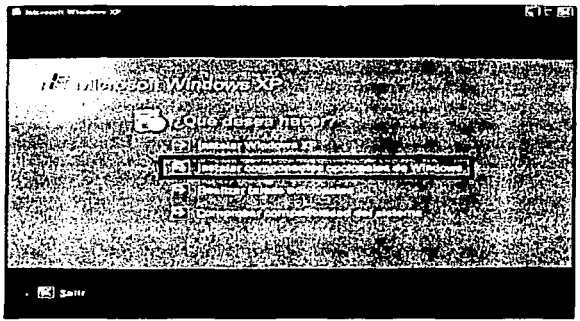

Imagen A-4

2) En el Panel de control, seleccionar la opción de ***"Agregar o quitar programas"*** y en la ventana que sale, pulsar sobre el icono de la izquierda marcado como ***"Seleccionar o quitar componentes de Windows"***. La ***imagen A-5*** nos muestra como se ve esta pantalla.

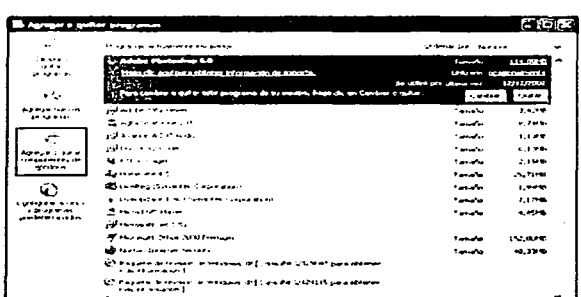

Imagen A-5

Ahora nos muestra la ventana para seleccionar los componentes adicionales de Windows que hay disponibles. En la lista, marcamos la opción ***"Servicios de Internet Information Server (IIS)"***. Por defecto se seleccionan unos cuantos componentes, dentro de los que ofrece la instalación de IIS. Nosotros podemos elegir qué componentes deseamos instalar apretando el botón marcado como ***"Detalles"***. Entre los componentes posibles se encuentran las extensiones de Frontpage, documentación, servicios adicionales de IIS, un servidor de FTP (para la transferencia de ficheros con el servidor por FTP), incluso uno de SMTP (para el envío de correos electrónicos).

Si no sabemos qué componentes instalar podemos dejar las opciones como aparecen en un principio, pues para la mayoría de los casos serán válidas. Sólo un detalle: puede ser adecuado no instalar las extensiones de Frontpage en caso de que no pensemos que se vayan a utilizar. La ***imagen A-6*** nos muestra como se ve esta pantalla.

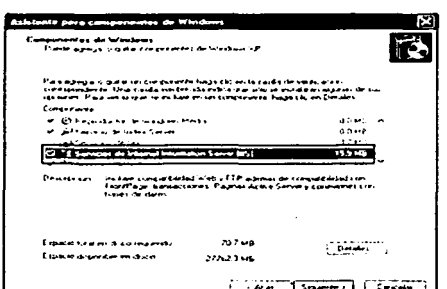

Imagen A-6

Una vez hemos instalado los componentes deseados, presionamos el botón de ***"Siguiente"*** para comenzar la instalación.

Acceder al servidor web

Podemos acceder al servidor web para comprobar si se ha instalado correctamente IIS®. Para ello simplemente debemos escribir **http://localhost** en Internet Explorer y debería aparecer una página web informando que IIS está correctamente instalado. Además, aparecerá la documentación de IIS en una ventana emergente, si es que fue instalada. La ***imagen A-7*** nos muestra como se ve esta pantalla.

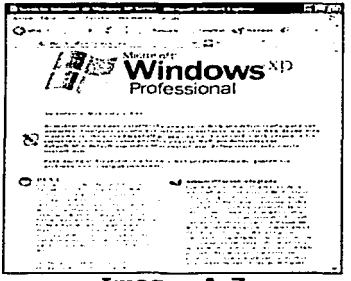

Imagen A-7

Igualmente, ahora solo resta ubicar las páginas dentro de **c:\inetpub\wwwroot** (o en la ubicación que hayamos especificado durante la instalación).

Apéndice B
Instalación y configuración de WS FTP®

Instalación

La instalación de WS FTP® es muy sencilla con pocos pasos a seguir, sólo se tienen que confirmar las opciones y responder a las preguntas que el asistente de instalación hace. Podemos obtener una copia de este programa desde internet descargándolo de la página http://www.download.com. Para iniciar la instalación, se debe ejecutar el archivo ***wsftp.exe***, algunas versiones cambian el nombre de este archivo por ejemplo, el WS FTP LE® se llama ***wsftple.exe***, en nuestro caso instalaremos la versión profesional de este programa, que se llama WS FTP Pro, cuyo archivo de instalación se llama ***wsftppro-cnet.exe,*** el cual debemos ejecutar para comenzar con la instalación.

En la primera pantalla, basta pulsar el botón "**Next**". Aparecerá entonces una pregunta (ver ***imagen B-1***). Responder "**Yes**" y continuará la instalación.

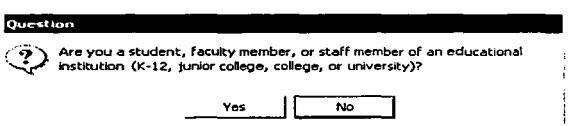

Imagen B-1

A continuación, aparecerá una pantalla con las características del programa, en esta pantalla se tiene que pulsar "**Yes**".

En la siguiente pantalla se podrá escoger la carpeta en la que se desea instalar el programa. El sistema mostrará la carpeta por defecto. Si se desea instalar el programa en una carpeta diferente, se tendrá que seleccionar el botón "**Browse...**" donde se abrirá una nueva ventana, preguntando dónde se desea instalar el programa. La ***imagen B-2*** nos muestra como se ve esta pantalla.

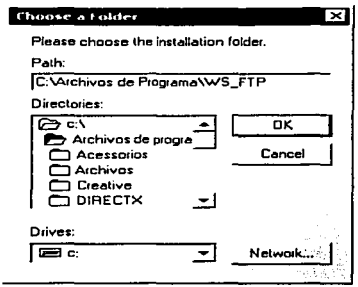

Imagen B-2

Una buena idea es instalar el programa en **"C:\Archivos de Programa\WS_FTP"**. Después de seleccionar el directorio, se tiene que pulsar el botón **"OK"**.

Posteriormente se quedará en la ventana el nombre de la carpeta que se eligió para instalar el programa, ahí tenemos que pulsar el botón **"Next"**.

Una nueva ventana se representará preguntando el nombre del grupo de programas donde colocar los íconos del WS FTP® finalmente se pulsa el botón **"Next"** y comenzará la instalación. Después de copiados e instalados todos los archivos, aparecerá otra ventana preguntando si se desean leer las notas sobre el programa y abrirlo inmediatamente. Si no se desean leer las notas ni abrir el programa en este momento, simplemente se tiene que pulsar el botón **"Finish"** y se cerrará el programa de instalación.

Configuración

Después que hemos instalado el WS FTP®, se tienen que seguir los siguientes pasos para configurar el programa y poder enviar archivos con toda facilidad y rapidez.

Para comenzar tenemos que ejecutar el programa. Luego de la pantalla de inicio, aparecerá una ventana solicitando que se ingrese el email del usuario, posteriormente se tiene que pulsar "**OK**".

Inmediatamente aparecerá una ventana en donde se lleva a cabo la configuración, la ***imagen B-3*** nos muestra como se ve esta pantalla.

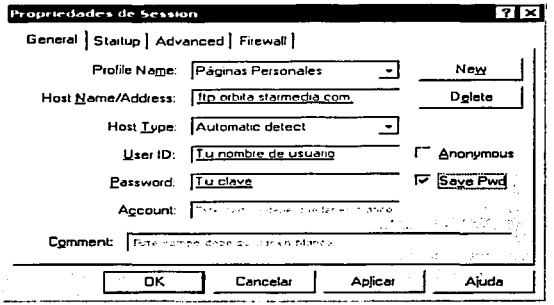

Imagen B-3

A continuación se presenta una breve descripción de cada parte de esta ventana de configuración:

- En "**Profile Name:**", se pone una breve descripción de nuestra configuración, por ejemplo, "**Páginas Personales**".
- En el campo "**Host Name/Address**", se escribe la dirección del servidor FTP , por ejemplo, "**ftp.orbita,starmedia.com**".
- En "**Host Type**" se selecciona por default la opción "**Automatic detect**".
- En el campo "**User ID:**" se coloca el Nombre de usuario (siempre en minúsculas), y en "**Password:**", tu clave de acceso.
- La opción "**Save Pwd**" hace que el programa guarde la clave de acceso y no se tenga que escribir cada vez que se conecte al servidor FTP.

Luego de llenar esos campos, basta pulsar "**OK**" y el programa iniciará la conexión con el servidor para poder realizar la transferencia de archivos.

Después de conectase, aparecerá una ventana como la de la ***imagen B-4***:

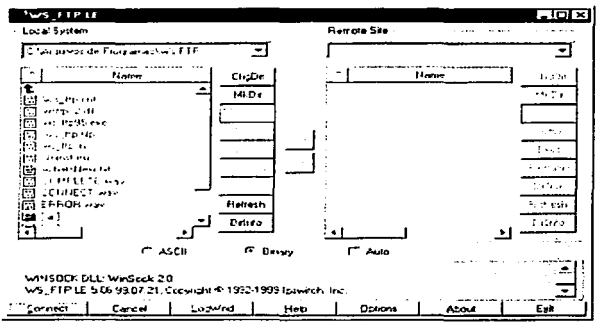

Imagen B-4

Cuando aparezca esta pantalla, se podrá observar que la parte izquierda de ella es "**la computadora local**", y la derecha es "**el servidor FTP**". Ahora se tiene que ir al directorio donde se encuentran los archivos y/o subdirectorios que se desean transferir, seleccionarlos en la parte izquierda (para seleccionar múltiples archivos basta con mantener oprimida la tecla "**Shift**" del teclado mientras haces la selección) y pulsar el botón con la flecha que señala hacia la derecha "=>", para que los respectivos archivos sean transferidos al servidor FTP. Para transferir archivos del servidor a la computadora local sólo se tiene que seleccionar los archivos en el lado derecho y pulsar el botón con la flecha que señala hacia la izquierda "<=". Con esto ahora podremos enviar y recuperar los archivos del y hacia el servidor.

Apéndice C
Instalación de Microsoft® SQL SERVER® 7.0

Microsoft® SQL Server® es un sistema gestor de bases de datos relacionales (SGBDR) o de sus siglas en inglés RDBMS (Relational Data Base Management System). Microsoft® SQL Server® 7.0 es el sucesor de Microsoft® SQL Server® 6.5 y las mejoras introducidas en esta nueva versión según sus desarrolladores, han disparado sus posibilidades en el mercado empresarial.

Debido a la gran cantidad de datos que los sistemas de información deben almacenar hoy en día, una de las características más importantes en este tipo de aplicaciones es su escalabilidad. SQL Server® 7.0 soporta bases de datos de hasta 1.048.516 TB (terabytes), lo que supone una mejora notable con respecto a la versión 6.5 que era de 1 TB de capacidad para una base de datos. Estas bases de datos de gran tamaño son conocidas como VLDB, Very Large DataBases.

Otra de las mejoras de SQL Server® 7.0 es la asignación dinámica de recursos, ya que asigna la memoria de forma dinámica desde el propio sistema operativo en respuesta a las demandas del sistema y continúa asignando memoria hasta utilizar toda la que quede disponible. A medida que las demandas de una base de datos disminuyen, el sistema va devolviendo al sistema operativo la memoria no utilizada.

A diferencia de lo que ocurría con SQL Server® 6.5, las bases de datos de SQL Server® 7.0 ya no residen en dispositivos de tamaño fijo, sino en archivos estándares del sistema operativo, facilitando que las bases de datos puedan crecer de forma dinámica a medida que las tablas de la base de datos aumentan de tamaño, eliminando la necesidad de una intervención por parte del administrador.

Todas estas características y más, le convierten en un candidato ideal para el desarrollo de aplicaciones que requieran acceder y manejar un gran volumen de información.

Algunas de la imágenes del ***apéndice C*** fueron extraídas del manual de Instalación realizado por Vicente Javier Fdez. Godoy de la oficina de Servicios de Informática de la UC3M.

INSTALACIÓN

Para llevar a cabo la instalación de SQL Server® 7.0, introducimos el CD etiquetado como **Microsoft SQL Server 7.0.**

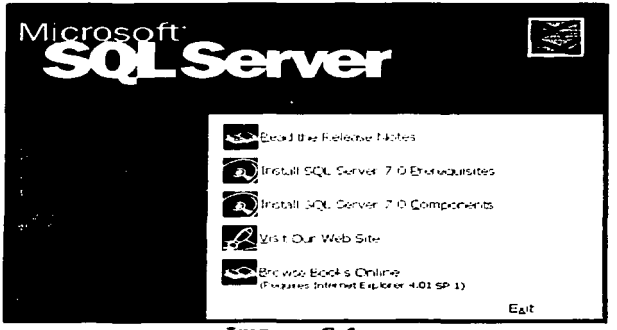

Imagen C-1

Inmediatamente se inicia la instalación y aparece una pantalla como la de la ***imagen C-1***. Seleccionamos **Install SQL Server 7.0 Components.**

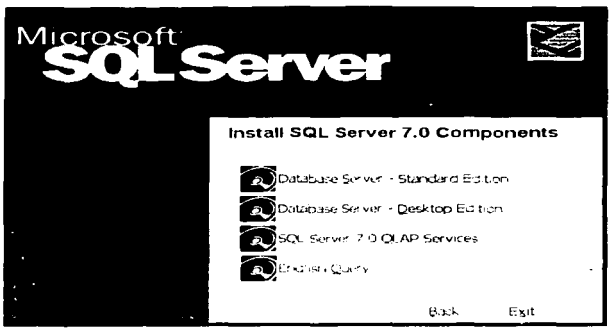

Imagen C-2

Posteriormente aparece una pantalla como la de la ***imagen C-2***, de la cual seleccionamos **Database Server – Standard Edition.**

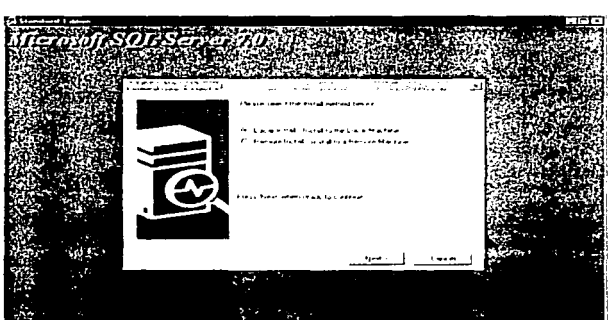

Imagen C-3

De la ***Imagen C-3*** seleccionamos **Local Install – Install to the local Machine** y hacemos click en **Next.**

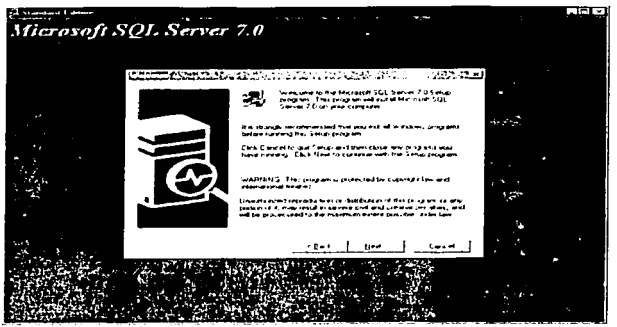

Imagen C-4

De la ***Imagen C-4*** pulsamos nuevamente **Next** para proseguir con la instalación.

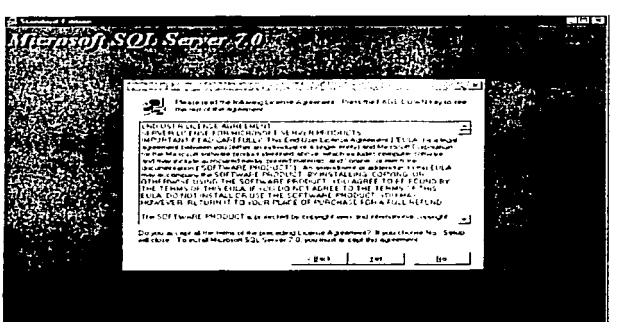

Imagen C-5

De la ***Imagen C-5*** aceptamos los términos del contrato pulsando **Yes**.

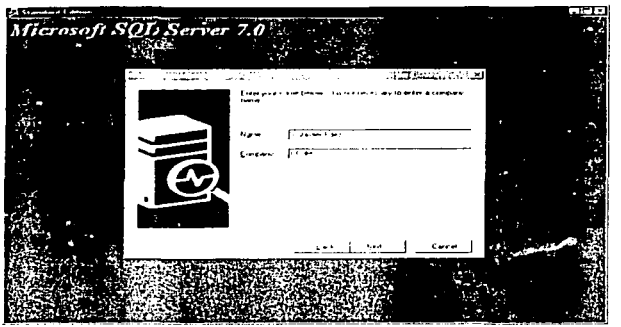

Imagen C-6

En la ***imagen C-6*** introducimos nuestro nombre de usuario y compañía y aceptamos pulsando **Next.**

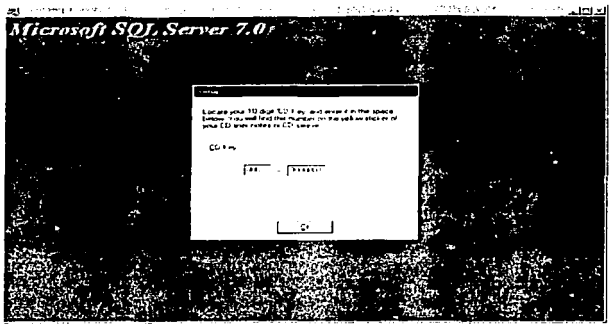

Imagen C-7

Para continuar con la instalación tecleamos la clave (CD Key) que viene impresa en el CD y aceptamos pulsando **Ok**, en la pantalla similar a la ***Imagen C-7***.

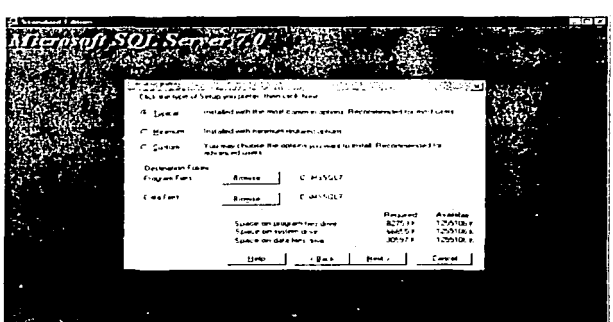

Imagen C-8

De la pantalla similar a la ***imagen C-8*** seleccionamos el tipo de instalación que queremos realizar, en nuestro caso **Typical** y hacemos click en **Next.**

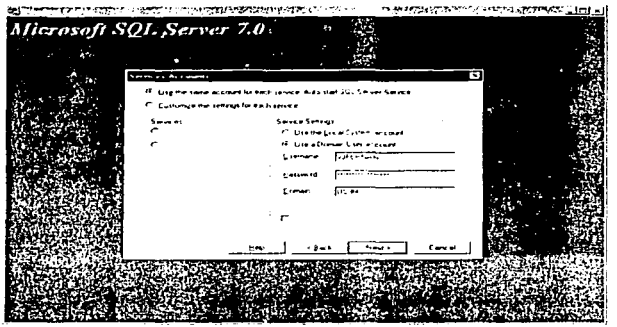

Imagen C-9

De la pantalla similar a la ***imagen C-9*** establecemos el tipo de cuenta que queremos usar para validarnos frente a la aplicación. Pulsamos **Next** para continuar.

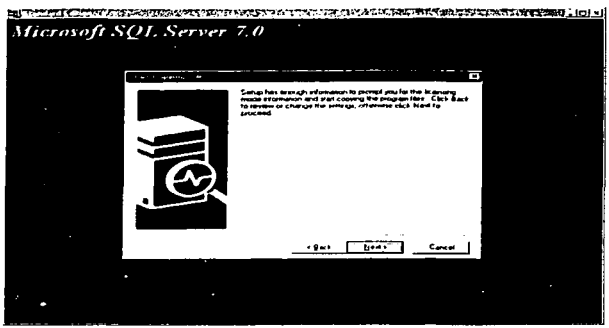

Imagen C-10

De la pantalla similar a la ***imagen C-10*** pulsamos **Next.**

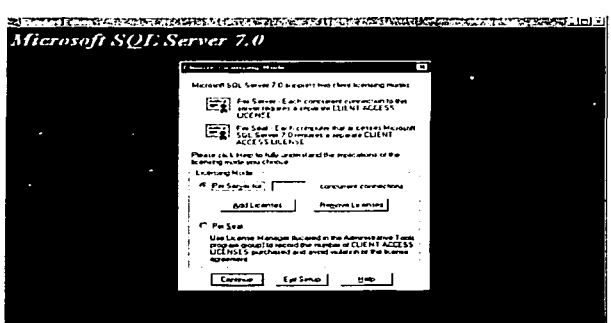

Imagen C-11

En la pantalla similar a la ***imagen C-11*** indicamos el tipo de licencia que vamos a usar. Seleccionamos **Per Server for** y pulsamos **Add Licenses** para añadir el número de licencias a usar.

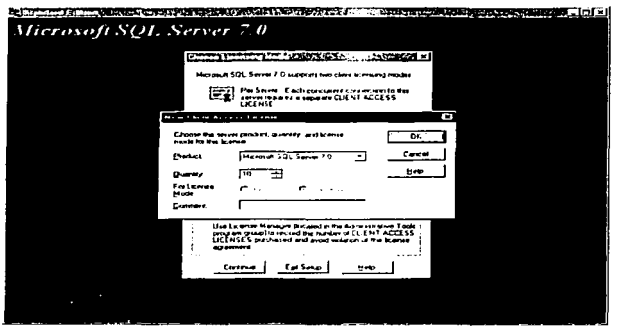

Imagen C-12

En la pantalla similar a la ***imagen C-12*** estableceremos un número de 10 y aceptaremos pulsando **Ok.** Continuamos la instalación pulsando **Continue.**

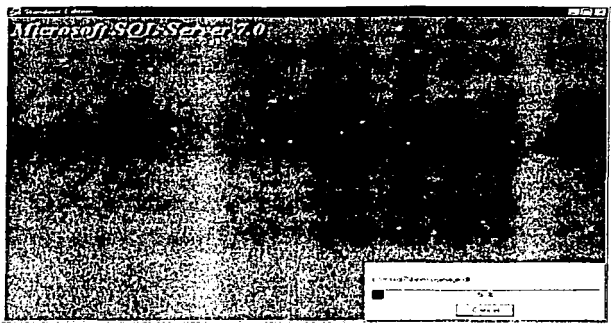

Imagen C-13

En la ***imagen C-13*** podemos observar como se inicia la copia de archivos.

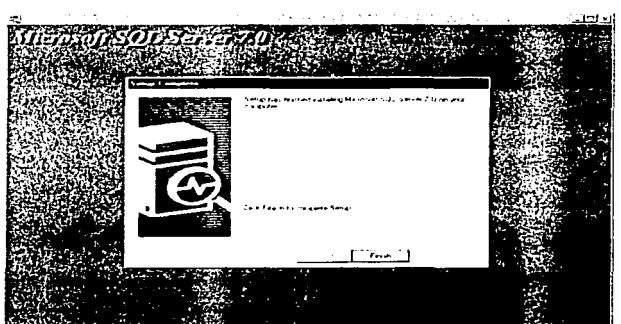

Imagen C-14

Como podemos observar en la ***imagen C-14*** pulsamos **Finish** para concluir con la instalación de SQL Server® 7.0.

ARRANCANDO EL SERVIDOR SQL SERVER® 7.0

Tras realizar la instalación podemos poner en funcionamiento el servidor SQL Server®. Para ello accedemos a **Inicio – Programas – Microsoft® SQL Server® 7.0 – Service Manager.** En la ***Imagen C-15*** podemos observar esto último.

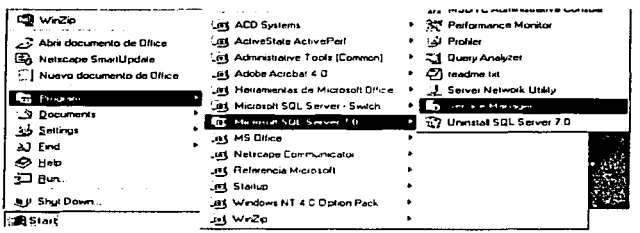

Imagen C-15

Finalmente ponemos en marcha el servicio haciendo click en el botón **Start/Continue,** como podemos observar en la ***imagen C-16.***

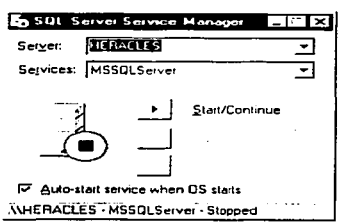

Imagen C-16

Si queremos que el servicio se arranque cuando se inicie el sistema operativo deberemos marcar el checkbox **Auto-start service when OS starts.** Si no ha habido ningún problema habremos finalizado la instalación y en la barra de inicio del sistema operativo nos debería aparecer un nuevo icono (una especie de CPU con una flechita verde), como el de la ***imagen C-17***.

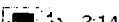

Imagen C-17

Glosario

@ (arroba) Este signo es uno de los componentes de las direcciones de correo electrónico y separa el nombre del usuario de los nombres de dominio del servidor de correo (ejemplo: rfcalvo@ati.es); el origen de su uso en Internet está en su frecuente empleo en inglés como abreviatura de la preposición at ("en", pero en un sentido espacial muy concreto y específico). Se usa también cada vez más frecuentemente en el lenguaje escrito políticamente correcto para evitar tener que repetir sustantivos y adjetivos según el género: así Estimad@s amig@s sustituye a Estimados amigos y estimadas amigas o a Estimadas/os amigas/os.

401 Unauthorized (401 No autorizado) 401 es un código de estado frecuente que indica a un usuario del Web que no está autorizado a acceder a una determinada página. 401 y otros códigos de estado forman parte del protocolo HTTP de WWW, escrito en 1992 por el inventor del Web, Tim Berners-Lee, que tomó muchos de dichos códigos de los correspondientes al FTP (File Transfer Protocol).

404 Not found (404 No encontrado) 404 es un código de estado frecuente que indica a un usuario del Web que no se ha encontrado (Not found) una determinada página. 404 y otros códigos de estado forman parte del protocolo HTTP de WWW, escrito en 1992 por el inventor del Web, Tim Berners-Lee, que tomó muchos de dichos códigos de los correspondientes al FTP (File Transfer Protocol).

A

Active Server Page -- ASP (Página de Servidor Activo) Una página ASP es un tipo especial de página HTML que contiene unos pequeños programas (también llamados scripts) que son ejecutados en servidores Microsoft Internet Information Server antes de ser enviados al usuario para su visualización en forma de página HTML. Habitualmente esos programas realizan consultas a bases de datos y los resultados de esas consultas determinan la información que se envía a cada usuario específico. Los ficheros de este tipo llevan el sufijo .asp. No confundir con Application Service Provider.

ActiveX (ActiveX) Lenguaje desarrollado por Microsoft para la elaboración de aplicaciones exportables a la red y capaces de operar sobre cualquier plataforma a través, normalmente, de navegadores WWW. Permite dar dinamismo a las páginas web.

address (dirección)) En Internet dícese de la serie de caracteres, numéricos o alfanuméricos, que identifican un determinado recurso de forma única y permiten acceder a él. En la red existen varios tipos de dirección de uso común: dirección de correo electrónico (e-mail address), IP (dirección internet) y dirección hardware o dirección MAC (hardware address o MAC address).

anonymous FTP (FTP anónimo) El FTP anónimo permite a un usuario de Internet la captura de documentos, ficheros, programas y otros datos contenidos en archivos existentes en numerosos servidores de información sin tener que proporcionar su nombre de usuario y una contraseña (password). Utilizando el nombre especial de usuario anonymous, o a veces ftp, el usuario de la red podrá superar los controles locales de seguridad y podrá acceder a ficheros accesibles al público situados en un sistema remoto.

Apache (Apache) Servidor HTTP de dominio público basado en el sistema operativo Unix. Apache fue desarrollado en 1995 y es actualmente uno de los servidores HTTP más utilizados en la red.

Applet (aplicacioncita,aplique ,applet) Pequeña aplicación escrita en Java y que se difunde a través de la red para ejecutarse en el navegador cliente.

B

Bookmark (marca de página,marca,marcapáginas) Señal o recordatorio que los internautas dejan en su aplicación de navegación para marcar un lugar interesante encontrado en la red Internet a fin de poder volver a él posteriormente.

Browser (visor, visualizador,hojeador,navegador) Aplicación para visualizar todo tipo de información y navegar por el espacio Internet. En su forma más básica son aplicaciones hipertexto que facilitan la navegación por los servidores de información Internet; cuentan con funcionalidades plenamente multimedia y permiten indistintamente la navegación por servidores WWW, FTP, Gopher, el acceso a grupos de noticias, la gestión del correo electrónico, etc.

C

Cascade Style Sheet -- CSS (Hoja de Estilo en Cascada) Es un conjunto de instrucciones HTML que definen la apariencia de uno o más elementos de un conjunto de páginas web con el objetivo de uniformizar su diseño.

cgi-bin (cgi-bin) Directorio de un servidor web donde suelen almacenarse los programas CGI. bin es una contracción de binario.

click (cliqueo/cliquear,clic,pinchazo/pinchar,pulsación/pulsar) Acción de tocar un mando cualquiera de un ratón una vez colocado el puntero del mismo sobre una determinada área de la pantalla con el fin de dar una orden al ordenador.

client (cliente) Un sistema o proceso que solicita a otro sistema o proceso que le preste un servicio. Una estación de trabajo que solicita el contenido de un fichero a un servidor de ficheros es un cliente de este servidor.

client-server model (modelo cliente-servidor) Modelo de comunicación entre ordenadores conectados a una red en el cual hay uno, llamado cliente, que satisface las peticiones realizadas por otro llamado servidor.

Common Gateway Interface -- CGI (Interfaz Común de Pasarela) Interfaz de intercambio de datos estándar en WWW a través del cual se organiza el envío y recepción de datos entre navegador y programas residentes en servidores WWW.

computer (computadora,computador,ordenador) Máquina electrónica capaz de procesar información siguiendo instrucciones almacenadas en programas. Antes que electrónicas estas máquinas fueron mecánicas o electromecánicas

cookie (espía,cookie,cuqui ,fisgón) Conjunto de caracteres que se almacenan en el disco duro o en la memoria temporal del ordenador de un usuario cuando accede a las páginas de determinados sitios web. Se utilizan para que el servidor accedido pueda conocer las preferencias del usuario al volver éste a conectarse. Dado que pueden ser un peligro para la intimidad de los usuarios, éstos deben saber que los navegadores permiten desactivar los (¿o las?) cuquis.

copyright (derecho de copia) Derecho que tiene un autor, incluido el autor de un programa informático, sobre todas y cada una de sus obras y que le permite decidir en qué condiciones han ser éstas reproducidas y distribuidas. Aunque este derecho es legalmente irrenunciable puede ser ejercido de forma tan restrictiva o tan generosa como el autor decida. El símbolo de este derecho es ©.

D

distributed database (base de datos distribuida) Conjunto de datos almacenados en diversos lugares que para un usuario aparecen como una base de datos única. Un ejemplo esencial en Internet es el Domain Name System.

Domain Name System -- DNS (Sistema de Nombres de Dominio) El DNS un servicio de búsqueda de datos de uso general, distribuido y multiplicado. Su utilidad principal es la búsqueda de direcciones IP de sistemas anfitriones (hosts) de Internet basándose en los nombres de éstos. El estilo de los nombres de host utilizado actualmente en Internet es llamado nombre de dominio. Los dominios originarios, a los que se ha añadieron algunos más en el año 2000, son: .com (comercial, empresas), .edu (educación, centros docentes), .org (organización sin ánimo de lucro), .net (operación de la red), gov (gobierno o administración pública) y .mil (ejercito de los EE.UU.). La mayoría de los países tienen un dominio propio. Por ejemplo, .mx (México), .es (España), .au (Australia).

Dynamic HTML (HTML dinámico) Extensiones del lenguaje HTML que permiten crear páginas web más animadas y expresivas.

E

eXtensible Markup Language -- XML (Lenguaje Extensible de Marcado) Lenguaje desarrollado por el W3 Consortium para permitir la descripción de información contenida en el WWW a través de estándares y formatos comunes, de manera que tanto los usuarios de Internet como programas específicos (agentes) puedan buscar, comparar y compartir información en la red. El formato de XML es muy parecido al del HTML aunque no es una extensión ni un componente de éste.

F

File Transfer Protocol -- FTP (Protocolo de Transferencia de Ficheros) Protocolo que permite a un usuario de un sistema acceder a, y transferir desde, otro sistema de una red. FTP es también habitualmente el nombre del programa que el usuario invoca para ejecutar el protocolo.

H

home page (página raíz, portada,página inicial) Primera página de un servidor WWW hyperlink (hipervínculo, nexo,hiperenlace) Puntero existente en un documento hipertexto que apunta a (enlaza con) otro documento que puede ser o no otro documento hipertexto.

hypertext (hipertexto) Aunque el concepto en sí es muy anterior al WWW (fue creado por el físico norteamericano Vannevar Bush en 1945), en Internet el término se aplica a los enlaces existentes en las páginas escritas en HTML, enlaces que llevan a otras páginas que pueden ser a su vez páginas de hipertexto. Las páginas hipertextuales son accedidas normalmente a través de navegadores WWW.

HyperText Markup Language -- HTML (Lenguaje de Marcado de Hipertexto) Lenguaje en el que se escriben las páginas a las que se accede a través de navegadores WWW. Admite componentes hipertextuales y multimedia

HyperText Transfer Protocol -- HTTP (Protocolo de Transferencia de Hipertexto) Protocolo usado para la transferencia de documentos WWW

I

internaut (internauta) Dícese de quien navega por la red Internet.

internet (La Red,*Internet*) Una internet (con "i" minúscula) es un conjunto de redes conectadas entre sí.
Internet (La Red,*Internet*) Red de telecomunicaciones nacida en 1969 en los EE.UU. a la cual están conectadas centenares de millones de personas, organismos y empresas en todo el mundo, mayoritariamente en los países más desarrollados, y cuyo rápido desarrollo está teniendo importantes efectos sociales, económicos y culturales, convirtiéndose de esta manera en uno de los medios más influyentes de la llamada Sociedad de la Información y en la Autopista de la Información por excelencia. Fue conocida como ARPANET hasta 1974.

Intranet (Intrarred,*Intranet*) Red propia de una organización, diseñada y desarrollada siguiendo los protocolos propios de Internet, en particular el protocolo TCP/IP. Puede tratarse de una red aislada, es decir no conectada a Internet.

Java (Java) Lenguaje de programación desarrollado por la empresa Sun para la elaboración de pequeñas aplicaciones exportables a la red (*applets*) y capaces de operar sobre cualquier plataforma a través, normalmente, de navegadores WWW. Permite dar dinamismo a las páginas web.

Java Server Page -- JSP (Página de Servidor Java) Una página JSP es un tipo especial de página HTML que contiene unos pequeños programas (también llamados *scripts*) que son ejecutados en servidores Netscape antes de ser enviados al usuario para su visualización en forma de página HTML. Habitualmente esos programas realizan consultas a bases de datos y los resultados de esas consultas determinan la información personalizada que se envía a cada usuario específico. Los ficheros de este tipo llevan el sufijo *.jsp*.

JavaScript (JavaScript) Lenguaje de programación para WWW desarrollado por Netscape. Al igual que VBScript, pertenece a la familia Java pero se diferencia de este último en que los programas están incorporados en el fichero HTML.

L

link (liga, puntero, vínculo/vincular,*enlace/enlazar*) Apuntadores hipertexto que sirven para saltar de una información a otra, o de un servidor a otro, cuando se navega por Internet o bien la acción de realizar dicho salto.

N

Netscape Navigator (Navegador Netscape) Navegador WWW creado en 1995 por Marc Andreessen, de la empresa norteamericana Netscape. Es uno de los navegadores Internet más difundidos.

network (red) Una red de ordenadores es un sistema de comunicación de datos que conecta entre sí sistemas informáticos situados en lugares más o menos próximos. Puede estar compuesta por diferentes combinaciones de diversos tipos de redes.

O

off line (off line,*desconectado* ,fuera de línea) Condición de estar desconectado de una red.

on line (on line,*conectado* ,en línea) Condición de estar conectado a una red.

P

page (página) Fichero (o archivo) que constituye una unidad significativa de información accesible en la WWW a través de un programa navegador. Su contenido puede ir desde un texto corto a un voluminoso conjunto de textos, gráficos estáticos o en movimiento, sonido, etc. El término página web se utiliza a veces, a mi entender de forma incorrecta, para designar el contenido global de un sitio web, cuando en ese caso debería decirse páginas web o sitio web.

password (palabra de paso,*contraseña*) Conjunto de caracteres alfanuméricos que permite a un usuario el acceso a un determinado recurso o la utilización de un servicio dado.

Personal Home Page Tools -- PHP (Herramientas para Páginas Iniciales Personales) Lenguaje de programación tipo *script* para entornos Web utilizado sobre todo en servidores Linux para personalizar la información que se envía a los usuarios que acceden a un sitio web. Es un programa de software libre con unas funciones muy semejantes a las de ASP y JSP.

Practical Extraction and Report Language -- PERL (Lenguaje Práctico de Extracción e Informes) Lenguaje de programación muy utilizado para la elaboración de aplicaciones CGI.

program (programa) Conjunto de instrucciones escritas en un determinado lenguaje (por ejemplo, COBOL, C+) que dirigen a un ordenador para la ejecución de una serie de operaciones, con el objetivo de resolver un problema que se ha definido previamente.

response time (tiempo de respuesta) Lapso de tiempo que transcurre entre que un usuario hace una petición a la red y la información pedida es recibida por éste. En Internet depende de múltiples factores, tales como ancho de banda, calidad de la línea, velocidad de módem, congestión de la red. Por definición un usuario nunca está satisfecho con el tiempo de respuesta de la red y se acostumbra rápidamente a las mejoras de éste.

S

script (guión,*script*) Conjunto de caracteres formado por mandatos y secuencias de tecleo, que se utiliza muy a menudo en Internet para automatizar tareas muy habituales como, por ejemplo, la conexión a la red (*login*)[

server (servidor) Sistema que proporciona recursos (por ejemplo, servidores de ficheros, servidores de nombres). En Internet este término se utiliza muy a menudo para designar a aquellos sistemas que proporcionan información a los usuarios de la Red.

servlet (servlet) Aplique o pequeña aplicación Java (*applet*) que se ejecuta en un servidor web y que se envía al usuario junto a una página web con objeto de realizar determinadas funciones, tales como el acceso a bases de datos o la personalización de dicha páginas web.

source code (código fuente) Conjunto de instrucciones que componen un programa informático. Estos programas se escriben en determinados lenguajes; el lenguaje que se utiliza para elaborar una página web, que puede considerarse en cierto sentido un programa, es el HTML.

T

tag (marca,etiqueta,*mandato*) Instrucción que se escribe al elaborar una página HTML. Un ejemplo es <P>, que indica el comienzo de un párrafo de texto. Cada uno de los mandatos que aparecen en una página es interpretado por el programa navegador para visualizar dicha página de forma adecuada en una pantalla.

U

Uniform Resource Locator/Universal Resource Identifier -- URL/URI (Localizador Uniforme de Recursos/Identificador Universal de Recursos) Sistema unificado de identificación de recursos en la red (el URI todavía no está implantado). Las direcciones se componen de protocolo, FQDN y dirección local del documento dentro del servidor. Este tipo de direcciones permite identificar objetos WWW, Gopher, FTP, News,... Ejemplos de URL son: <http://www.anaya.es> o <ftp://ftp.ati.>

UNIX, Unix (Unix,UNIX) Sistema operativo interactivo y de tiempo compartido creado en 1969 por Ken Thompson. Reescrito a mitad de la década de los '70 por ATT alcanzó enorme popularidad en los ambientes académicos, y más tarde en los empresariales, como un sistema abierto, robusto, flexible y portable, muy utilizado en los entornos Internet. De él deriva el sistema operativo Linux

upload (cargar,*subir* ,subirse) En Internet, proceso de transferir información desde un ordenador personal a un servidor de información.

V

Virtual Basic Script -- VBScript (*VBScript* ,Virtual Basic Script) Lenguaje de programación para WWW desarrollado por Microsoft. VBScript y JavaScript, de Netscape, son muy similares.

W

Web web (telaraña,güeb,malla,*web*) Servidor de información WWW. Se utiliza también para definir el universo WWW en su conjunto. En el primer caso quizás debería ir en minúscula; en el segundo, en mayúscula.

Web editor, web editor (editor de Web) Persona que se encarga de gestionar y organizar los contenidos de un servidor WWW. Si comparamos con un periódico, el editor del web sería el director o el jefe de redacción mientras que el administrador de web (*webmaster*) sería el director técnico o el jefe de rotativas.

web server (servidor web) Máquina conectada a la red en la que están almacenadas físicamente las páginas que componen un sitio web. Dícese también del programa que sirve dichas páginas.

Webmaster, webmaster (administrador de Web) Persona que se encarga de la gestión y mantenimiento de un servidor web, fundamentalmente desde el punto de vista técnico; no hay que confundirlo con el editor de web (*webeditor*).

website (sitio web) Colección de páginas web dotada de una dirección web única.

Windows (Windows) Sistema operativo desarrollado por la empresa Microsoft y cuyas diversas versiones (3.1, 95, 98, NT, 2000, Me) dominan de forma abrumadora el mercado de los ordenadores personales. La palabra *windows* significa literalmente "ventanas".

World Wide Web -- WWW W3 (Malla Mundial,Telaraña Mundial,*WWW*) Sistema de información distribuido, basado en hipertexto, creado a principios de los años 90 por Tim Berners-Lee, investigador en el CERN, Suiza. La información puede ser de cualquier formato (texto, gráfico, audio, imagen fija o en movimiento) y es fácilmente accesible a los usuarios mediante los programas navegadores. Es preciso destacar el hecho poco habitual de que tanto Berners-Lee como el CERN renunciaron a la explotación comercial de este extraordinario invento.

Bibliografía

Bobadilla Sancho Jesús
Superutilidades para Webmasters, McGraw-Hill.

Buczek Greg
Instant ASP Scripts, Osborne.

Buyens Jim
Aprenda desarrollo de bases de datos web ya, McGraw-Hill.

Date C.J.
Introducción a los sistemas de bases de datos, Addison Wesley Iberoamericana.

De Carli James
Professional ASP Data Access, Wrok

De Miguel A. y Piattini M.
Concepción y diseño de bases de datos. Del modelo E/R al modelo relacional, Ra-ma.

De Miguel A. y Piattini M.
Diseño de bases de datos relacionales, Ra-ma.

De Miguel A. y Piattini M.
Fundamentos y modelos de bases de datos, Ra-ma.

Elmasri R. y Navathe S.
Sistemas de bases de datos, conceptos fundamentales, Addison Wesley Iberoamericana.

Hansen G.W. y Hansen J.V.
Diseño y Administración de Bases de datos, Prentice-Hall.

Kauffman John
Beginning ASP Databases, Peperback.

Korth H. y Silberschatz A.
Fundamentos de bases de datos, McGraw-Hill.

Lovejoy Elijah
Essential ASP for Web Professionals, Prentice Hall.

Lovejoy Elijah
Guía esencial ASP, Prentice-Hall.

Mellor Robert
ASP Learning by Example, ABF.

Meyer Jeanine
Creating Database Web Applications with PHP and ASP, Peperback.

Mitchell Scott
Designig Active Server Pages, O'Reilly.

Orós Juan Carlos
Diseño de páginas web interactivas con JavaScript,Ra-ma.

Ullman J.D. Widom J.
Introducción a los sistemas de Bases de Datos, Pearson.

Ullman J.D.
Principles of database systems, Computer Science Press.

Vossen G.
Data Models, Database Languages and Database Management System, Addison-Wesley.

Walther Stephen
Teach yourself E-Commerce Programming with ASP in 21 days, Sams.

Weissinger Keyton
ASP in a Nutshell, 2nd Edition, O'Reilly.

Referencias de Internet

http://bibliotecainformatica.iespana.es/bibliotecainformatica/programacion/ASP/
http://www.aspfree.com/
http://www.desarrolloweb.com
http://www.desarrolloweb.com/articulos/244.php?manual=8
http://www.google.com
http://www.pcwebopaedia.com
http://www.uco.es/~i72cafef/tiagdi/bibliografia.html
http://www.webexperto.com/