UNIVERSIDAD LASALLISTA

BENAVENTE

ESCUELA DE INGENIERÍA EN COMPUTACIÓN

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

"CREACIÓN DE UNA PÁGINA WEB DINÁMICA CON PHP Y ASP.NET "

TESIS

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA:

ENRIQUE RAMÓN GARCÍA

ASESOR: **ING. ALEJANDRO GUZMÁN ZAZUETA**

Celaya, Gto. **OCTUBRE del 2006**

AGRADECIMIENTOS

Esta Tesis la dedico a Dios, que me dio vida para poder realizar todo lo que he querido.

A mis padres, por todo el esfuerzo que hicieron para que yo lograra tener una carrera universitaria. Todo ese esfuerzo fue lo que me impulsó para seguir adelante y tratar de superarme.

Gracias mamá por darme ese apoyo, por darme la confianza de que yo podía hacer lo que me propusiera.

Gracias papá por estar siempre interesado en lo que hacía, por los consejos que tuviste hacia mí, y por la misma confianza que me dabas en cada paso que daba.

Le agradezco a mis hermanos por enseñarme que una de las tantas cosas importantes en esta vida es superarse día a día, eso me ayudó mucho para no caer en la mediocridad y seguir con esa filosofía hasta el final.

A mis maestros, que dedicaron tanto tiempo orientándome sobre los aspectos profesionales de mi carrera, y muchas veces aspectos personales. Gracias por enseñarme el potencial que tengo y cómo llegar a desarrollarlo.

Y por último, muchas gracias a mi asesor. Gracias Ing. Alejandro Guzmán Zazueta por enseñarme todas las utilidades y facilidades que te puede dar el echarle ganas y dar un poquito más de esfuerzo que los demás. Eso me ha ayudado mucho, y por supuesto también los múltiples conocimientos de programación que me dio.

INDICE

INTRODUCCIÓN

Por la creciente demanda de servicios y oportunidades tecnológicas el mercado mundial se ha posicionado dentro de la WEB, y esto hace que esta presencia sea cada vez de mejor calidad diariamente. Ahora prácticamente todos los negocios que quieren ser reconocidos por el público cuentan con página Web propia, ya pueden ser páginas Web estáticas o dinámicas.

Las páginas Web estáticas se están quedando muy atrás con respecto a las dinámicas, ya que no pueden ofrecer los servicios hacia el navegador como lo hacen las dinámicas, por lo tanto es importante que pueda conocer un empresario de nivel medio-bajo el desarrollo de estas páginas, porque solo así podrá competir con los grandes consorcios que acaparan el mercado nacional e internacional.

Es importante destacar que en la gran mayoría de las páginas que utilizan transacciones, acceso a información grande, etc., se tiene en uso las páginas dinámicas. Lo que quiero es darle una perspectiva a las personas con conocimientos básicos de lenguajes de programación para que tenga un esparcimiento mayor sobre el desarrollo de estas páginas, ya que facilitan el uso de los recursos de las empresas, así como disminuir tiempo en acceso a la información y que además, es la misma información que tienen en la empresa, porque con estas páginas se accede a base de datos y esto puede hacer que desde cualquier lugar del mundo se verifique esta información. Además de que le dará un crecimiento enorme a la calidad de la empresa, y una mejor imagen ante sus clientes.

Por todo esto es que se requiere comprender ¿qué son las páginas dinámicas?, ¿qué es lo que contienen?, ¿de qué forma benefician a la empresa?, ¿de qué forma favorecen al acceso a la información y transacción de datos, productos, etc.?

La hipótesis correspondiente es, que al conocer cómo se puede desarrollar una página Web dinámica con los 2 tipos de lenguajes de programación de páginas Web más comunes en la actualidad, se aplique hacia un negocio u empresa, la cuál tendrá que tener una base de datos de sus productos. Se facilitará el manejo de su información ya que podrá ser manipulada desde cualquier parte del mundo, así mismo se les dará un servicio de excelencia a sus clientes, ya que contarán con la información del negocio al instante. Por lo tanto tendrá un impacto público nacional e internacional, y este impacto mostrará una creciente ganancia dentro del negocio.

Para tener este tipo de ganancias hay que tener primero los conocimientos de cómo llegar a lograrlo, para esto se explica en el capítulo 1 una breve cronología de lo que es el Internet en México y en el mundo, además de la composición de una página Web, y el lenguaje que trabaja, todo esto con el fin de tener un criterio amplio del los cambios que se han dado y de cómo se fue trabajando para poder llegar a lo que tenemos en la actualidad, y empezar a comprender cómo se trabaja con esta tecnología.

Después se explicará en el capítulo 2 un poco de los manejadores de base de datos, los lenguajes de programación y los programas que se pueden utilizar en las páginas dinámicas, esto para poder elegir cuál es el que más le convendría al programador, o al negocio. Para esto se necesita que el programador tenga conocimientos básicos de creación de base de datos y de lenguajes de programación (como C, HTML, Visual Basic y .Net).

Se tendrá que conocer cuáles son los requerimientos de la empresa, hacia quién va dirigida y sus características, esto se muestra en el capítulo 3, para poder identificar cuáles son las partes que se tienen que tomar en cuenta para resaltar dentro de la página, y para dar a conocer a los clientes un amplio esquema de lo que se quiere vender y de cómo trabaja el negocio.

Seguirá en el capítulo 4 la explicación detallada de cómo poder crear una página Web dinámica sobre una empresa específica, con el fin de dar entendimiento del gran potencial que se tiene con este tipo de tecnología. Por lo tanto se tomarán las características de un negocio, cuáles son sus tipos de movimientos y requerimientos que tiene, y sus alcances hacia sus clientes, y de ahí partir para crear su página Web dinámica.

El desarrollo de esta metodología ayudará en muchísimo el tiempo de creación de páginas, y de organizar las ideas de cómo realizarlo; y ni que decir del beneficio hacia los negocios que lo apliquen, en difusión, tiempo de negociaciones, de transacciones, y por supuesto de ganancias monetarias.

CAPÍTULO 1

FUNDAMENTACIÓN

1.1 HISTORIA DE INTERNET

Internet ha trasformado las comunicaciones absolutas y actualmente forma parte de los negocios en el mundo entero.

Las primeras páginas Web tal como las conocemos hoy, aparecieron hace 10 años atrás, con la publicación de un documento acerca de partículas físicas del científico Paul Kunz del Stanford Linear Accelerator Center, configurando, a su vez, el primer servidor Web fuera de EE.UU., concretamente en Europa, cuando colocó allí los archivos de sus estudios.

Si se busca personalizar la invención de Internet, según analistas, el crédito debiera ir hacia Tim Berners-Lee, investigador inglés que trabajó para el laboratorio CERN en Génova, Suiza, en el que realizó las primeras investigaciones. Hoy pertenece al W3C (World Wide Web Consortium).

Esta forma de comunicación de datos llamó la atención a muchos investigadores, luego a inversionistas y, como sabemos hoy, a millones de personas.

De manera resumida la cronología de Internet puede ser la siguiente:

1966

El inventor inglés Donald Davies lanza su idea de enrutamiento de "paquetes" (vulgarmente denominado ruteo) e interesa al NPL, Laboratorio Nacional de Física de Inglaterra, en la construcción de una red de computadoras para probar la validez de su revolucionaria idea[1].

1969

La comisión del Departamento de Defensa de EE.UU., ARPAnet, desarrolló los primeros elementos técnicos.

[1] En http://www.aunmas.com/future/internet_historia

1985

Se crea el primer sitio con un nombre de dominio: Symbolics.com.

1990

Nace el primer proveedor comercial de acceso a Internet a través de discado, denominado "The World".

Chile se conecta a la NSFNet, junto con Argentina, Brasil, Australia, Bélgica, Grecia, India, Irlanda, Corea, España y Suiza.

1991

Mark MaCahill, de la Universidad de Minnesota, presenta el primer Gopher de tipo "point - and click".

Se lanza en el CERN Research Center, la primera World Wide Web, desarrollada por Tim Berners Lee.

Paul Kunz configura un servidor Web en el Stanford Linear Acelerador Center. Marc Andreesen, estudiante del NCSA de la Universidad de Illinois, desarrolla MOSAIC, el primer browser.

Philip Zimmerman lanza Pretty Good Privacy (PGP), y Apple Lanza Quick Time, software que posibilita ver películas en todo tipo de computadores, trasformándose en estándar de la industria.

El tráfico total excede el trillón de bytes o 10 billones de paquetes por mes. Más de 100 países están interconectados con 600.000 computadoras y con aproximadamente 5.000 redes separadas.[2]

1992

El bibliotecario Jean Armour Polly acuña el término "surfing the Internet". Se realiza las primeras transmisiones de voz y vídeo sobre Internet.

En enero se crea la Internet Society. John Sculley, de Apple Computers, acuña el termino personal Digital Assistant (PDA) al referirse a mini-

[2] En http://www.aunmas.com/future/internet_historia

computadores (las versiones de prueba Newton son mostradas en 1993).

Nace la revista ComputerWord en Chile.

1993

La Casa Blanca crea su primer sitio Web.

Aparece el primer browser MOSAIC gráfico.

1994

Aparece el primer banner comercial, en el sitio Hotwired.com.

Linus Torvalds Lanza la versión 1.0 del Kernel de Linux.

Marc Andreesen y Jin Clark fundan Netscape Communications.

Nace el comercio electrónico por medio de Internet, se puede pedir pizzas a Pizzas Hut, y el primer cyber-banco, First Virtual, abre sus puertas.

La primera cyber-radio, RTFM, transmite desde Las Vegas.

En julio se crea el Word Wide Web Consortium (W3C) con el objeto de trabajar en pos de la estandarización de sistemas aplicados a la Web.

Se inaugura proyecto Access Nova, entre Universidad de Chile y NTT, en visita de presidente Frei a Japón.

David Filo y Jerry Yang, estudiantes del doctorado de ingeniería eléctrica en la Universidad de Stanford fundaron Yahoo! en abril de 1994 como manera de seguir sus propios intereses en Internet.[3]

1995

En abril, nacen los primeros seis Internet Backbone Providers comerciales (PSINet, UUNET, ANS, AOL, Sprint, y MCI), y 300 ISPs se conectan a ellos.

Netscape Communication lanza la primera gran IPO (Initial Public Offering) en el Nasdaq.

[3] En http://www.aui.es/historia/ihistoria.htm

La National Science Foundation privatiza NSFNet, dejando Internet en manos privadas.

El 23 de mayo Sun Microsystems lanza Java, desarrollo por James Gosling y su equipo de trabajo.

Microsoft lanza su browser Internet Explorer, y Windows 95. Real Audio es lanzado al mercado.

1996

Comienza la "guerra de los Browser entre Netscape y Microsoft, dando pie a un nuevo escenario en el desarrollo de software, con actualizaciones cada 3 meses.

1997

El dominio business.com se vende en US $150.000.

1998

La empresa Network Solutions registra 2 millones de dominios. Netscape hace público código fuente de su navegador.

El Departamento de Comercio de los Estados Unidos presenta su propuesta de privatización de DNS.

1999

Napster lanza su servicio de intercambio de archivos musicales.

2000

Se lanza un masivo ataque hacker contra los sitios mas grandes, incluyendo Yahoo, Amazon y Ebay.

2001

Napster es forzada a suspender su servicio.

Aparecen nuevos dominios: .biz y .info entre otros.

Internet hace referencia a un sistema global de información que:

1. Está relacionado lógicamente por un único espacio de direcciones global basado en el protocolo de Internet (IP) o en sus extensiones.

2. Es capaz de soportar comunicaciones usando el conjunto de protocolos TCP/IP o sus extensiones u otros protocolos compatibles con IP.

3. Emplea, provee, o hace accesible, privada o públicamente, servicios de alto nivel en capas de comunicaciones y otras infraestructuras relacionadas.

Internet ha cambiado en sus dos décadas de existencia. Fue concebida en la era del tiempo compartido y ha sobrevivido en la era de los ordenadores personales, cliente-servidor, y los network-computer. Se ideó antes de que existieran las LAN, pero ha acomodado tanto a esa tecnología como a ATM y la conmutación de tramas. Ha dado soporte a un buen número de funciones desde compartir ficheros, y el acceso remoto, hasta compartir recursos y colaboración, pasando por el correo electrónico y, recientemente, el World Wide Web. Pero, lo que es más importante, comenzó como una creación de un pequeño grupo de investigadores y ha crecido hasta convertirse en un éxito comercial con miles de millones de dólares anuales en inversiones.

No se puede concluir diciendo que Internet ha acabado su proceso de cambio. Aunque es una red por su propia denominación y por su dispersión geográfica, su origen está en los ordenadores, no en la industria de la telefonía o la televisión. Puede (o mejor, debe) continuar cambiando y evolucionando a la velocidad de la industria del ordenador si quiere mantenerse como un elemento relevante. Ahora está cambiando para proveer nuevos servicios como el transporte en tiempo real con vistas a soportar, por ejemplo, audio y vídeo.

La disponibilidad de redes penetrantes y omnipresentes, como Internet, junto con la disponibilidad de potencia de cálculo y comunicaciones asequibles en máquinas como los ordenadores portátiles, los PDA y los teléfonos celulares, está posibilitando un nuevo paradigma de informática

y comunicaciones "nómadas".

La cuestión más importante sobre el futuro de Internet no es cómo cambiará la tecnología, sino cómo se gestionará esa evolución.

Con el éxito de Internet ha llegado una proliferación de inversores que tienen intereses tanto económicos como intelectuales en la red.

Se puede ver en los debates sobre el control del espacio de nombres y en la nueva generación de direcciones IP una pugna por encontrar la nueva estructura social que guiará a Internet en el futuro. Será difícil encontrar la forma de esta estructura dado el gran número de intereses que concurren en la red. Al mismo tiempo, la industria busca la forma de movilizar y aplicar las enormes inversiones necesarias para el crecimiento futuro, por ejemplo para mejorar el acceso del sector residencial.

Si Internet sufre unos traspiés no será debido a la falta de tecnología, visión o motivación. Será debido a que no se ha podido hallar la dirección justa por la que marchar hacia el futuro.

1.2 CRONOLOGÍA DE INTERNET EN MÉXICO

1989

- México se conecta al Internet por primera vez; el Instituto Tecnológico y de Estudios Superiores de Monterrey (campus Monterrey) ITESM se conecta hacia la Universidad de Texas en San Antonio (UTSA), por medio de una línea dedicada analógica a 9600 bps; siendo el primer nodo de Internet en este país.[4]

- Se conecta la Universidad Nacional Autónoma de México (UNAM) vía satélite de 56 Kbps con el Instituto de Astronomía en la Ciudad de México con el Centro Nacional de Investigación Atmosférica (NCAR)

[4] En http://www.enterate.unam.mx/Articulos/mayo/mx.htm

de Boulder, Colorado, en los Estados Unidos.

1992

- Enero - surge MEXnet, formada por: ITESM, Universidad de Guadalajara, Universidad de las Américas, ITESO, Colegio de Postgraduados, LANIA, CIQA, Universidad de Guanajuato, Universidad Veracruzana, Instituto de Ecología, Universidad Iberoamericana, IT de Mexicali.

- Junio - MEXnet establece una salida digital de 56 kbps al Backbone de Internet.

1993

- El CONACYT y el ITAM se conectan a Internet; existían ya una serie de Redes en el País: MEXnet, Red UNAM, Red ITESM, BAJAnet, Red Total CONACYT, RUTyC (desapareció ese mismo año), SIRACyT.

1994

- Se forma la Red Tecnológica Nacional (RTN), Backbone nacional integrado por MEXnet y CONACyT y agrupando a un gran número de instituciones educativas y comerciales en toda la República, desde Baja California hasta Quintana Roo.

- Internet se abre a nivel comercial en nuestro país, ya que hasta entonces, solamente instituciones educativas y de investigación podían realizar su enlace a Internet.

1995

- Diciembre - Se crea el Centro de Información de Redes de México (NIC -México) el cual se encarga de la coordinación y administración de los recursos de Internet asignados a México, tales como la administración y delegación de los nombres de dominio ubicados

bajo .MX.

1996

- El INFOTEC crea el Centro de Tecnologías Avanzadas cuyo objetivo es desarrollar servicios de contenido de valor agregado en Internet y todos aquellos que se desarrollan con nuevas tecnologías del Intranet y de Multimedia.

- Nace la Sociedad Internet, Capítulo México, una asociación internacional no gubernamental no lucrativa para la coordinación global y cooperación en Internet.

1997

- Existen más de 150 Proveedores de Acceso a Internet (ISP's) que brindan sus servicios en el territorio mexicano, ubicados en los principales centros urbanos: Ciudad de México, Guadalajara, Monterrey, Chihuahua, Tijuana, Puebla, Mérida, Nuevo Laredo, Saltillo, Oaxaca, por mencionar sólo algunos.

1999

- El instituto Mexicano de la Propiedad Industrial (IMPI), le solicitó oficialmente por primera vez a NIC México, la suspensión de un dominio por cuestiones de propiedad industrial; este dominio en controversia fue nestle.com.mx.

  Más tarde, la filial mexicana de Nestlé recobró su legítimo derecho a usar su nombre en Internet.[5]

[5] En http://www.enterate.unam.mx/Articulos/mayo/mx.htm

1.3 HISTORIA DE LA WORLD WIDE WEB

La Web se inicia en marzo de 1989 propuesto por el investigador Tim Berners-Lee perteneciente al CERN, como un proyecto de desarrollo de un sistema de hipertexto, es decir, un sistema de creación y distribución de documentos, que permitiera compartir información desarrollada en diferentes aplicaciones, de forma sencilla y eficiente, entre equipos de investigadores ubicados en distintos lugares geográficos.

Luego se abocaron a desarrollar una solución que permitiera cubrir los siguientes requerimientos:

- Obtener una interfaz consistente, es decir, el sistema debería permitir una conexión que al menos asegurara una transferencia de datos consistente (lo que envío es obtenido intacto).
- Permitir incorporar un amplio rango de tecnologías y distintos tipos de documentos.
- Proveer de una herramienta que permita leer los documentos desde cualquier lugar y por cualquier individuo que este navegando dentro de este almacén, y deberá permitir que cualquier documento sea accesible en forma paralela por dos o más personas de forma sencilla.

Luego de esta primera etapa, se comenzó a desarrollar lo que conocemos como browser para poder acceder a la información desde cualquier lugar de forma rápida, sencilla, e independiente de la plataforma utilizada por el usuario.

El primer browser, construido en 1992, fue llamado WWW y estaba orientado al trabajo vía FTP desde el CERN. Paralelamente se encontraba disponible para sistemas X Windows el browser Viola quien fue líder en este campo, ya que permitió los primeros vistazos a gráficos y un sistema de hipertexto basado en mouse (no olvidemos que este era uno de los objetivos principales de la Web).

A principios de 1993 se vio surgir al browser Mosaic, que cumplía con todos los requerimientos que se buscaban (funcionamiento en diversas plataformas, poseer una interfaz grafica y fácil de usar), lo que produjo su éxito inmediato. Después aparecieron Netscape de Netscape Inc. e Internet Explorer de Microsoft. Finalmente a fines de 1994 y principios de 1995, se formó el Consorcio World Wide Web o W3C que esta bajo la dirección del fundador de la Web.

Respecto al tamaño de la Web, es necesario además dar una lista de datos referentes a su crecimiento, para tener una visión más amplia del futuro y de la razón que hace urgente el organizar esta información de manera que optimice su acceso.

- Actualmente la Web esta constituida aproximadamente por 72 millones de sitios Web, lo que es aproximadamente unas 1000 millones de páginas.
- Diariamente son creadas alrededor de 1.5 millones de páginas.
- La Web duplica su tamaño cada 8 meses, en promedio.
- El espacio ocupado por la Web completa es de aproximadamente 7 terabytes.

La cantidad de información contenida en la Web, que crece día a día, representa un reto, tanto para las personas que se dedican a buscar información, como para las personas que hacen sus páginas y quieren que tengan alta frecuencia de visita.

Desde un punto de vista de recuperación de la Web, los grandes desafíos son:

- Datos distribuidos sobre diferentes plataformas.
- Datos volátiles.
- Gran volumen de datos.
- Datos redundantes y no estructurados.

- Calidad de los datos.
- Datos heterogéneos.

1.3.1. Buscadores

Actualmente se conocen dos mecanismos de búsqueda, mejor llamados Buscadores:

- **Directorios o buscador sin Robot**:

Estos buscadores son administrados por personas, lo que los hace más eficientes al momento de buscar información, ya que representan más o menos los requerimientos que podría tener algún usuario al momento de buscar cierta información. Debido a esta razón están limitados en el tamaño de la base de datos que contiene los documentos indexados, es decir, limitan el campo de búsqueda, ya que estos se indexan después de un análisis que también es hecho por personas, por lo que su crecimiento se limita a la velocidad de trabajo de las mismas (incluye actividades como: agregar páginas nuevas, eliminar páginas que han expirado o que han sido eliminadas, etc.).

- **Máquinas de búsqueda o buscador con Robot**

La indexación de documentos es realizada automáticamente a través de Software que indexa las páginas existentes en la Web.

1.3.2. Ranking

No hay información clara de cómo las máquinas de búsquedas realizan un ranking, pero la mayoría usa variaciones del modelo vector o Boolean.

Una de las mayores diferencias en el ranking para la recuperación de información en la WWW es el uso de hyperlinks.

El número de hyperlinks (enlaces) que apuntan a una página sirve como una medida de su popularidad y calidad.

Enlaces comunes es también una medida de relación de las tópicas tratadas en las páginas.

Ejemplos de ranking basados en enlaces son los siguientes:

- **WebQuery**: da un rango a las respuestas a una consulta en base a cuán conectadas están.

  Adicionalmente, extiende el grupo de respuestas a las páginas altamente conectadas al grupo inicial de respuestas.

- **HITS***:* este esquema considera el grupo de páginas S que apuntan al grupo de respuestas originales y las páginas que son apuntadas desde el grupo de respuestas originales.

  Páginas en el grupo de respuestas que tienen muchos enlaces apuntando a ellas son consideradas relevantes (autoridades).

  Páginas que tienen muchos enlaces de salida son conectores (ellas deberían apuntan a contenido similar).

  Conectores y autoridades son conceptos que se retroalimentan: mejores autoridades son derivadas por enlaces desde buenos conectores y buenos conectores vienen de enlaces desde buenas autoridades.

- **PageRank**: se basa en la idea de que existe un usuario aleatorio a quien se le asigna una página en forma aleatoria y se mantiene avanzando por los enlaces de esa página hasta que se aburre y salta a otra página en forma aleatoria.

  En este contexto, la relevancia de una página está determinada por la probabilidad que el usuario visite esa página y la relevancia de las páginas que apuntan a ella.

1.4 LAS URLS

Una URL nos indica tanto una dirección de Internet como el servicio que se espera que ofrezca el servidor al que corresponde la dirección. Tiene el siguiente formato:

servicio://maquina:puerto/ruta/fichero@usuario

Donde el servicio podrá ser uno de los siguientes:

http

Es el servicio invocado para transmitir páginas Web y el que se usa normalmente en los enlaces.

https

Es una innovación sobre el anterior, que permite acceder a servidores (generalmente comerciales) que ofrecen el uso de técnicas de encriptación para proteger los datos que se intercambian con terceras personas.

TFP

Permite trasmitir ficheros desde servidores de TFP anónimo.

Si no se le pide un fichero sino un directorio, en general el navegador se encargará de mostrar el contenido del mismo para poder escogerlo cómodamente.

Utilizando la @ se puede acceder a servidores privados.

mailto

Para poder mandar un mensaje.

Por ejemplo, la URL mailto:quiquehanck@hotmail.com mandaría un mensaje a esta dirección.

news

Para poder acceder a foros de discusión. Se indica el servidor y el grupo.

Por ejemplo news://news.ibernet.es/es.comp.demos conectaría con el foro es.comp.demos en el servidor nacional de Telefónica.

telnet

No es implementado generalmente por los navegadores, que suelen invocar un programa externo. Permite la conexión con otros ordenadores y entrar en ellos como si nuestro ordenador fuese una terminal del mismo.

1.5 EL PROTOCOLO HTTP

El Protocolo de Transferencia de Hipertexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente - servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP.[6] Fue propuesto por Tim Berners -Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto, el 80), y espera las solicitudes de conexión de los clientes Web.

Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en sencillas operaciones de solicitud / respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado.

[6] En http://cdec.unican.es/libro/HTTP.htm

Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan; cada objeto Web (documento HTML, fichero multimedia o aplicación CGI) es conocido por su URL.

Los recursos u objetos que actúan como entrada o salida de un comando HTTP están clasificados por su descripción MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido.

La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permitirá que el receptor trate adecuadamente los datos.

1.5.1. Etapas de una transacción HTTP

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

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

2. El cliente Web descodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección DNS o IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.

3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.

4. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la versión del protocolo HTTP empleada y un conjunto variable de información, que incluye datos sobre las capacidades del navegador, datos opcionales para el servidor,...

5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.

6. Se cierra la conexión TCP.

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

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

1.5.2. Comandos del protocolo

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

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

Figura 1. Formato del comando general de HTTP

Cada comando actúa sobre un objeto del servidor, normalmente un fichero o aplicación, que se toma de la URL de activación.
La última parte de esta URL, que representa la dirección de un objeto dentro de un servidor HTTP, es el parámetro sobre el que se aplica el

comando. Se compone de una serie de nombres de directorios y ficheros, además de parámetros opcionales para las aplicaciones.

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

GET

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

HEAD

Solicita información sobre un objeto (fichero): tamaño, tipo, fecha de modificación, etc.

Es utilizado por los gestores de cachés de páginas o los servidores Proxy, para conocer cuándo es necesario actualizar la copia que se mantiene de un fichero.

POST

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

La operación que se realiza con la información proporcionada depende de la URL utilizada.

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

El envío del contenido de un formulario utiliza GET o POST, en función del atributo de <FORM METHOD="..."> . Además, si el cliente Web tiene un caché de páginas recientemente visitadas, puede utilizar HEAD para comprobar la última fecha de modificación de un fichero, antes de traer una nueva copia del mismo.

1.5.3. Comandos adicionales (HTTP/1.1)

PUT

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

DELETE

Elimina el documento especificado del servidor.

LINK

Crea una relación entre documentos.

UNLINK

Elimina una relación existente entre documentos del servidor.

1.6 JAVASCRIPT

Javascript es un lenguaje de programación utilizado para crear pequeños programitas encargados de realizar acciones dentro del ámbito de una página web. Con Javascript es posible crear efectos especiales en las páginas y definir interactividades con el usuario.

El navegador del cliente es el encargado de interpretar las instrucciones Javascript y ejecutarlas para realizar estos efectos e interactividades, de

modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador.
Entre las acciones típicas que se pueden realizar en Javascript existen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, javascript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

JavaScript se realiza dentro del propio documento HTML. Esto quiere decir que en la página se mezclan los dos lenguajes de programación, y para que estos dos lenguajes se puedan mezclar sin problemas se han de incluir unos delimitadores que separan las etiquetas HTML de las instrucciones Javascript.
Estos delimitadores son las etiquetas <SCRIPT> y </SCRIPT>. Todo el código Javascript que se ponga en la página ha de ser introducido entre estas dos etiquetas.

En una misma página es posible introducir varios scripts, cada uno que podría introducirse dentro de unas etiquetas <SCRIPT> distintas. La colocación de estos scripts es indiferente, pero en determinados casos esta colocación será muy importante.

También se puede escribir Javascript dentro de determinados atributos de la página, como el atributo onclick. Estos atributos están relacionados con las acciones del usuario y se llaman manejadores de eventos.

CAPÍTULO 2

MANEJADORES DE BASE DE DATOS Y LENGUAJES DE PROGRAMACION

2.1 INTRODUCCIÓN A BASE DE DATOS

Un gestor de base de datos es un programa que permite introducir, almacenar, ordenar y manipular datos. Deben permitir en principio:

- Introducir datos.
- Almacenar datos.
- Recuperar datos y trabajar con ellos.

La organización de las bases de datos mediante distintas tablas relacionadas por campos comunes se le llama Base de Datos Relacional. Cuando se utiliza una sola tabla hablamos de una Base de Datos Plana.

No todos los programas de gestión de base de datos tienen esta capacidad de manejar bases de datos relacionales. Por eso, antes de elegir uno, se debe considerar si es necesario o no esta capacidad.

Hoy en día todos los gestores de base de datos tienen la capacidad relacional. Algunos de los mas conocidos son: Oracle, Fox, Access, Postgres, Microsoft SQL Server.

2.1.1. Justificación de base de datos

La finalidad sobre el manejo consistente de la base de datos es obtener una mejor solución con métodos preventivos sobre los siguientes aspectos:

- Análisis de los sistemas tradicionales.
- Redundancia.
- Dificultad de mantenimiento (Actualización).
- Consistencia de datos.
- Peticiones inesperadas.
- Aumento de tiempo de CPU.
- Problemas de Seguridad

2.1.2. Definición de base de datos

Es un conjunto de información que tiene un significado y que se encuentra organizada en entidades de manera que se minimice la redundancia, mantenga la integridad y seguridad de la información.

> "Es un conjunto ordenado de datos los cuales son manejados según la necesidad del usuario, para que un conjunto de datos pueda ser procesado eficientemente y pueda dar lugar a información, primero se debe guardar lógicamente en archivos." [1]

Desde el punto de vista de la informática, una base de datos es un sistema formado por un conjunto de datos almacenados en disco que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

2.1.3. Ventajas de las bases de datos

Algunas ventajas sobre los mecanismos que se deben utilizar para proteger las bases de datos son:

- Independencia de datos y tratamiento.
- Cambios en datos no implica cambios en programas y viceversa.
- Coherencia en resultados.
- Reduce redundancia.
- Se evita inconsistencia.
- Mejora en la disponibilidad de datos.
- Cumplimento de ciertas normas.
- Restricciones de seguridad.
- Operaciones sobre sus datos.
- Más eficiente gestión de almacenamiento.
- Permite mantener la integridad en la información.

[1] En http://www.itlp.edu.mx/publica/tutoriales/basedat1/tema1_1.htm

2.1.4. El sistema manejador de base de datos (DBMS)

Es un conjunto de programas que se encarga de manejar la información y todos los accesos a la base de datos. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta.

Una de las ventajas del DBMS es que puede ser invocado desde programas de aplicación que pertenecen a sistemas tradicionales escritos en un lenguaje de alto nivel, para la creación o actualización de la base de datos o bien para efectos de consulta a través de lenguajes propios.

2.1.5. Modelo entidad - relación (E-R)

Se basa en la percepción de un mundo real que consiste en un conjunto de objetos básicos llamado Entidad y de relaciones entre estos objetos.

Se realiza para facilitar el desarrollo de bases de datos, permitiendo especificar un esquema conceptual, este esquema representa la estructura lógica general de la base de datos.

Entidades, Atributo y Relación

- Una entidad es un objeto que existe y puede distinguirse de otros objetos.
- Una entidad en la base de datos se define como un conjunto de Atributos del mismo.
- Una relación es una asociación entre varias entidades

Límites de mapeo

En un esquema entidad - relación puede definir ciertas limitantes con las que debe cumplir los datos, pero una de las más importantes es la cordialidad de mapeo, que expresa el número de entidades con los que puede asociarse a otra entidad mediante una relación.

Para un conjunto binario de relaciones R entre los conjuntos de entidades A y B la cordialidad de mapeo debe de ser una de las siguientes:

- Uno a Uno. Una entidad A esta asociada únicamente con una entidad B y viceversa.
- Uno a Muchos. Una entidad A esta asociada con cualquier cantidad de identidades en B y una identidad B solo puede relacionarse con una identidad A.
- Muchos a Uno. Una identidad A esta asociada con una identidad B y una identidad B esta asociada con muchas identidades en B.
- Muchos a Muchos. Una entidad en A esta asociada con cualquier cantidad en B y una Entidad en B esta asociada con cualquier cantidad de entidades en A.

Características de las Entidades y Relaciones

- Atributo: Propiedad característica que tiene un tipo de entidad o relación.
- Valores. Contenido correcto de los atributos.
- Dominio. Conjunto de los posibles valores de los atributos.

Componentes gráficos de la E - R

- Rectángulo. Conjunto de entidades.
- Elipse. Representan atributos.
- Rombo. Relaciones entre conjunto de entidades.
- Líneas. Unen los atributos con los conjuntos y los conjuntos de entidades con la relaciones.

2.1.6. Usuarios

En los sistemas con base de datos se cuenta con tres tipos de Usuarios:

- El Programador de Aplicaciones. Quien es el encargado de escribir los programas que manipulan los datos.
- El Usuario Final. Quien interactúa con los sistemas creados por el programador de aplicaciones.
- El Administrador de Bases de Datos o DBA.

2.2 TIPOS DE MANEJADORES DE BASE DE DATOS

2.2.1. SQL Server 2000 (Structure Query Language)

Es un Lenguaje que surge de un proyecto de investigación para el acceso a Base de Datos Relacionales.

Hoy en día, es el estándar del lenguaje de base de datos y la mayoría de los sistemas lo soporta.

Como su nombre lo indica, el SQL Server permite hacer consultas a la base de datos; las sentencias SQL se clasifican según su finalidad donde rigen los siguientes sublenguajes:

DLL

Lenguaje de definición de datos, permite crear y definir nuevas bases de datos, campos e índices.

- Create. Utilizado para crear nuevas tablas, campos e índices.
- Drop. Empleado para eliminar tablas e índices.
- Alter. Utilizado para modificar las tablas, agregando campos o combinado la definición de los campos.

DML

Permite generar consultas para eliminar, filtrar y extraer datos de la base de datos.

- Select. Utilizado para construir registros de la base de datos que satisfaga un criterio determinado.
- Insert. Utilizado para cargar lotes de datos en la base de datos en una unida operación.
- Update. Utilizado para modificar los valores de campos y registros especificados.
- Delete. Utilizado para eliminar registros de una tabla de base de datos.

El motor de base de datos de Microsoft SQL Server 2000 se ejecuta como un servicio en los sistemas operativos Microsoft Windows NT o Microsoft Windows 2000. No se ejecuta como servicio en Microsoft Windows 98 porque este sistema operativo no admite servicios.[2] Como consecuencia, SQL Server 2000 puede ser utilizado para Windows XP.

El servicio SQL Server asigna recursos del equipo de un modo eficaz entre varios usuarios simultáneos. También implementa reglas de empresa definidas en procedimientos almacenados y desencadenadores, asegura la coherencia de los datos y evita problemas lógicos como el tener a dos personas intentando actualizar los mismos datos a la vez.

2.2.1.1. Tipos de Datos

SQL Server 2000 así como los demás programas para desarrollar base de datos tienen la similitud de manejar muchos tipos de datos, aunque a veces cambia el tipo de declaración son universales, y son los siguientes:

---

[2] En DELGADO, Albert , ***Microsoft SQL Server 2000. Edición especial***, Editorial Pearson Educación, España, 2001, p. 325.

- Números exactos (Integres)
  - Bigint. Números enteros comprendidos entre -2^{63} y $2^{63}-1$
  - Int. Números enteros comprendidos entre -2^{31} y $2^{31}-1$
  - Smallint. Números enteros comprendidos entre -215 y 215-1
  - Tinyint. Números enteros comprendidos entre 0 y 255
  - Bit. Datos enteros con valor 1 y 0
- Números decimales
  - Decimal. Números de precisión y escala fijas. Cuando se utiliza la precisión máxima, los valores permitidos están comprendidos entre $-10^{38}+1$ y $10^{38}-1$
  - Numeric. Funcionalmente equivalente a decimal
- Monetario
  - Money. Valores de moneda comprometidos entre -263 y 263-1 con una precisión de una diezmilésima de la unidad monetaria
  - Smallmoney. Valores de moneda comprometidos entre -214,748.3648 y 214,748.3647 con una precisión de una diezmilésima de la unidad monetaria
- Numéricos con aproximación
  - Float. Números con precisión de coma flotante comprendidos entre -1.79E +308 y 1.79E +308
  - Real. Números con precisión de coma flotante comprometidos entre -3.40E +38 y 3.40E +38
- Tiempo
  - Datetime. Datos de fecha y hora comprometidos entre el 1 de enero de 1753 y el 31 de diciembre de 9999, con una precisión de 3.33 milisegundos

        - Smalldatetime. Datos de fecha y hora comprometidos entre el 1 de enero de 1900 y 6 de junio de 2079, con una precisión de un minuto.
        - Timestamp. Un número único para toda la base de datos que se actualiza cada vez que se actualiza una fila
    - Cadena de caracteres
        - Char. Datos de caracteres no Unicode de longitud fija con una longitud máxima de 8,000 caracteres
        - Varchar. Datos no Unicode de longitud variable con una longitud máxima de 8,000 caracteres
        - Text. Datos no Unicode de longitud variable con una longitud máxima de 231 -1 caracteres
    - Cadenas binarias
        - Binary. Datos binarios de longitud fija con una longitud máxima de 8,000 bytes
        - Varbinary. Datos binarios de longitud variable con una longitud máxima de 8,000 bytes
        - Image. Datos binarios de longitud variable con una longitud máxima de 2,147,483,647 bytes

Además de existen los operadores referentes a las instrucciones de SQL Server y son los siguientes:

Operadores de comparación

Operador	Significado
=	Igual a
>	Mayor que
<	Menor que
>=	Mayor igual que
<=	Menor igual que
<>	Diferente
!=	No igual a
!>	No es mas grande que
!<	No es menor que

Tabla 1. Operadores de comparación de SQL

Operadores lógicos

Operador	Significado
AND	Verdadero si ambas expresiones son verdaderas
OR	Verdadera si alguna expresión es verdadera
NOT	La negación de un valores boléanos
BETWEEN	Verdadero si el operando se encuentra en el rango
LIKE	Verdadero si el operando se equipara a un modelo
IN	Verdadero si el operando es igual a uno de una lista
SOME	Verdadero si alguno de un conjunto de comparaciones son verdaderas
ANY	Verdadero si cualquiera de un conjunto de comparaciones son verdaderas
ALL	Verdadero si todo el conjunto de comparaciones son verdaderas

Tabla 2. Operadores lógicos de SQL

2.2.1.2. Integridad de Datos

La integridad de datos garantiza la calidad de los datos de la base de datos. Hay cuatro categorías de integridad de datos:

- Integridad de entidad
- Integridad de dominio
- Integridad referencial
- Integridad definida por el usuario

Hay varias maneras de forzar cada tipo de integridad

Tipo de integridad	Opciones recomendadas
Entidad	Restricción PRIMARY KEY Restricción UNIQUE Propiedad IDENTITY
Dominio	Definición predeterminada (DEFAULT) Restricción FOREIGN KEY Restricción CHECK, RULE NOT NULL
Referencial	Restricción FOREIGN KEY Restricción CHECK
Definida por el Usuario	Todas las restricciones en columnas y tablas de CREATE TABLE Procedimientos almacenados Triggers

Tabla 3. Categorías de integridad de datos

Integridad de Entidad

La integridad de entidad define una fila como entidad única para una tabla determinada.

La integridad de entidad fuerza la integridad de la columna o columnas de los identificadores o la clave principal de una tabla (mediante índices, restricciones UNIQUE, restricciones PRIMARY KEY o propiedades IDENTITY).

Integridad de Dominio

La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Puede forzar la integridad de dominio si restringe el tipo (mediante tipos de datos), el formato (mediante las reglas RULE y las restricciones CHECK), o el intervalo de valores posibles (mediante restricciones FOREIGN KEY, restricciones CHECK, definiciones DEFAULT, definiciones NOT NULL y reglas).

Integridad Referencial

La integridad referencial protege las relaciones definidas entre las tablas cuando se crean o se eliminan registros.

En Microsoft SQL Server 2000, la integridad referencial se basa en las relaciones entre las llaves foráneas y las llaves primarias o entre las claves externas y las claves únicas.

La integridad referencial garantiza que los valores clave son coherentes en las distintas tablas.

Para conseguir esa coherencia, es preciso que no haya referencias a valores Inexistentes y que, si cambia el valor de una clave, todas las referencias a ella se cambien en consecuencia en toda la base de datos.

Cuando se fuerza la integridad referencial, SQL Server impide a los usuarios:

- Agregar registros a una tabla relacionada si no hay ningún registro asociado en la tabla principal.
- Cambiar valores en una tabla principal de manera que queden registros huérfanos en una tabla relacionada.
- Eliminar registros de una tabla principal cuando hay registros relacionados coincidentes.

Integridad Definida por el Usuario

La integridad definida por el usuario le permite definir reglas de la compañía específicas que no pertenecen a ninguna otra categoría de integridad.

Todas las categorías de integridad son compatibles con la integridad definida por el usuario.

Una de las restricciones que más se utilizaran en tablas es el *CONSTRAINT* que se utiliza con el objeto de mantener la integridad referencial de manera automática (*Primary Key* y *Foreign Key*), así como la integridad de dominio a través de *Defaults* y *Checks*.

También se utiliza para el *Unique* que crea un índice único que no permite que dos filas el mismo valor de índice.

Después de ser creados los campos de la tabla se puede comenzar a hacer los contraints, y se hará de la siguiente manera:

CONSTRAINT /*nombre de la restricción*/ TIPO DE RESTRICCION *(unique, primary key, etc.)*/ (*nombre del campo*)

- Si la restricción es Check después del nombre del campo se coloca una palabra reservada como LIKE y lo que se desea restringir, que pueden ser letras, números, caracteres, etc.
- Si la restricción es Foreign además de lo anterior se sigue con

REFERENCES/*nombre de la tabla a referenciar*/(nombre del campo a referenciar)

Los anteriores son los pasos para crear la restricción mientras se va creando la tabla, pero si ya está creada se tiene que utilizar el ALTER sobre la tabla para hacer las restricciones.

2.2.2. Microsoft Access

Es un sistema de administración de bases de datos relacionales, diseñado especialmente para ser utilizado bajo Windows.

La gama de tareas en las que puede aplicarse Access es prácticamente ilimitada, desde una agenda personal hasta la más compleja base de datos empresarial.

Grandes ventajas pueden obtenerse al utilizar este programa. Por tratarse de un sistema de administración de bases de datos relacionales,

Access puede, por ejemplo, establecer relaciones entre una base de datos con información sobre vendedores, con otra sobre productos vendidos y con una sobre clientes.

Por otra parte, es posible trabajar con una base Access utilizándola en forma independiente o estableciendo vinculaciones con diferentes bases de datos externas, como las realizadas con dBase.

Finalmente, con Access resulta muy fácil introducir datos en la Web, pues pueden crearse páginas para ser conectadas en forma directa a una base de Access o de Microsoft SQL Server.

Microsoft Access maneja el mismo lenguaje que maneja el SQL Server, por lo tanto es la misma tipografía, solo que en Microsoft Access maneja el funcionamiento de manera más gráfica y supuestamente más sencilla, en donde solo en ventanas puedes ir creando de manera eficaz las necesidades que requerirás para tu base de datos.

Además cuenta con un modelo de formularios, los cuales hacen vistoso y fácil el acceso al contenido de la base de datos.

Para explicarse mejor voy a ir mostrando el contenido de Microsoft Access y apoyándome en el punto anterior de SQL Server no repetiré las instrucciones que maneja el lenguaje.

Para cada uno de los objetos que contiene la base de datos Access se pueden generar utilizando la *Vista Diseño*, o sea, diseñar parte por parte, ó utilizando el *Asistente*, que es lo que ayuda Microsoft con Access para reducir tiempo de creación.

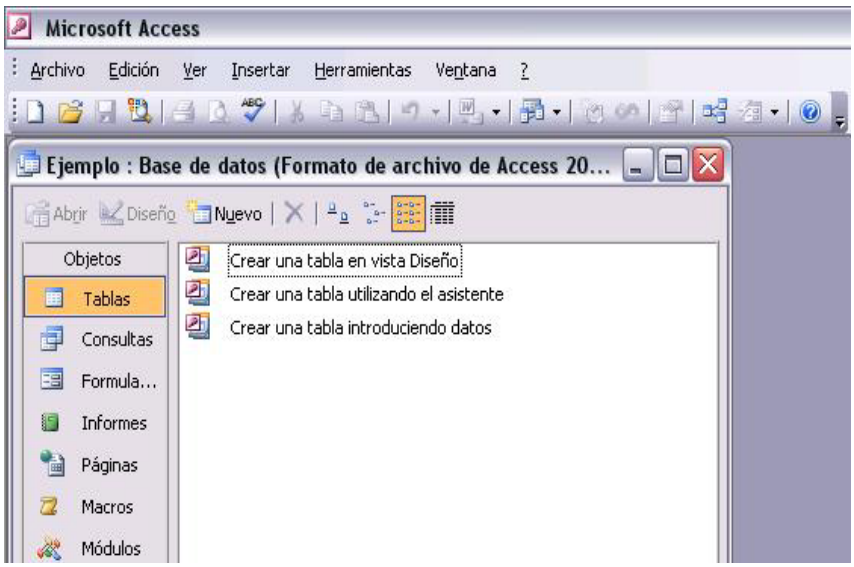


Figura 2. Objetos de Access

2.2.2.1. Tablas

La parte de Tablas contiene 3 opciones para poder crear o manipular una tabla:

1) Crear una tabla en vista Diseño. Diseñas cada uno de los elementos de la tabla, con sus tipos de datos y nombres.
2) Crear una tabla utilizando el asistente. El asistente ayuda con ejemplos a desarrollar la tabla que más se asemeje a lo que requieres.
3) Crear una tabla introduciendo datos. Crea la tabla mientras se van introduciendo los datos.

2.2.2.2. Consultas

Esta parte te permite hacer las consultas de los datos que se encuentran en cada uno de los campos de tus tablas. Esto puedes hacerlo con alguna de las 2 opciones que contiene Access.

1) Crear una consulta en vista Diseño. En esta forma se va a seleccionar la tabla o si existiese otra consulta de dónde tomar los campos.

Los tipos de consultas que se pueden hacer son:

- Consulta de selección. Sirve para encontrar un dato sobre el campo de una tabla o consulta creada.
- Consulta de actualización. Sirve para actualizar el contenido de un campo.
- Consulta de datos anexados. Sirve para traspasar el contenido de los campos de una tabla hacia otra, ya sea de la misma base de datos, o hacia otra.

2) Crear una consulta utilizando el asistente. Con esta opción te facilita si es que quieres hacer una consulta sencilla y evitar meterte muy a fondo. Únicamente tienes que seleccionar de una ventana cual tabla quieres utilizar, cuales campos requieres consultar.

2.2.2.3. Formularios

Los formularios son la forma de manipular en manera de diseño y de forma más fácil los datos de las tablas.

Así como en lenguajes de programación visuales como Visual Basic se pueden crear campos de texto, así de esta manera Access permite que el usuario pueda acceder al contenido de la base de datos de manera visual.

No es muy cómodo utilizar los formularios para fines empresariales porque se tiene que abrir el programa de Access, luego la base de datos, luego ir a formularios y acceder al que se requiera. Más bien es útil para fines personales.

También para crear los formularios hay 2 formas:

1) Crear formulario en vista Diseño. Con esta opción podrás crear tu formulario como en Visual Basic, o sea, te ponen una pantalla que va a ser tu ventana, y sobre ella colocas los elementos que requieras como cajas de texto, botones, imágenes, etc. Cada uno de los elementos contiene sus propiedades.

2) Crear formulario utilizando el asistente.De la misma manera que con las tablas y las consultas, Access te brinda una ventana en la que solo seleccionas las tablas y los campos que vas a requerir que aparezcan, cómo quieres que aparezcan y el diseño de toda la ventana. Es una manera fácil de crear un formulario sencillo.

2.2.2.4. Informes

Así como existen programas que te hacen informes de los datos contenidos en una base de datos como Crystal Report,

Access también contiene su propio manejador de informes. Tal vez no sean tan complejos como los que puedes hacer en Crystal Report pero así como en los formularios, son de gran ayuda si lo requieres para un uso personal.

De igual manera que los anteriores hay 2 formas de crear informes:

1) Crear un informe en vista Diseño. Te da la opción de crear el informe diseñandolo tu mismo en cada una de sus partes.

2) Crear un informe utilizando el asistente. Aparece una ventana como en los demás asistentes en donde elegirás las tablas, campos y cómo irán ordenados para que aparezcan en el informe.

2.2.2.5. Páginas

Access tiene esta opción de crear Páginas Web de manera simple, las cuales al ser terminadas quedarán en tipo HTML o con un tipo de HTML con reconocimiento de Access, que ayuda a poder manipularla de nuevo en su diseño.

Esta opción contiene 3 tipos de manera de ser creada:

1) Crear una página de acceso a datos en vista Diseño. Muestra una ventana parecida a la de otros programas para crear páginas Web, pero con la interacción hacia la base de datos de Access.

2) Crear una página de acceso a datos utilizando el asistente. Otra vez aparece una ventana en donde seleccionas los campos de las tablas o consultas que requieras y sigues los pasos sencillos para la creación de los requerimientos y diseño de la página.

3) Modificar una página Web que ya existe. Seleccionas la página que quieres modificar de la ruta en donde está guardada y la modificas como en la de Vista Diseño

2.2.2.6. Macros

Los macros son un conjunto de acciones que pueden ser utilizadas en la base de datos, ya sea crear abrir formularios, mostrar un cuadro de texto, etc., eso depende de lo que quieras hacer ya que los Macros pueden manipular todas las acciones de la base de datos.

2.2.2.7. Módulos

Un módulo es un conjunto de instrucciones que se encuentran en un programa.

Hay instrucciones que son locales y públicas. Las locales sólo funcionan en cierta parte del programa y las públicas pueden ser ejecutadas sobre todo el programa. Pues el módulo actúa como instrucciones públicas y sirven para no estar repitiendo éstas en cada parte del programa.

Para utilizar los módulos por consecuencia tienes que tener un lenguaje de programación, y el lenguaje que utiliza Microsoft Access es el Microsoft Visual Basic y es donde se tendrán que programar las instrucciones que debe de llevar el módulo.

2.3 CREACIÓN DE LA BASE DE DATOS EN MICROSOFT ACCESS 2003

Ya conociendo qué es lo que puedes hacer con Access se puede iniciar la creación de la base de datos ya lo demás depende de los requerimientos de la empresa.

Pondré un ejemplo de una base de datos de un negocio de Ventas.

Para mejor uso de las restricciones las tablas las crearé en modo Diseño.

La empresa debe clasificar los productos que va a vender, entonces se creará una tabla de ***Clasificación***.

	Nombre del campo	Tipo de datos
🔑	idClasif	Autonumérico
	NomClasif	Texto

Como se muestra, va a contener 2 campos, los id para esta tabla y las demás que vayamos a crear hacen referencia al número de control y el Nom se refiere al nombre.

La llave que muestra al lado izquierdo hace notar que el campo contiene una referencia *Primary key*, para colocarla solo se tienen que seleccionar los campos que se va a hacer este tipo de referencia y en los iconos del menú principal aparece una llave como la que esta en la figura, eso quiere decir que no se puede duplicar el número de control, y además en el tipo de dato tiene un autonumérico para que en cada creación de un dato vaya aumentando automáticamente el número de control.

Ahora se creará la tabla de ***Producto***.

	Nombre del campo	Tipo de datos
🔑	idProducto	Autonumérico
	NomProducto	Texto
	Existencia	Número
	Precio	Texto
	idClasif	Número

Estos son los campos que contendrá, el campo de idClasif que aparece en la de Productos es porque se va a hacer una referencia

de tipo *Foreign* para que los productos tengan una clasificación única.

Para hacer la relación, en la ventana principal de la base de datos en el menú principal en la parte de Herramientas aparece una opción que se llama *Relaciones*.

Aparecerá una ventana en donde se seleccionarás las tablas existentes para poder verlas y manipular las relaciones. En este caso seleccionaré las 2 tablas existentes.

Ya que aparecen las tablas en pantalla sólo basta con seleccionar con el ratón el campo idClasif de la tabla Clasificación (el campo que está en negritas), que así es como se muestra cuando tiene *Primary key*, y sin soltar el clic del ratón se traspasa hacia el otro campo idClasif de la tabla Productos y se suelta el botón del ratón.

Al hacer esto se abrirá una ventana en la cual mostrará la relación entre los campos, ahí se le tiene que “exigir integridad referencial” para que sirva como lo había explicado.

Quedará entonces de la siguiente manera:

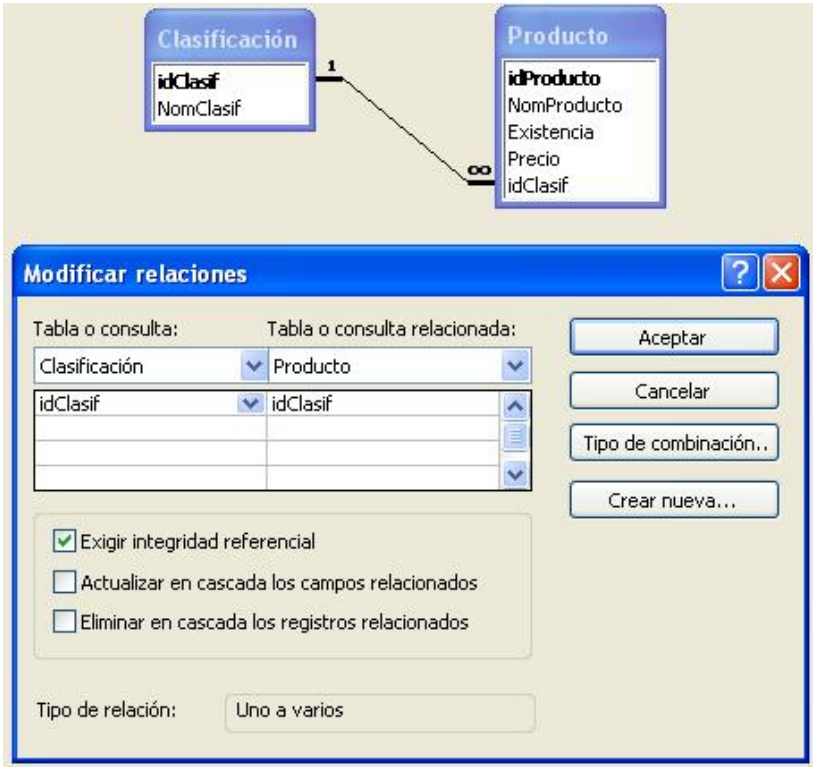


Figura 3. Relación de las tablas Clasificación y Producto

De la misma manera se creará para la tabla de ***ClasSer***, ***Servicio***, ***Pais***, ***Provincia***, ***Ciudad***, ***Usuario*** y ***CompraVirtual***.

	Nombre del campo	Tipo de datos
🔑	idClasSer	Autonumérico
	NomClasSer	Texto

	Nombre del campo	Tipo de datos
🔑	idServicio	Autonumérico
	NomServicio	Texto
	Precio	Número
	idClasSer	Número

Las tablas ClasSer y Servicio son similares a las tablas de Clasificación y de Producto. Así que tendrán los mismos tipos de relaciones. Estas tablas identificarán la clasificación del servicio y el servicio que le corresponde a cada clasificación

Las relaciones quedarán de la siguiente manera:

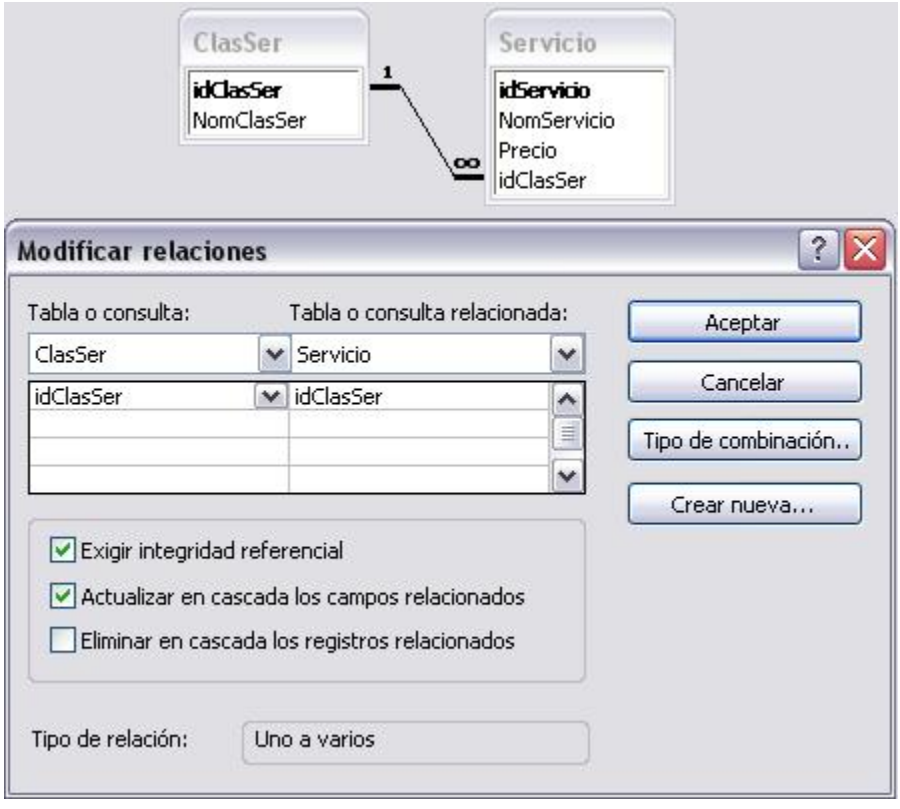


Figura 4. Relación de las tablas ClasSer y Servicio

	Nombre del campo	Tipo de datos
🔑	idPais	Autonumérico
	NomPais	Texto

	Nombre del campo	Tipo de datos
🔑	idProvincia	Autonumérico
	NomProvincia	Texto
	idPais	Número

	Nombre del campo	Tipo de datos
🔑	idCiudad	Autonumérico
	NomCiudad	Texto
	idProvincia	Número

Las tablas **Pais**, **Provincia** y **Ciudad** servirán para tener almacenado los países, las provincias de estos países y sus ciudades de las respectivas provincias. De igual manera las tablas Pais, Provincia y Ciudad irán relacionadas.

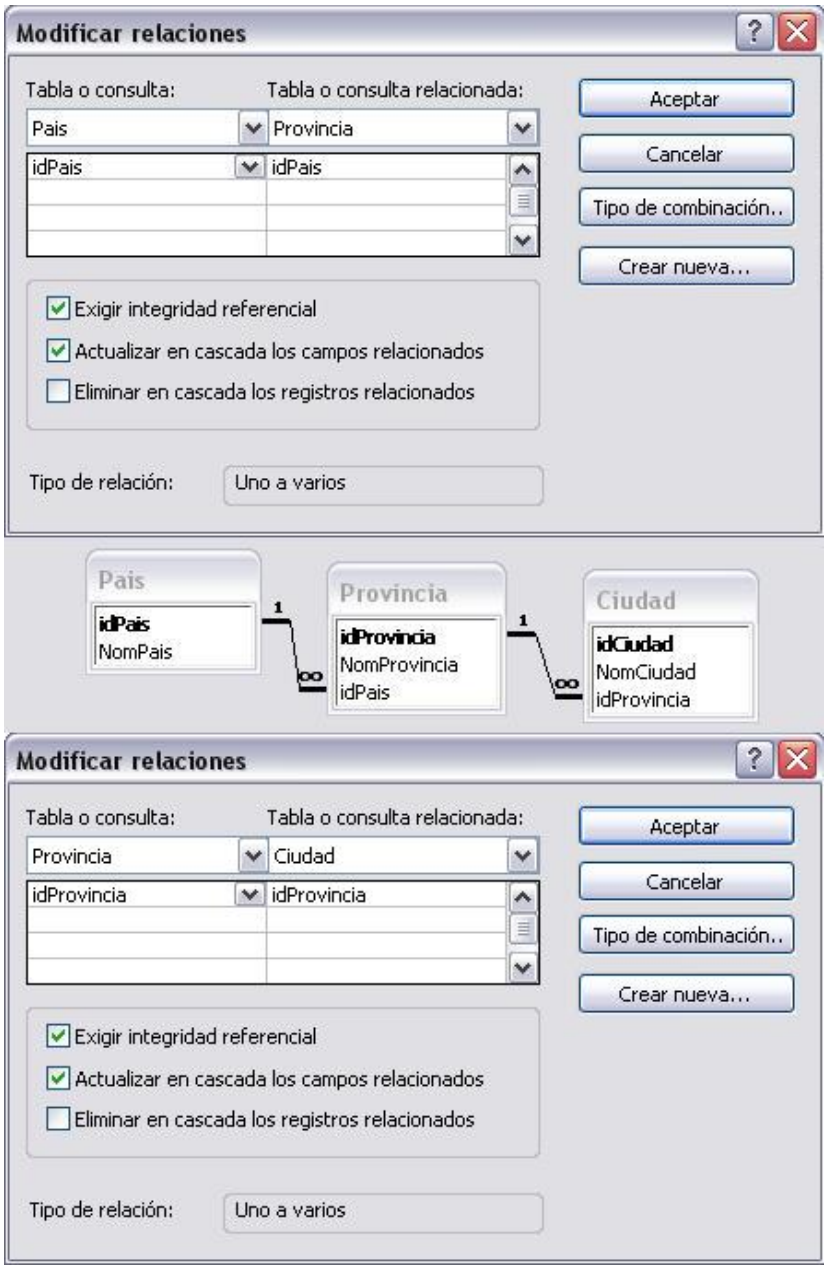


Figura 5. Relación de las tablas Pais, Provincia y Ciudad

	Nombre del campo	Tipo de datos
🔑	idUsuario	Autonumérico
	NomUsuario	Texto
	Apellidos	Texto
	Nick	Texto
	Password	Texto
	Nacimiento	Fecha/Hora
	Sexo	Texto
	Pais	Texto
	Provincia	Texto
	Ciudad	Texto
	Ocupacion	Texto
	Sector	Texto
	Educacion	Texto

La tabla **Usuario** nos servirá para almacenar los datos de una persona, que será un usuario registrado para fines correspondientes a la página.

Esta tabla no esta relacionada con ninguna.

	Nombre del campo	Tipo de datos
🔑	idCompra	Autonumérico
	NomUsuario	Texto
	NomClasificacion	Texto
	idProducto	Número
	NomProducto	Texto
	Cantidad	Número
	Fecha	Fecha/Hora

La tabla **CompraVirtual** nos servirá para almacenar los datos de una compra, que llevará la clasificación del producto, su nombre, la cantidad de producto que se va a comprar, y la fecha en que se compra.

Esta tabla tampoco estará relacionada con ninguna.

Quedarán las tablas de la siguiente manera:

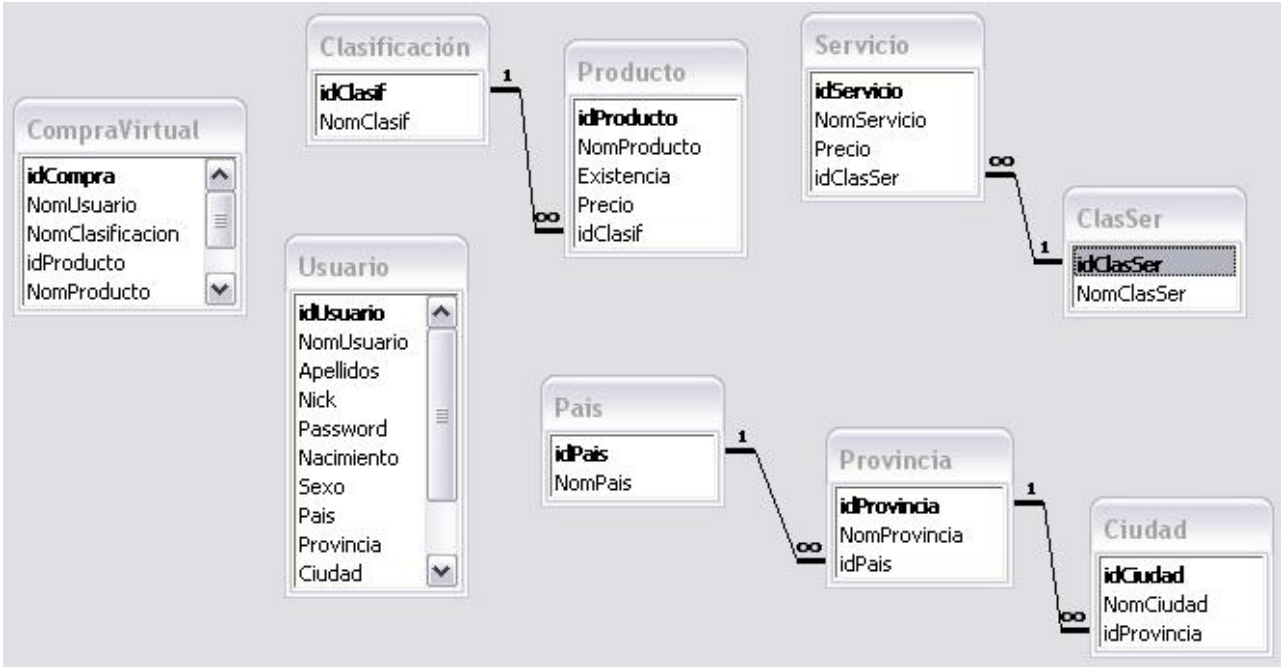


Figura 6. Diagrama de tablas de la base de datos

2.4 CREACIÓN DE LA BASE DE DATOS EN MICROSOFT SQL SERVER 2000

Para crear la base de datos en SQL se tendrá que abrir el “Administrador corporativo” de SQL, ahí se seleccionará en la parte de Base de datos como se muestra en la imagen, con el botón derecho aparecerá la opción de crear una nueva Base de datos.

Figura 7. Administrador corporativo de SQL

Ya creada la base de datos, se debe acceder al “Analizador de consultas” que es donde se van a crear las tablas. Este se abre desde el menú del Administrador Corporativo, en la parte de Herramientas.

Ya estando en el Analizador de Consultas se selecciona de la barra de herramientas del menú principal la base de datos en la que se van a crear las tablas.

Entonces se comenzará escribiendo en la parte blanca del lado derecho de la ventana la creación de las tablas de la siguiente manera, tomando en cuenta las tablas que se pusieron como ejemplo en Access:

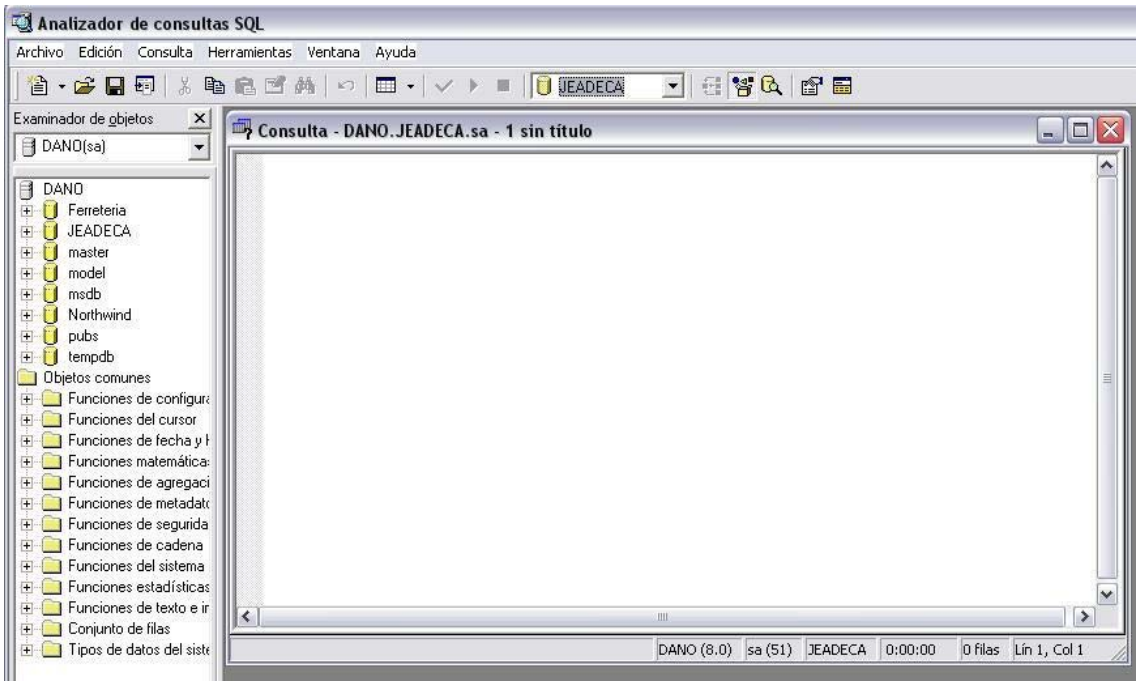


Figura 8. Area de trabajo del Analizador de consultas de SQL

```
Create Table Clasificación (
        idClasif integer not null primary key,
        NomClasif varchar(30)
)
Create Table Producto (
        idProducto integer not null primary key,
        NomProducto varchar(30),
        Existencia integer,
        Precio numeric(10,2),
        idClasif integer,
        Constraint prod1 foreign key (idClasif) References Clasificación
        (idClasif)
)
Create Table ClasSer (
        idClasSer integer not null primary key,
        NomClasSer varchar(20)
)
```

```
Create Table Servicio (
        idServicio integer not null primary key,
        NomServicio varchar(50),
        Precio numeric(10,2),
        idClasSer integer,
        Constraint serv1 foreign key (idClasSer) References ClasSer
        (idClasSer)
)

Create Table Pais (
        idPais integer not null primary key,
        NomPais varchar (20)
)

Create Table Provincia (
        idProvincia integer not null primary key,
        NomProvincia varchar(20),
        idPais integer,
        Constraint provin1 foreign key (idPais) References Pais (idPais)
)

Create Table Ciudad (
        idCiudad integer not null primary key,
        NomCiudad varchar(30),
        idProvincia integer,
        Constraint ciud1 foreign key (idProvincia) References Provincia
        (idProvincia)
)
```

```
Create Table Usuario (
        idUsuario integer not null primary key,
        NomUsuario varchar(15),
        Apellidos varchar(50),
        Nick varchar(15),
        Password varchar(10)
        Nacimiento smalldatetime,
        Sexo varchar(8),
        Pais varchar(20),
        Provincia varchar(20),
        Ciudad varchar(30),
        Ocupacion varchar(30),
        Sector varchar(40),
        Educación varchar(30)
)


Create Table CompraVirtual (
        idCompra integer not null primary key,
        NomUsuario varchar(15),
        NomClasificacion varchar(30),
        idProducto integer,
        NomProducto varchar(30),
        Cantidad integer,
        Fecha smalldatetime
)
```

2.5 LENGUAJES DE PROGRAMACIÓN Y SOFTWARE CORRESPONDIENTES

Para poder implementar la posibilidad de acceder desde la página Web a la base de datos existen lenguajes que interactúan entre la parte del diseño visual y el de la base de datos, como son PHP, ASP y JSP.

ASP y JSP son muy similares, la diferencia entre estos son las plataformas en que trabajan. ASP esta limitada para arquitecturas basadas en la tecnología Microsoft, y JSP sigue la filosofía de la arquitectura JAVA.

Entonces para poder utilizar los lenguajes de manera en que el programador elija entre una y otra plataforma se tomará en cuenta PHP y ASP.

2.5.1. Servidor Apache

Un servidor web es un programa que implementa el protocolo HTTP (hypertext transfer protocol). Este protocolo está diseñado para transferir lo que llamamos hipertextos, páginas Web o páginas HTML (hypertext markup language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos[3].

Las páginas que contienen exclusivamente código HTML se pueden desarrollar y probar sin la intervención de un servidor Web, ya que el código HTML es interpretado en el navegador del usuario. PHP, como otros lenguajes, se ejecuta en un servidor antes de que la página sea enviada al usuario que realizó la petición.

Cuando llega una petición a un servidor Web, éste localiza el documento solicitado por el cliente y, en función de una serie de parámetros de la

[3] En http://es.wikipedia.org/wiki/Servidor_web

propia configuración del servidor, decide la acción a realizar con el documento.

Para explicar mejor esta interactividad entre usuario-servidor podemos comparar cómo se obtienen un documento estático y uno dinámico (que es la base de nuestra explicación total).

OBTENCIÓN DE UN DOCUMENTO ESTÁTICO

Figura 9. Diagrama de petición de una página estática

1. El usuario pulsa sobre un link solicitando un documento (un archivo HTML, una imagen en formato jpg, etc.) y el navegador envía la petición al servidor utilizando el protocolo HTTP (protocolo de transferencia de hipertexto),
2. La solicitud llega hasta el servidor Web correspondiente a través de la red. El servidor localiza el documento solicitado.
3. El servidor lee el documento del sistema de archivos y envía al cliente una copia exacta del mismo.
4. El documento llega al cliente y se visualiza su contenido en el navegador del usuario.

OBTENCIÓN DE UN DOCUMENTO DINÁMICO

Figura 10. Diagrama de petición de una página dinámica

1. El usuario pulsa sobre un link solicitando un documento (un archivo .phtml o .php) y el navegador envía la petición al servidor utilizando el protocolo http (protocolo de transferencia de hipertexto).
2. Llega la solicitud al servidor y localiza el documento. Por la extensión del nombre del archivo determina que se trata de un archivo que contiene código PHP y lanza el intérprete.
3. El intérprete ejecuta el script solicitado y genera un resultado (habitualmente una página HTML) que se devuelve al servidor para que éste a su vez lo transfiera al cliente.
4. Se visualiza el documento en el navegador del usuario.

A una solicitud del usuario el proceso es exactamente el mismo: pulsa sobre un link y recibe la información requerida. Sin embargo, en el servidor el proceso realizado antes de enviar la información ha sido diferente, ya que en el segundo caso ha sido necesaria la intervención del intérprete de PHP para elaborar dinámicamente el contenido.

El servidor Apache es un software que esta estructurado en módulos. La configuración de cada módulo se hace mediante la configuración de las

directivas que están contenidas dentro del módulo. Los módulos del Apache se pueden clasificar en tres categorías:

- **Módulos Base:** Módulo con las funciones básicas del Apache.
- **Módulos Multiproceso:** son los responsables de la unión con los puertos de la máquina, acepando las peticiones y enviando a los hijos a atender a las peticiones.
- **Módulos Adicionales:** Cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y rapidez del código.

El resto de funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma que no es necesario volver a instalar el software.[4]

Entonces se podrá decir que un servidor Apache es totalmente efectivo para la interacción entre nuestra página Web con algunos lenguajes (específicamente PHP) y el cliente con su petición de nuestra página (específicamente los archivos y módulos a tratar).

2.5.2. PHP

PHP, (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje "Open Source" interpretado de alto nivel. Especialmente pensado para desarrollos Web y el cual puede ser inmerso en páginas HTML. La mayoría de su sintaxis es similar a C, Java, Perl y es fácil de aprender. El objetivo

[4] En http://www.desarrolloweb.com/articulos

de este lenguaje es, permitir a los creadores de páginas Web, escribir páginas dinámicas de una manera rápida y fácil.[5]

PHP puede procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Características y entorno de PHP

- Es software libre, lo que implica menores costos y servidores más baratos que otras alternativas, a la vez que el tiempo entre el hallazgo de un fallo y su resolución es más corto. Además, el volumen de código PHP libre es mucho mayor que en otras tecnologías, siendo superado por Perl, que es más antiguo. Esto permite construir sitios realmente interesantes con sólo instalar scripts libres como el conocido PHP Nuke.
- Es muy rápido. Su integración con la base de datos MySQL y SQL Server, también veloz, le permite constituirse como una de las alternativas más atractivas para sitios de tamaño medio-bajo.
- Su sintaxis está inspirada en C, ligeramente modificada para adaptarlo al entorno en el que trabaja, de modo que si estás familiarizado con esa sintaxis, PHP o JSP son las opciones más atractivas.
- Su librería estándar es realmente amplia, lo que permite reducir los llamados "costos ocultos", uno de los principales defectos de ASP.
- PHP es relativamente multiplataforma. Funciona en toda máquina que sea capaz de compilar su código, entre ellas diversos sistemas operativos para PC y diversos Unix. El código escrito en PHP en cualquier plataforma funciona exactamente igual en cualquier otra.
- El acceso a las bases de datos de PHP es muy heterogéneo, pues dispone de un juego de funciones distinto por cada gestor.

[5] En http://www.cursodephp.com/

- PHP es suficientemente versátil y potente como para hacer tanto aplicaciones grandes que necesiten acceder a recursos a bajo nivel del sistema como pequeños scripts que envíen por correo electrónico un formulario rellenado por el usuario.
- Existen menos especialistas en PHP que en ASP en nuestro país.
- Como lenguaje, PHP 4 padece ciertas carencias: no soporta polimorfismo ni tiene excepciones u otro sistema de errores aceptable, cosas que se solucionan en la nueva versión PHP 5.
- PHP tiene una de las comunidades más grandes en Internet, con lo que no es complicado encontrar ayuda, documentación, artículos, noticias, y más recursos.
- PHP esta revolucionando Internet a la vez que evoluciona a pasos gigantescos.

¿Qué se puede hacer con PHP?

- Generación de contenidos multimedia dinámicos (gráficos, flash, pdf): podemos añadir un grado más allá de interactividad y vistosidad a nuestra web, mediante la generación de dichos contenidos multimedia dependiendo de los datos enviados por un formulario, información guardada en una base de datos, etc.
- Conexión de una web a otros servicios Internet (FTP, POP3, IMAP, LDAP, SMTP,...): PHP puede hacer las veces de interfaz web agradable al usuario para facilitar el acceso a determinados servicios Internet. Sirvan como ejemplo los muchos proveedores de correo web, que permiten interactuar con un servicio de correo a través de una web.
- Generación de contenidos para dispositivos móviles: : gracias a HAWHAW (servicio de Aplicaciones WEB), podemos generar contenidos adecuados a los últimos dispositivos como PDA y teléfonos móviles, sin preocuparnos de la implementación concreta del protocolo utilizado por cada uno de ellos.

- Interactuar con XML: cada vez más datos se almacenan en XML. Y PHP puede controlar cada aspecto relacionado con esta tecnología, desde su modificación y su transformación (en HTML, por ejemplo), hasta su análisis sintáctico y la búsqueda de información concreta. Además, PHP nos permite generar con facilidad cualquier formato de contenido basado en XML, como gráficos vectoriales SVG o canales de contenido sindicado RSS (para mantener a nuestra audiencia constantemente informada de nuestras actualizaciones).
- Desarrollar servicios web: los servicios web permiten llevar el intercambio de información entre servidores un paso más allá en cuanto a facilidad de implementación. Los protocolos XML-RPC y SOAP, ambos basados en XML, permiten utilizar el protocolo http para intercambiar información de muy diversos tipos. Como ejemplo, GOOGLE web APIs (basado en SOAP), permite utilizar el motor de búsqueda de Google.com en nuestra página web, y modificar los resultados obtenidos en XML de cualquier forma.
- Crear scripts de línea de comandos o aplicaciones GUI portables: PHP también puede funcionar desde la línea de comandos, por lo que podría ser su substituto de Perl o Phyton para tareas administrativas y de automatización. Y además, mediante su interacción con la librería GTK, permite crear programas GUI (interfaz gráfica de usuario, como los programas de Visual Basic o Delphi). Dichos programas creados con PHP-GTK son altamente portables entre distintas plataformas.

Aparte de todo esto, nuestros desarrollos en PHP pueden perfeccionarse y facilitarse teniendo en cuenta otros aspectos como:

- Utilización de metodologías con arquitecturas multinivel con MVC (Modelo-Vista-Controlador): esta metodología permite separar la presentación de la lógica, facilitando el mantenimiento y desarrollo de cualquier proyecto. Para ello se pueden utilizar los sistemas de

plantillas (como Smarty, el más avanzado y potente), y librerías de código (como PEAR, un gran repositorio de clases libres)

- Seguridad: la poca confianza de los usuarios en la venta on-line y el manejo de datos sensibles debe ser eliminada mediante métodos de fortificación de las diferentes aplicaciones que participan en nuestros desarrollos web y la utilización de una programación segura. La encriptación de datos y la habilitación de conexiones seguras por SSL también son aspectos a tener en cuenta.
- Monitorización y optimización: la vigilancia de los parámetros de rendimiento nos puede llevar a conseguir aplicaciones web más rápidas y estables. Como complemento, existen cachés de compilación de PHP que pueden acelerar algunos scripts PHP en más de un 55%.

Todas estos aspectos nos permitirían generar, por ejemplo, una tienda virtual con aspectos avanzados como generación dinámica de facturas en pdf, contenidos para dispositivos móviles, atractivas presentaciones dinámicas en flash, datos almacenados en XML, control mediante una aplicación GUI, canales de información a los usuarios, y además segura, eficiente y fácil de actualizar y desarrollar.

PHP está muy relacionado con el lenguaje de hipertextos HTML; tanto es así que el código PHP aparece normalmente insertado dentro de un documento HTML. El documento PHP, una vez interpretado correctamente en el servidor genera una página HTML y para diferenciar ambos lenguajes dentro de un mismo documento se utilizan etiquetas de comienzo y final de código PHP. Las etiquetas mas habituales para delimitar los bloque de código PHP son las siguientes:

```
<?php
echo "Esto es una instrucción PHP";
?>
```

O solamente colocar:

```
<?
echo "Esto es una instrucción PHP";
?>
```

La extensión de los archivos en PHP es muy importante, ya que estas muestran si el servidor va a procesar el documento por el intérprete PHP, y las extensiones que contienen códigos PHP son:

- .php3 (para versión PHP 3.x)
- .php4 (para versión PHP 4.x)
- .php (esta es la que se utiliza para guardar los programas en PHP)
- .phps (para ver la sintaxis del código resaltado en colores)

Comentarios

Los comentarios son parte del programa que se utiliza para poner explicaciones sobre parte del código, documentando los pasos que se están haciendo o explicando cierta parte de código.

PHP ofrece la posibilidad de insertar comentarios de tres formas distintas. La primera consiste en emplear dos caracteres / seguidos (//), para comentar una única línea de código. La segunda es utilizando el carácter almohadilla “#”, también para comentar una única línea de código, con la diferencia que el comentario termina con el retorno de carro de la línea comentada o con el símbolo de finalización de interpretación de PHP.
La tercer tipo de comentario es de tipo multilínea, es decir, nos permite comentar varias líneas de código fuente, de tal forma que el comienzo se indica con la secuencia de caracteres /* y el final con */.

Variables

La forma principal de almacenar valores en el medio de un programa son las variables. Los puntos más importantes a recordar son:

- Todas las variables en PHP comienzan con el símbolo de pesos $, seguido, al menos, por una letra o un guión bajo (“_”) para después continuar por cualquier combinación de letras, de dígitos y de guiones bajos.
- Las variables no necesitan ser declaradas antes de ser usadas, se crean en el instante en que son utilizadas por primera vez.
- El valor de una variable es igual al valor más recientemente asignado.
- Las variables son asignadas e inicializadas con el operador de asignación '=', con la variable a la izquierda del operador y la expresión a evaluar a la derecha.
- Las variables no tienen un valor intrínseco, sino que toman el tipo del último valor asignado.
- Las variables que se usan antes de ser asignadas tienen un valor por defecto.
- El nombre de la variable es sensible a minúsculas y mayúsculas. Por ejemplo $pagina y $Página son variables distintas.

Existen variables predefinidas por PHP.

La siguiente tabla muestra alguna de las variables que PHP ofrece al programador para facilitar su tarea.

Variable	Significado
argv	Array de argumentos pasados al script. Cuando el script se ejecuta desde la línea de comandos, esto da un acceso, al estilo de C, a los parámetros pasados en línea de comandos. Cuando se le llama mediante el método GET, contendrá la cadena de la petición. Requiere que esté activada la directiva register_argc_argv en el archivo de inicialización.
argc	Contiene el número de parámetros de la línea de comandos pasados al script (si se ejecuta desde la línea de comandos). Requiere que esté activada la directiva register_argc_argv
PHP_SELF	El nombre del archivo que contiene el script que se esta ejecutando, relativo al directorio raíz de los documentos. Si PHP se está ejecutando como intérprete de línea de comandos, esta variable no está disponible.
_COOKIE	Un array asociativo de variables (clave, valor) pasadas al script actual mediante cookies HTTP. Sólo está disponible si el seguimiento de variables ha sido activado mediante la directiva de configuración track_vars o la directiva<?php_track_vars?>.
_GET	Un array asociativo de variables (clave, valor) pasadas al script actual mediante el método HTTP GET. Sólo está disponible si --variable tracking-- ha sido activado mediante la directiva de configuración track_vars o la directiva <?php_track_vars?>.
_POST	Un array asociativo de variables (clave, valor) pasadas al script actual mediante el método HTTP POST. Sólo está disponible si –variable tracking—ha sido activado mediante la directiva de configuración track_vars o la directiva <?php_track_vars?>.
_POST_FILES	Array asociativo que contiene información de los archivos recibidos usando el método POST.
_ENV_VARS	Array asociativo de pares (clave, valor) del entorno.
_SERVER_VARS	Array asociativo de pares (clave, valor) del servidor
_SESSION_VARS	Array asociativo de pares (clave, valor) de sesión

Tabla 4. Variables predefinidas de PHP

Tipo de datos

PHP soporta los siguientes tipos:

- Entero
- Números en punto flotante
- Booleano
- Cadena

• Array
• Objeto

El tipo de una variable normalmente no lo indica el programador; en su lugar, lo decide PHP en tiempo de ejecución, dependiendo del contexto en el que se utilice esa variable.

Además existen funciones que PHP contiene como las de fecha y hora. Estas son las que proporciona PHP:

Función	Descripción
time()	Obtiene la marca de tiempo UNIX actual
checkdate()	Valida una fecha en formato gregoriano
date()	Da formato a la hora y fecha locales
getdate()	Obtiene información sobre la fecha y la hora locales
gettimeofday()	Obtiene la hora actual
gmdate()	Formatea la fecha y la hora a formato GMT
gmmktime()	Obtiene la marca de tiempo UNIX de una fecha con formato GMT
gmstrftime()	Formatea la fecha y la hora con formato GMT a las características locales
microtime()	Obtiene la marca de tiempo UNIX actual en microsegundos
mktime()	Obtiene la marca de tiempo UNIX para una fecha dada
strftime()	Formatear la hora actual de acuerdo con las características locales
strtotime()	Traduce una fecha u hora descritas en inglés a su correspondiente marca de tiempo UNIX

Tabla 5. Funciones de fecha y hora en PHP

Esta función nos permite darle un formato específico a una cadena que contendrá una fecha y una hora. Acepta como parámetro una cadena de formato y un parámetro timestamp; si éste se omite, se tomará el instante de ejecución de la orden.

Para **date(formato [,timestamp])**:

Valores	Descripción
a	"a.m." o "p.m."
A	"A.M." o "P.M."
d	Día del mes con dos dígitos y con cero a la izquierda, de "01" a "31"
D	Día de la semana con tres caracteres, "Fri"
F	Nombre del mes, en texto completo, "January"
h	Hora en formato, de "01" a "12"
H	Hora en formato, de "00" a "23"
g	Hora en formato, de "1" a "12" (sin ceros)
G	Hora en formato; de "0" a "23" (sin ceros)
i	Minutos, de "00" a "59"
j	Día del mes en formato "1" a "31"
l	Día de la semana, en texto completo, "Friday"
L	"1" or "0", 1 si es año bisiesto y 0 si no lo es
m	Mes de "01" a "12"
M	Mes con tres caracteres, "Jan"
n	Mes de "1" a "12" (sin cero)
s	Segundos de "00" a "59"
S	Sufijo ordinal en inglés, "th", "nd"
t	Número de días del mes dado, de "28"a "31"
U	Segundos desde el valor de inicio 'epoch' (01-01-1970)
w	Día de la semana de "0" (domingo) a "6" (sábado)
Y	Año con cuatro dígitos, "2004"
y	Año con dos dígitos, "04"
z	Día del año de "0" a "365"
Z	Diferencia horaria en segundos de "-43200" a "43200"

Tabla 6. Formato para la fecha y hora con la función Date

Para **getdate ([timestamp])**:

Clave	Descripción
seconds	Identifica los segundos
minutes	Identifica los minutos
hours	Identifica las horas
mday	Identifica el día del mes
wday	Identifica, en número, el día de la semana
mon	Identifica, en número, el mes
year	Identifica, en número, el año
yday	Identifica, en número, el día del año
weekday	Identifica, en texto, el día de la semana
month	Identifica, en texto, el mes

Tabla 7. Formato para la fecha y hora con la función Getdate

Formularios

Cuando se envía un formulario para su procesamiento, se produce el emparejamiento de sus controles, a través de sus respectivos nombres, con los valores actuales que tienen. Estas parejas variable-valor configuran el conjunto de datos del formulario que se envían al agente servidor.

PHP, a través de un conjunto de variables globales, es capaz de recuperar el conjunto de datos del formulario que han sido enviados desde el cliente (esto es, el navegador) para, después, poder trabajar con ellos. Las tres variables principales para realizar esta operación son:

Variable	Contenido
_POST	Array que contiene las variables pasadas a través del método POST su uso es análogo al array
_GET	Array que contiene las variables pasadas a través del método GET su uso es análogo al array
_REQUEST	Array que contiene las variables pasadas a través de cualquier mecanismo de entrada

Tabla 8. Variables de paso por formulario

2.5.3. ASP.NET

ASP.NET es un marco de trabajo de programación generado en Common Language Runtime que puede utilizarse en un servidor para generar eficaces aplicaciones Web.

ASP .NET es la nueva tecnología propuesta por Microsoft para enfrentar los desafíos de interconexión entre dispositivos y sitios Web del nuevo milenio. En versiones anteriores de Visual Basic, las opciones brindadas para la programación de estas últimas no contaba con las ventajas que ofrecían otras herramientas, como visual InterDev (ambiente poderoso de desarrollo para crear aplicaciones Web manejadas en base de datos).

En esta versión, ahora se cuenta con un excelente editor de páginas para servidor activo ASP .NET y HTML, el cual viene incluido por el entorno de desarrollo como diseñador natural. Adicionalmente se han agregado varias características, a los efectos de que la programación de una aplicación para la Web cuente con todas las funcionalidades del lenguaje, y además se efectúe en forma similar a como se hacía bajo el modelo Windows.

Las técnicas brindadas para dibujar un formulario de Windows no difieren en mucho de las ofrecidas para realizar la misma tarea en una página Web. Ahora se cuenta con un nuevo tipo de proyecto llamado aplicación Web ASP .NET, el cual hace uso de formularios Web o Web Forms, los cuales cumplen un rol similar al de los formularios estándares en el modelo Windows. Básicamente un formulario Web es exhibido de igual forma que uno Windows, pero, en realidad, el primero es gestionado internamente como una página de servidor activo y etiquetas HTML.

La creación de la interfaz es muy sencilla, ya que basta con crear un nuevo proyecto de este tipo, y posteriormente arrastrar los controles del cuadro de herramientas, para así crear una nueva interfaz gráfica para la Web.

ASP.Net ofrece algunas ventajas que destacan de los demás modelos de programación de Web como:

- **Mejor rendimiento.** ASP.NET es un código de Common Language Runtime compilado que se ejecuta en el servidor. A diferencia de sus predecesores, ASP.NET puede aprovechar las ventajas del enlace anticipado, la compilación just-in-time, la optimización nativa y los servicios de caché desde el primer momento. Esto supone un incremento espectacular del rendimiento antes de siquiera escribir una línea de código.
- **Compatibilidad con herramientas de primer nivel.** El marco de trabajo de ASP.NET se complementa con un diseñador y una caja de herramientas muy completos en el entorno integrado de programación (Integrated Development Environment, IDE) de Visual Studio. La edición de los controles de servidor de arrastrar y colocar y la implementación automática son sólo algunas de las características que proporciona esta eficaz herramienta.
- **Eficacia y flexibilidad.** La biblioteca de clases de .NET Framework, la Mensajería y las soluciones de Acceso a datos se encuentran accesibles desde el Web de manera uniforme. ASP.NET es también independiente del lenguaje, por lo que puede elegir el lenguaje que mejor se adapte a la aplicación o dividir la aplicación en varios lenguajes. Además, la interoperabilidad de Common Language Runtime garantiza que la inversión existente en programación basada en COM se conserva al migrar a ASP.NET.
- **Simplicidad.** ASP.NET facilita la realización de tareas comunes, desde el sencillo envío de formularios y la autenticación del cliente hasta la implementación y la configuración de sitios. Por ejemplo, el marco de trabajo de página de ASP.NET permite generar interfaces de usuario, que separan claramente la lógica de aplicación del código de presentación, y controlar eventos en un sencillo modelo de procesamiento de formularios de tipo Visual Basic.
- **Facilidad de uso.** ASP.NET emplea un sistema de configuración jerárquico, basado en texto, que simplifica la aplicación de la

configuración al entorno de servidor y las aplicaciones Web. Debido a que la información de configuración se almacena como texto sin formato, se puede aplicar la nueva configuración sin la ayuda de herramientas de administración local. Esta filosofía de "administración local cero" se extiende asimismo a la implementación de las aplicaciones ASP.NET Framework. Una aplicación ASP.NET Framework se implementa en un servidor sencillamente mediante la copia de los archivos necesarios al servidor. No se requiere el reinicio del servidor, ni siquiera para implementar o reemplazar el código compilado en ejecución.

- **Escalabilidad y disponibilidad.** ASP.NET se ha diseñado teniendo en cuenta la escalabilidad, con características diseñadas específicamente a medida, con el fin de mejorar el rendimiento en entornos agrupados y de múltiples procesadores. Además, el motor de tiempo de ejecución de ASP.NET controla y administra los procesos de cerca, por lo que si uno no se comporta adecuadamente (filtraciones, bloqueos), se puede crear un proceso nuevo en su lugar, lo que ayuda a mantener la aplicación disponible constantemente para controlar solicitudes.
- **Posibilidad de personalización y extensibilidad.** ASP.NET presenta una arquitectura bien diseñada que permite a los programadores insertar su código en el nivel adecuado. De hecho, es posible extender o reemplazar cualquier subcomponente del motor de tiempo de ejecución de ASP.NET con su propio componente escrito personalizado.
- **Seguridad.** Con la autenticación de Windows integrada y la configuración por aplicación, se puede tener la completa seguridad de que las aplicaciones están a salvo.

Como ya fue mencionado, ASP.Net trabaja sobre Web Forms, en las cuales se hace un diseño de todo tipo de lenguaje, pero para dar una facilidad de programación y diseño lo más conveniente es utilizar con Visual.Net.

Para poder lograr esta versatilidad ASP.Net pueden emplearse 2 archivos diferentes, uno para la presentación y otro para el código de los eventos desarrollados, esto permite que las 2 partes puedan ser tratadas en forma independiente. Las extensiones son las siguientes:

- .aspx (contiene la información de la interfaz)
- .aspx.vb (contiene el código)

Las aplicaciones ASP .NET crean la interfaz en el servidor a través de una pagina, la cual posteriormente es enviada al explorador o cliente a través del protocolo http, y empleando como formato el archivo html.

Propiedades, métodos y eventos de páginas Web

Desde el punto de vista de ASP .NET una página es un objeto que gana o hereda sus funcionalidades de una clase llamada ***Page***, la cual sirve como contenedor para otro controles. La idea es la de proveer mediante este objeto un modelo similar al propuesto por Windows.

Toda página es entonces un objeto derivado de *Page*, el cual cuenta con varios eventos, propiedades y métodos. Los eventos más comunes utilizados son: INIT, PreRender, Load y Unload. Dos de ellos son familiares (Load y UnLoad) a los desarrolladores de Visual Basic y, de hecho, tienen cierto parecido. Cuando una página es requerida por un explorador, ASP .NET crea un objeto de tipo *Page* conteniendo la misma, y luego ejecuta su evento INIT, el cual es el encargado de configurar los diferentes controles contenidos por ella. Una vez que éstos han sido establecidos (pero antes de que ésta sea enviada al cliente a través de http) el evento load es iniciado, y es allí donde debe situarse todo el

código necesario para inicializar los diferentes valores y propiedades de controles de la misma. Una vez que la pagina esta lista para ser enviada al explorador (pero antes que esto suceda) el evento PreRender es iniciado. El mismo tiene como principal diferencia con Load que en el momento en que éste es ejecutado, todos los cambios sobre la pagina derivados de Load y la implementación de otros eventos ya han sido efectuados. Por último, cuando todas las tareas han sido finalizadas (pero antes de que la página sea enviada al explorador) el evento UnLoad es iniciado. La siguiente figura muestra dicho proceso:

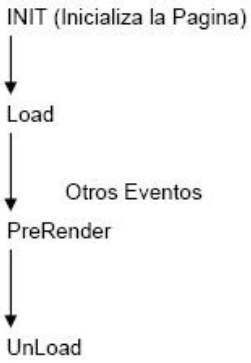


Figura 11. Proceso de funcionamiento de la página

Vimos anteriormente que los formularios Web estaban constituidos por dos archivos: aquel que contenía el código y aquel que contenía la información sobre la presentación. Si bien dijimos que ambos eran vistos por ASP .Net como un objeto de tipo Page, en realidad existe una separación aún más fina.

Es importante comprender dicha diferencia, ya que cuando se está en modo de diseño en Visual Basic y se accede a la ventana de propiedades mediante la tecla F4, la misma exhibe aquellas correspondientes al objeto *Document* (presentación), más las del objeto *Page*, ver la siguiente figura:

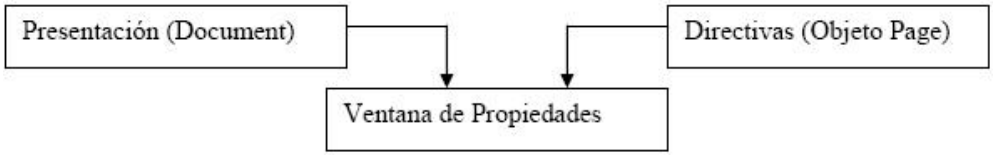


Figura 12. Diagrama de los archivos constituidos en el formulario Web

Las propiedades del documento (*Document*) afectan a la presentación (color, tamaño, etc.), mientras que las del objeto *Page* modifican la forma en que el código será tratado. Por ejemplo la propiedad *bgColor* modifica el color del fondo de la página y pertenece a *Document*, mientras que la propiedad *buffer* indica si la misma será generada y enviada al cliente a medida que el servidor la vaya construyendo, o después de que se haya finalizado con la misma, y pertenece a *Page*.

Como vemos, la ventana de propiedades es una serie de características de diseño (Document) y de ejecución (Page). Si bien algunas se traducen en código HTML y otras en lineamientos a seguir para la ejecución de la pagina, todas ellas son almacenadas como atributos dentro del archivo aspx, aunque en algunos casos pueden también ser establecidas desde programación (estas ultimas no viajan al explorador).

Funciones de ASP .Net

En el *espacio de nombres System.Web* para poder disponer de una variedad de clases que darán funcionalidad a nuestras aplicaciones Web.

HttpResponse. Cuando se necesita mandar información al cliente como *cookies*, salida HTML y direcciones se utiliza la clase *HttpResponse* mediante el objeto *Response* y el método *Write*.

```
Response.Write("La fecha es: " & Now.Date)
```

HttpRequest. Cuando se solicita información del explorador de una computadora cliente, tal como *cookies*, valores de las formas o una consulta se utiliza la clase *HttpRequest* mediante el objeto *Request.* Cabe mencionar que en ASP.NET muchas de las funciones de solicitud se hacen en forma automática, por esto se puede prescindir de éste para obtener valores, aunque sí se utiliza para algunas otras funciones.

```
Valor=Request.Form(“txtCaja.Text”)
```

Objeto.Redirect(“http://paginaWeb”). El objeto *Response* tiene un método llamado *Redirect* el cual dirige la página actual hacia otra; sin embargo, hay que tomar en cuenta que cuando el navegador de Internet encuentra una página que dirige hacia otra, el servidor envía la orden: *“se debe de ir a otra página”*, después de esto el explorador toma la dirección destino y va hacia esa página, por lo tanto este método ocasiona una ida y vuelta extra al servidor.

```
Response.Redirect(“http://www.otraPagina.com”)
```

Lock y Unlock

Los miembros *Lock* y *Unlock* se utilizan para que no existan problemas de concurrencia, ya que se debe alterar una variable por sesión a la vez.

```
'Pongo un candado para que nadie más, además del que accedió primero,
'pueda alterar la variable. Además de incrementar en uno el contador

Application.Lock Application("Contador")=Application("Contador") + 1

'Quito el candado para que otra sesión pueda aumentar el contador

Application.Unlock
```

Variables de Aplicación

Como lo explicamos antes, cada que una página es solicitada se regenera completamente. Es por eso que no guarda valores, y se utilizan las variables de aplicación, en las que no se permiten almacenar valores que puedan ser compartidos por todos los usuarios que acceden a la aplicación.

El objeto *Application* permite que se almacene información dentro de la memoria del servidor Web, mientras que el servicio de IIS (Internet Information Server) no sea detenido. Por ejemplo, podemos indicar en qué momento se inició la aplicación y almacenarlo en una variable llamada *T_DeComienzo*.

```
'Now indica la fecha y hora actual del servidor.

Application("T_DeComienzo")=Now

Para acceder a esta variable lo podemos hacer de la siguiente forma:
Response.Write("La aplicación comenzó: " & Application ("T_DeComienzo"))
```

Escribirá en la pantalla del navegador de Internet la fecha en que comenzó la aplicación.

También podríamos almacenar el número de visitantes que ha tenido nuestra página, para comenzar necesitamos asignar algún valor a la variable.

```
Application.Add("Clientes",2)
```

Para obtener el valor de esta variable podemos utilizar el método *Get* del objeto *Application*, cabe mencionar que no es necesario declarar previamente la variable.

```
Variable=Application.Get("Clientes")
```

Variables de Sesión

Cuando se necesita guardar información por usuario, entonces se utilizan las *variables de sesión*. La diferencia de este tipo de variables contra las de aplicación es que las variables de sesión tienen un conjunto de valores por usuario y las de aplicación son globales, las comparten todos los usuarios que acceden a la aplicación.

El valor de estas variables son mantenidas mientras la sesión no termine, ya sea que el tiempo máximo de espera se alcance o el usuario cambie a otra página que no pertenezca a la misma aplicación.

Es importante establecer lo que es una sesión para ASP.NET, cada ventana del navegador de Internet abierta es interpretada como una sesión. Por ejemplo, si tenemos tres ventanas del navegador abiertas en

la misma computadora, será interpretada por ASP.NET como tres sesiones abiertas, sin embargo generalmente existe una sola ventana abierta del navegador de Internet por usuario.

Para guardar la fecha y hora en que comenzó la sesión.

```
Session("InicioDeSesión")=Now
```

Para obtener el valor.

```
Response.Write("La sesión comenzó: " & Session("InicioDeSesión"))
```

Controles de servidor ASP.NET

Además de (o en vez de) utilizar bloques de código <% %> para programar contenido dinámico, los programadores de páginas ASP.NET pueden utilizar controles de servidor ASP.NET para programar páginas Web. Los controles de servidor se declaran dentro de un archivo .aspx mediante etiquetas personalizadas o etiquetas HTML intrínsecas que contienen un valor de atributo **runat="server"**.

Las etiquetas HTML intrínsecas las controla uno de los controles del espacio de nombres **System.Web.UI.HtmlControls**. A cualquier etiqueta que no esté explícitamente asignada a uno de los controles se le asigna el tipo de **System.Web.UI.HtmlControls.HtmlGenericControl**.

La sintaxis de cómo se van a crear los procesos o eventos en los formularios Web están establecidos con la sintaxis de Visual.Net, por lo tanto para poder realizar estos eventos hay que tener conocimientos sobre el lenguaje.

Más adelante en la parte de la creación de la página se destacarán los elementos de código en los cuales se podría desconocer el cómo hacer la conexión hacia la base de datos.

ADO .NET

ADO .NET plantea una evolución natural, ya que se basa en la manipulación de información obtenida de orígenes de datos en forma desconectada. La idea principal de ADO .NET es la de emplear las conexiones únicamente bajo demanda; esto quiere decir que la aplicación estará por defecto desconectada del origen de datos, y solamente en el momento de requerir los servicios del mismo se establecerá una comunicación a éste.

Esta aproximación permite al motor de datos resguardar un gran número de recursos. Existen dos tipos de ejecución factibles de ser realizadas sobre un origen de datos. Veamos entonces cómo resuelve el modelo desconectado propuesto por ADO .NET cada una de ellas.

- Acciones Unidireccionales
- Acciones Bidireccionales

El primer paso involucra aquellas acciones que no retornan filas, como una actualización, modificación o eliminación de uno o varios registros. Para este caso la solución en forma desconectada es relativamente sencilla, ya que en el momento de requerir la acción, el mismo podrá establecer una conexión con el origen, enviar la sentencia SQL y posteriormente desconectarse del mismo.

El segundo caso involucra aquellas acciones que dependen de ambos lados, como aquellas que retornan filas al ordenador. Evidentemente, la solución de esto último bajo el modelo desconectado requiere de una mayor complejidad.

Al igual que en el caso anterior, se abre la conexión al motor de datos para ejecutar la sentencia SQL, la cual retornará un conjunto de filas, que serán trasladadas y almacenadas en forma local, y posteriormente se cerrará la conexión con el origen. Para lograr esto ADO .NET cuenta con un conjunto de objetos que sirven de buffer para guardar en el cliente las

filas obtenidas de la consulta al servidor. Como consecuencia, las acciones realizadas sobre las filas (modificaciones, eliminaciones, etc.) serán gestionadas en forma local, o lo que es igual, utilizando un juego privado de datos.

En ADO ya se contaba con una característica similar denominada “Cursores Desconectados”, pero el proceso de conexión y desconexión debía realizarse en forma explicita. En ADO .NET los objetos están naturalmente desconectados del origen, y son estos últimos los que permiten en forma local filtrar, ordenar y hasta agregar, eliminar o modificar una o varias filas. Por supuesto que las acciones serán realizadas sobre la copia privada, sin que ello afecte a la información guardada en el motor de datos. ADO .NET cuenta con métodos que permiten posteriormente sincronizar la información del cursor local con el origen, y de esta forma hacer efectivos los cambios.

Las Clases de ADO.NET

ADO .NET incluye varias clases para la manipulación de datos, las cuales están incluidas en varios espacios de nombres:

```
Imports System.Data
Imports System.Data.OleDb
Imports System.Data.SqlClient
```

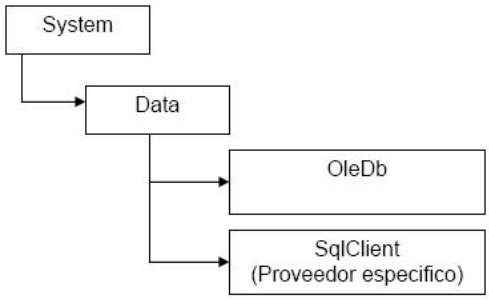


Figura 13. Diagrama de clases ADO.NET

Las clases en ADO .NET se separan en dos grandes grupos, teniendo en cuenta la tarea que las mismas desempeñan:

1. Consumidores de Datos.
2. Gestores de Información.

Dentro del primer grupo se encuentran las clases que se encargan exclusivamente de realizar una conexión y obtener los datos. Como ejemplo de estas tenemos, las localizadas dentro de OleDb y SQLClient.

Dentro del segundo grupo se hallan aquellas que gestionan la información una vez obtenida. Esto permite a ADO .NET emplear las mismas clases para gestionar los datos, sin importar si éstos fueron obtenidos a través de un proveedor específico (SQLClient) o genérico (OleDb). De acuerdo con ello, quienes obtienen la información podrán estar relacionados con un motor de datos específico, pero quienes lo gestionen posteriormente serán comunes a todas ellas.

¿Cuándo se debe utilizar un proveedor específico?

Los proveedores de datos en .NET permiten acceder a orígenes de datos específicos. Se puede usar el espacio de nombres System.Data.SQLClient para acceder a SQL Server 2000 y versiones anteriores, y el espacio de nombres System.Data.OLEDB para acceder a cualquier origen de datos expuesto a través de OLEDB.

En Visual Basic y versiones anteriores de ASP se tenía el Concepto *Recorset*, el cuál es una tabla que contiene los datos que manejará nuestra aplicación ASP. Esta tabla almacena el resultado obtenido por las consultas realizadas sobre la base de datos a la que nos encontremos conectados mediante el objeto *Connection*.[6]

En ADO .NET el objeto RecordSet no existe, pero en su reemplazo se ofrece uno llamado DataSet. El mismo es el elegido para gestionar los cursores en ADO .Net, así como también para el intercambio de datos entre aplicaciones o componentes.

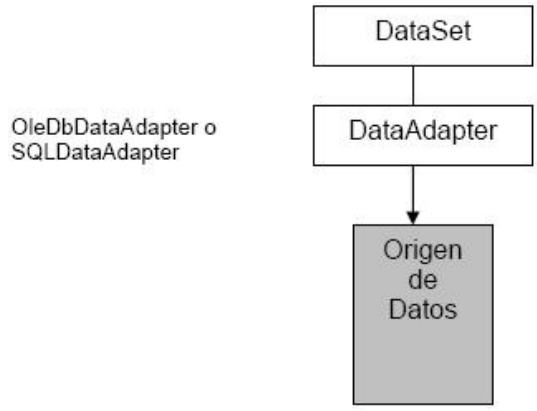
[6] En http://www.aspfacil.com/articulos/040401.asp

El DataSet permite almacenar filas de uno o varios resultados, pero, a diferencia del primero, en éste es posible establecer las relaciones entre sus integrantes. Sin embargo, la diferencia más importante entre un RecordSet y un DataSet es que este último representa una vista local o en memoria de las filas obtenidas de un origen de datos.

Esencialmente, un objeto DataSet obtiene un conjunto de filas de una o más tablas, y posteriormente las aloja en memoria. Una vez realizado este proceso, el mismo procede a desconectarse del origen de datos.

El DataSet se encarga posteriormente de la gestión local de las filas obtenidas, así como también de las posibles acciones a ser realizadas sobre las mismas (modificaciones, eliminaciones, etc.) todas las acciones serán aplicadas sobre la copia privada o local de datos, sin que ello interfiera en el origen. Posteriormente, se necesitará sincronizar la información con este último, a los efectos que los cambios puedan hacerse efectivos.

Para dicha tarea, se involucra un segundo objeto llamado DataAdapter, el cual funciona de intermediario o puente sincronizador entre la copia privada y el origen de datos.



Esencialmente, un DataAdapter establece una conexión, y luego envía al origen de datos la acción SQL por cada registro modificado, agregado o eliminando en la copia local, y posteriormente se desconecta.

Figura 14. Diagrama de funcionamiento del DataSet

Esta tarea no necesariamente debe realizarse al finalizar cada una de las acciones, sino que la misma puede ser llevada a cabo durante períodos

determinados de tiempo, el DataSet puede también desechar la copia local y obtener un nuevo juego de datos más actualizado si así lo desea.

2.5.4. Microsoft Visual Studio 2003

Es uno de los paquetes principales para desarrollar elementos .Net, y por consecuencia paginas con extensión para asp.net (.aspx). Para tener un mejor entendimiento sobre la plataforma Microsoft.NET definiré los rasgos más importantes sobre esta

Objetivos de la infraestructura .Net

Microsoft ha tenido que implementar un nuevo conjunto de tecnologías, las cuales remplazan y/o mejoran defectos de su plataforma anterior. A continuación se resumen los objetivos principales que intenta solucionar la nueva infraestructura .Net:

1. Eliminar de raíz los inconvenientes de modelo COM. Para ello, emplea un nuevo estándar con una nueva aproximación basada en ensamblados (se les llama ensamblados a los nuevos componentes y aplicaciones factibles de generar desde cualquier lenguaje de Visual Studio .Net y productos compatibles con la infraestructura) y un almacén o caché global de los mismos.

2. Proveer manejo automático de la memoria empleada por los componentes, sin que el desarrollador tenga que ocuparse de esta tarea. Esto es cierto incluso para aplicaciones implementadas en Visual C++.

3. Ofrecer una mayor sencillez para utilizar componentes desarrollados en otros lenguajes, e incluso residentes en otros sistemas operativos.

4. Hacer más fácil la instalación, a los efectos de que sea posible dejar un componente funcional simplemente copiándolo al directorio de la aplicación.

5. Proveer el mismo modelo para utilizar un componente localmente o a través de una red.

6. Hacer más fácil el desarrollo de aplicaciones distribuidas, y especialmente de soluciones para Internet mediante un único entorno de desarrollo.

7. Proveer ejecución segura a través de un nuevo modelo de seguridad para aplicaciones y componentes (semejante a los servlets de Sun Microsystems).

8. Hacer más fácil el desarrollo mediante un conjunto de funcionalidades previamente hechas, las cuales cubran varios aspectos comúnmente utilizados.

9. Proveer características de orientación a objetos para todos los lenguajes de la infraestructura en forma nativa.

Como se mencionó, Visual Studio puede crear diferentes tipos de lenguajes, y para cada uno de los diferentes tipos de lenguajes se pueden crear diferentes tipos de proyectos con sus elementos correspondientes.

Los tipos de proyectos que disponibles son los siguientes:

- Proyectos de Visual Basic
- Proyectos de C#
- Proyectos de C++
- Proyectos de J#
- Proyectos de instalación e implementación
- Otros Proyectos
- Soluciones de Visual Studio

Para cada uno de estos Visual Studio tiene diferentes tipos de plantillas. Las plantillas son los elementos de los cuales consta el proyecto para desarrollas los diferentes tipos de recursos que se requieren programar.

Por mencionar algunos de los tipos de plantillas generales para los lenguajes como Visual Basic, C#, y J# son los siguientes:

- Aplicación para Windows
- Biblioteca de Clases
- Biblioteca de controles de Windows
- Aplicación de Web ASP.NET
- Servicio Web ASP.NET
- Aplicación de Web de ASP.NET Mobile
- Bibliotecas de controles Web
- Aplicación de consola
- Servicio de Windows

Estos se utilizan dependiendo de la necesidad de programación, y por lo tanto la que se va a utilizar para nuestro proyecto va a ser la de Aplicación de Web ASP.NET.

Solo cabe mencionar que la instalación de Microsoft Visual Studio 2003 es tan sencillo como los demás paquetes de instalación de Microsoft, solo es seguir los pasos de instalación y el paquete quedará completamente configurado. Pero antes de realizar la instalación se debe de tener en cuenta que esté instalado el Servidor de paginas Web Avanzado de Microsoft (IIS) para poder tener acceso a los recursos ASP.NET. Este servidor de páginas Web se distribuye gratuitamente con las versiones de Windows basadas en NT, como puede ser Windows 2000 Profesional o Windows 2000 Server, así como en Windows XP en sus versiones Profesional y Server.

Tan sólo con entrar a Panel de Control de Windows, en el icono de “Agregar o Quitar Programas” y en la parte de “Agregar o Quitar

componentes de Windows" se busca el recuadro que muestre el elemento "Servicios de Internet Information Server (IIS)" se selecciona y Windows hará el resto. Dejará configurado todo lo demás creando una carpeta llamada "Inetpub" que es donde se alojarán los elementos creados por las páginas Web para ser cargados al servidor.

2.5.5. Macromedia Dreamweaver MX 2004

Macromedia Dreamweaver es una de las herramientas de edición profesional para la creación de páginas Web de manera sencilla y con la cual se pueden utilizar los instrumentos que requerimos para acceder a la base de datos con PHP, o si así se requiere crear de manera grafica una página Web en PHP y ASP.Net.

Otro de los beneficios de Dreamweaver es que se adapta a otro programa de Macromedia como es Flash. Macromedia Flash es uno de los programas que en la actualidad ha tenido mayor impacto por su alcance visual, ya que maneja gran cantidad de posibilidades de animación y efectos que hacen de una página Web más interesante sensorialmente.

Al poder Dreamweaver explotar las posibilidades de Flash, nos da las herramientas de elaborar un entorno visual completo y un contexto de gran impacto al usuario final que visite la página.

Las funciones de edición visual de Dreamweaver MX 2004 permiten agregar rápidamente diseño y funcionalidad a las páginas, sin la necesidad de programar manualmente el código HTML.

No obstante, si prefiere crear el código manualmente, Dreamweaver también incluye numerosas herramientas y funciones relacionadas con la codificación. Además, Dreamweaver le ayuda a crear aplicaciones Web dinámicas basadas en bases de datos empleando lenguajes de servidor.

Se puede crear tablas, editar marcos, trabajar con capas, insertar comportamientos JavaScript, etc., de una forma muy sencilla y visual.

Además incluye un software de cliente FTP completo, permitiendo entre otras cosas trabajar con mapas visuales de los sitios web, actualizando el sitio web en el servidor sin salir del programa.

Al comenzar a utilizar Dreamweaver MX 2004 aparecerá una pantalla:

Figura 15. Ventana principal de Dreamweaver

En dicha pantalla podemos seleccionar una serie de elementos para crear nuestra página, ya sea de HTML, PHP, ASP.Net o los demás tipos de lenguajes con los que se puede crear una Web; y un menú común de todos los programas.

El programa utiliza sitios, los cuales sirven para poder utilizar los recursos que se necesitaran en la página los cuales se quedaran alojados dentro de una carpeta.

Antes de crear alguna página se necesita definir el sitio por lo menos, y para realizar esto solo se necesita acceder a la parte del menú

Ya ingresado en “Administrar Sitios” aparecerá una ventana en la cual se tendrá la lista de los sitios. Como no existe ninguno se seleccionará en *“Nuevo”*.

Figura 16. Ventana de Administrar sitios de Dreamweaver

Ahí se seguirán los pasos correspondientes para el crear el sitio seleccionando la pestaña de “*Básicas”* ó “*Avanzadas”*.

En *Básicas* se va paso por paso el ayudante comenzando por el nombre del sitio. Después la tecnología de servidor en la cual se podrá elegir entre los alguna de las 2 que se han mencionado como es PHP o ASP.Net VB. Al seleccionar uno sigue la opción de especificar cómo se van a trabajar los archivos durante la etapa de desarrollo, aquí se recomienda editar localmente y luego cargar al servidor de prueba remoto, para poder desarrollar con mayor comodidad; también se requiere especificar la ruta en la cual se van a almacenar los archivos. Dando siguiente pasará a la parte en que se especifica la conexión del servidor de prueba, la cual es *“Local/Red”*, y la ruta en la cual se copiarán los archivos para comprobarlos antes de subirlos al servidor, de igual manera se selecciona una ruta en la cual van a ser copiados y que sea diferente a la ruta anterior.

Posteriormente se tiene que definir la URL para examinar la raíz del sitio, la cual es *http://localhost/*.

Y por último se pregunta si se quiere activar la protección y desprotección de archivos para que no se pueda editar el archivo al mismo tiempo por

más personas, y en el cuál dependerá de cuántas personas estarán colaborando con el diseño del sitio, pero si sólo va a ser una sola persona no es necesario activar la opción.

Y en *Avanzadas* aparecerán varias categorías:

- Datos locales. Se especifican el nombre del Sitio y la ruta de la carpeta Raíz como se hizo anteriormente.
- Datos remotos. Se especifican el Acceso (Local/Red), la carpeta remota (la otra ruta en donde serán copiados los archivos para ser comprobados).
- Servidor de prueba. Se especifican el modelo de servidor (PHP o ASP.Net VB), el acceso (local/Red), la carpeta del servidor (la misma ruta que en los datos remotos) y el prefijo de la URL (http://localhost/).

Las demás opciones dependerán de la página y del diseñador.

Para cualquier página creada Dreamweaver visualiza la parte de código y de diseño. Pueden observarse independientemente o visualizarlas conjuntamente en la opción de dividir.

CAPÍTULO 3

LA EMPRESA

3.1 CARACTERISTICAS DE LA EMPRESA

Para poder tener un resultado factible de la creación de la página Web dinámica se tienen que conocer principalmente las características de la empresa. Estas muestran el tipo de empresa que es, su forma de trabajar, a quien va dirigido su producto o actividad, etc.; pero como manera principal debe tener una Visión, Misión, Política, Objetivos y Valores que darán a conocer a sus consumidores o clientes.

El tipo de empresa que puede ser no se puede especificar tanto, pero generalizando un poco se podría tomar como de ventas, de servicios, informativa, de entretenimiento o meramente personal.

La forma de trabajar se refiere al manejo de información, productos, clientes o servicios con los que cuenta para que el cliente o usuario se de cuenta de los servicios que la empresa les brinda.

Para esto también la empresa debe especificar a quien va dirigida para poder brindar un servicio cómodo y eficaz al usuario final.

MISION – VISION – POLITICA – OBJETIVO – META

Estos términos suelen generar confusión porque no hay un criterio unificado sobre su significado y uso, pero en el momento de definir, establecer e implementar las políticas de la empresa, los conceptos asociados a estos términos son de gran utilidad.[1]

[1] en http://www.estrucplan.com.ar/articulos

Figura 17. Filosofía de la empresa

La definición de la misión ayuda a clarificar las posiciones y creencias de los integrantes de la dirección de la empresa y a unificar criterios básicos. No es estrictamente necesario definirla para arribar a las políticas, especialmente en el caso de un único director o dueño, pero siempre es recomendable hacerlo, sobre todo para transmitir al personal los valores que constituyen la base de la cultura de la empresa.

La visión es imprescindible, sobre ella se construyen las políticas de la empresa.

Ejemplos de visión son:

- ⇒ Si la empresa es mediana, ¿seguirá siendo mediana o la dirección quiere construir una gran empresa?
- ⇒ Si la empresa es líder en su país, ¿se plantea extender su liderazgo al continente o al mundo?

Quizás la empresa nunca podrá realizar su visión, pero trabajará siempre en pos de ella. Si la política es el faro, la visión es la estrella.

Los objetivos definen los logros cuantitativos y medibles que llevarán al cumplimiento de la política.

¿Y las metas? Cuando el cumplimiento del objetivo implica el cumplimiento de diferentes etapas o actividades, simultáneas o no, es posible hablar de “metas”. También deben ser cuantitativas y medibles.

Así, por ejemplo, si un objetivo para el 2003 es “Aumentar las ventas un 5%”, pueden establecerse metas trimestrales que acumuladas conforman el objetivos del 5%.

3.2 NECESIDADES DE LA EMPRESA

Las necesidades de la empresa tienen que ser marcadas antes de realizar la página Web dinámica, ya que de esto depende el resultado de la elaboración de esta.

Para que esto suceda se tiene que hacer un amplio estudio de los requerimientos de los clientes y de las capacidades de la empresa para poder llenar estos requerimientos.

En otras palabras, si es que la empresa quiere crecer más debe tener los suficientes recursos para poder cumplir las exigencias del cliente.

Si es viable la empresa entonces se tienen que aterrizar estos requerimientos. Por ejemplo, si se tiene una empresa de venta de productos computacionales, ésta vende sólo para la ciudad de Celaya Guanajuato, pero quiere crecer más de la ciudad y vender en todos los alrededores, entonces sus requerimientos son estos:

- Llegar a todo tipo de público,
- Dar a conocer la empresa por medio de una página Web,

- Que el cliente pueda hacer sus pedidos desde la página,
- Tener un acceso a esos pedidos de manera inmediata,
- Que muestre la página todos los productos que tenemos de manera que el cliente pueda conocerlos, etc.

De esta manera se facilitará el desarrollo de la página y cubrirá todas las necesidades de la empresa y del cliente.

CAPÍTULO 4

DISEÑO Y CREACION DE LA PÁGINA WEB DINÁMICA

4.1 REQUERIMIENTOS PARA LA PÁGINA

Para poder tener una página eficaz ya en su finalización tiene que contener todos los puntos que requiera mostrar la empresa. Como ya se tienen planteados los puntos de los cuáles la empresa quiere darse a conocer, ahora se tendrán que aplicar dentro del contenido dentro de la página Web.

¿Qué es lo que requiere la empresa, negocio u organización que se muestre en la página?

Emplearé la misma plataforma de la creación de la base de datos para comenzar a dar una idea de qué es lo que se tiene que comenzar a hacer.

Comenzaré con esta parte tan sencilla. En nuestra base de datos plantee la idea de una empresa que contenga productos relacionados hacia una clasificación específica. Por otra parte di a entender que puede acceder un usuario y registrarse con sus datos personales.

Sus requerimientos son el mostrar la clasificación de productos con los que cuenta, y de tales desplegar de manera ordenada cada uno de estos dentro de la página. Y por la otra parte es que el usuario que entre pueda ser tomado en cuenta como usuario frecuente de la página, registrándose con la empresa por medio de algunos datos personales que les pueda brindar, y así tener un control de eficacia optimo para el contacto directo con él en el caso de que se interese por alguno de los productos que se manejan.

Esto es un simple ejemplo de cómo se pueden determinar los requerimientos, pero para ser más exactos se tendrán que tener todas las partes de los requerimientos y estos podrían ser:

- Requerimientos Visuales,
- Requerimientos de datos y

- Requerimientos de respuesta

4.1.1. Requerimientos visuales

Se refiere al contenido de colores, formas y diseño que tendrá la página.

Deben de adecuarse a lo que representa mejor la empresa; por ejemplo, no tendrá las mismas formas una página diseñada a una empresa que vende videojuegos que a una que vende muebles, ya que la de videojuegos necesitará más colores llamativos que capte la atención de sus clientes que por mayoría son de niños y adolescentes, y la que vende muebles necesitará unos colores mas formales para poder hacer sentir al cliente más cómodo, y que como clientes serán personas de edad más adulta.

4.1.2. Requerimientos de datos

Se refiere al contenido que el usuario va a consultar específicamente.

Tal vez el usuario entró a buscar cierta información específica, pero se le puede dar a conocer una gran diversidad de referencias sobre algo que no tenía en la mente encontrar pero que podría importarle en ese momento. Así se tendrán ya 2 razones por las cuales tiene que permanecer en el sitio Web. Y si así se le va añadiendo información que le agrade saber se tendrá entonces qué pensar todas las variantes que un usuario puede llegar a consultar y que la empresa posee para informar o brindar.

4.1.3. Requerimientos de respuesta

Se refiere al nivel que se tendrá para poder interactuar con el usuario.

Pueden formularse preguntas dentro de la empresa como ¿los usuarios habrán encontrado todo lo que querían?, ¿cómo saber qué es lo que

quieren?, ¿de que manera pueden saber ellos si ya tenemos lo que les interesa?

De ahí conlleva el asunto de tener en cuenta un nivel de respuesta, ya sea directa o indirectamente, si va a ser de un solo canal o de manera reciproca.

Por ejemplo, se podrá tener una página Web que mantenga informado al público de clima diariamente. La empresa climatológica diariamente informará sus resultados hacia el usuario y nada más. No se tendrá en este aspecto comunicación recíproca en la página. Pero en otro caso podría colocarse un apartado de preguntas u opiniones sobre la misma página de la empresa climatológica, lo cual hará saber a la empresa cuáles son algunas de las necesidades que ellos no habían tomado en cuenta que el público requiere.

Todo esto depende del tipo de empresa y de la información que quiere darse a conocer.

4.2 CREACIÓN DE LA PÁGINA WEB DINÁMICA

Ya teniendo una dualidad de manejadores de base de datos, de lenguajes para utilizar en aspectos dinámicos, y de software que nos ayudan a diseñar nuestros códigos HTML de manera sencilla podemos a empezar a crear una página Web dinámica que se adapte a nuestras necesidades.

Para una mejor explicación se crearán las páginas con un lenguaje, un programa para diseñar la página y los 2 tipos de base de datos, entonces quedará de la siguiente manera:

- ACCESS/SQL-PHP-DREAMWEAVER
- ACCESS/SQL-ASP.NET-VISUAL STUDIO.NET

Para poder utilizar PHP y Dreamweaver en Apache sin tener tantos problemas te recomiendo utilizar el paquete WAMP5[1] en su versión 1.4.4, ya que solo se instala el paquete y al irse instalando te pregunta cuál es la carpeta que va a utilizar el servidor para poder comunicarse con PHP y listo.

NOTA: ese mismo directorio se tendrá que poner al configurar el sitio en Dreamweaver en su apartado correspondiente.

Entonces crearé una página con las mismas características en cada uno de los equipos correspondientes, con el objetivo de que el programador tenga manera de elegir cual utilizar.

En las bases de datos contendrá las siguientes tablas:

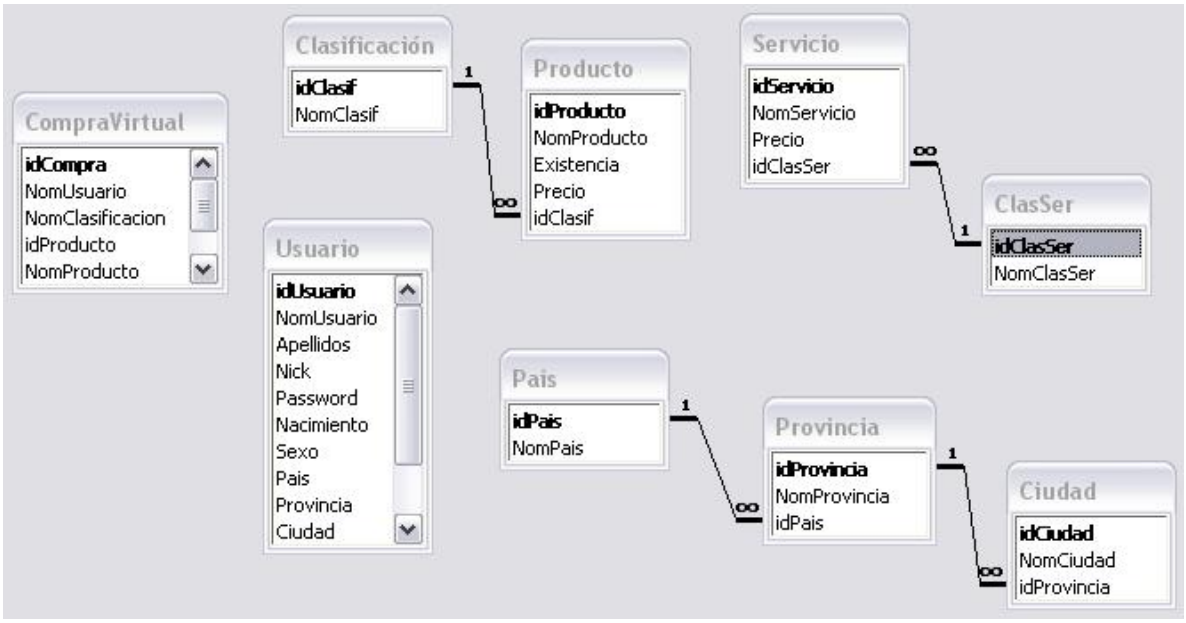


Figura 18. Tablas de la base de datos “Jeadeca”

[1] Lo puedes descargar en http://www.wampserver.com/en/index.php

La página será hecha para una empresa pequeña que se dedica a venta, reparación y mantenimiento de equipo de cómputo. Sus características son las siguientes:

- **Misión**. Ayudar a las personas, empresas, escuelas, etc., a encontrar una solución para sus necesidades computacionales de manera sencilla, ágil y comprometida.
- **Visión**. Expandir nuestras aportaciones a todas las personas y empresas sin que ellos tengan la necesidad de lidiar con inconvenientes en poder encontrar los servicios que necesiten.
- **Políticas.** Honestidad, excelente comunicación, comprensión y pasión por el trabajo de cada uno de nuestros consumidores o clientes.
- **Objetivos.** Mantener un equilibrio de trabajo y personal para poder cumplir todas las expectativas laborales y del consumidor, apropiar las necesidades de los clientes para así darle la importancia que se merecen. Siguiendo este reglamente podemos localizar o satisfacer a un cliente más de los que tengamos cada trimestre.

Sus expectativas son impactar con sus servicios a todas las personas, empresas o lugar que requieran de mantenimiento, reparación o venta de equipo de cómputo. Por lo tanto al tener la posibilidad de contar con una página Web darán a conocer a todo este sector sus servicios como si se estuvieran comunicando personalmente con la empresa.

La página deberá tener colores sobrios para que den un aspecto formal al cliente, y que de esa manera pueda sentirse cómodo mientras navega en la página. Además se requiere que tenga pequeñas animaciones que hagan elegante ciertas secciones.

Que cuente con absolutamente todos los productos que la empresa vende y repara. Además formularios de llenado de datos para que la empresa

conozca y tenga acceso más ágil con el cliente y poderle brindar un servicio de mejor calidad y más directo.

Con esto poder saber qué es lo que quiere el consumidor y que por medio de la página nos mande su solicitud del servicio y la empresa comunicarse directamente con él y darle una respuesta pronta.

4.2.1. Access / SQL - PHP - Dreamweaver

Ya conociendo las tablas que ya fueron creadas con anterioridad comenzaremos a utilizar Dreamweaver para la creación de la página.

En la pantalla principal se dará la opción de crear página dinámica PHP.

Esta primera página la utilizaremos como página principal, en la cuál los usuarios entrarán como primera instancia. Tendrá como nombre ***index.php*** y su diseño será de la siguiente manera:

4.2.1.1. index.php

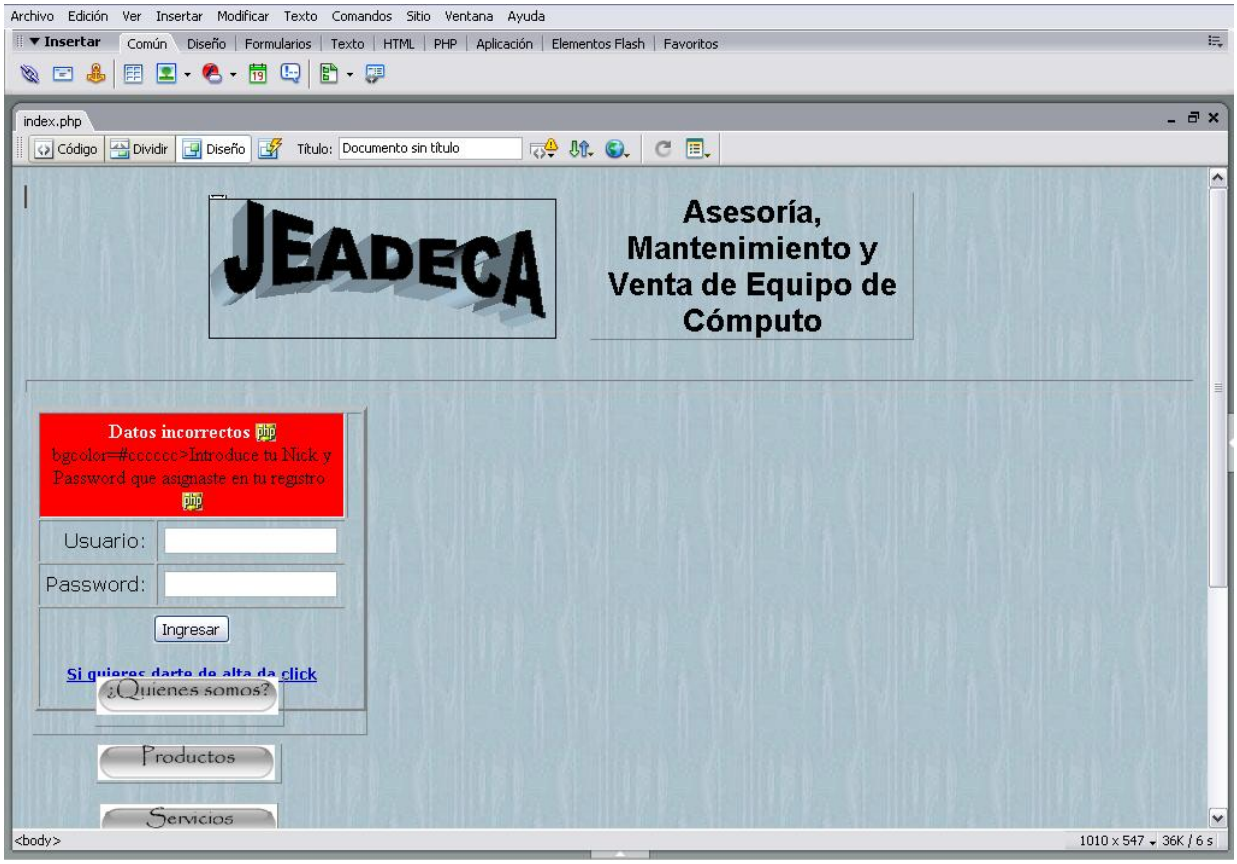


Figura 19. Diseño de *index.php*

Como se ve en la imagen anterior, hay una imagen que es el logo de la empresa, texto al lado derecho de esta que referencia a lo que se dedica la empresa; debajo de ella una línea que es una regla horizontal que va a separar la parte del encabezado de las páginas con el contenido de cada una.

Más abajo se encuentra un formulario, que en el cual se encuentra una tabla que nos servirá para pedir los datos de ingreso de los usuarios, y en el caso de que no este registrado, una opción de texto para que se de de alta. Debajo del formulario se encuentran unos botones los cuales hacen que en el centro de la página despliegue información dependiendo de cuál botón se active.

Ya descrito el contenido de la página voy a explicar cómo se hizo cada parte. Principalmente se hizo en PHP y no HTML porque la página va a recibir información con variables por URL o como se explicó antes por el método *_GET*.

Cada uno de los elementos de la página esta dentro de una capa, las capas se sitúan en la parte del menú en *INSERTAR—OBJETOS DE DISEÑO—CAPA*. La razón por la cuál se insertan todos los elementos en una capa es para poder manipular su posición libremente.

Entonces se inserta una capa y dentro de la capa se inserta una imagen. Para insertar una imagen se debe de ir en el menú en *INSERTAR—IMAGEN.*

Para el texto que esta al lado de la imagen se inserta una nueva capa y dentro de la capa se comienza a escribir. Y para poder modificar el tamaño de la letra o el tipo de fuente en la parte de propiedades se modifican.

Para la regla no es necesario insertar la capa. La regla se encuentra en el menú en *INSERTAR—HTML—REGLA HORIZONTAL*.

El formulario se encuentra en *INSERTAR—FORMULARIOS*. La tabla en *INSERTAR*. En la tabla el programa preguntará el número de filas y

columnas, pero en propiedades se puede modificar. En este caso coloqué 4 filas y 2 columnas. La primera fila la voy a convertir en una sola celda, con el Mouse selecciono las dos celdas de la fila, ya que estén seleccionadas, en las propiedades hay una opción que dice “Celdas” “combina las celdas seleccionadas usando extensores”, se selecciona esa opción y se convierten las 2 celdas en una sola.

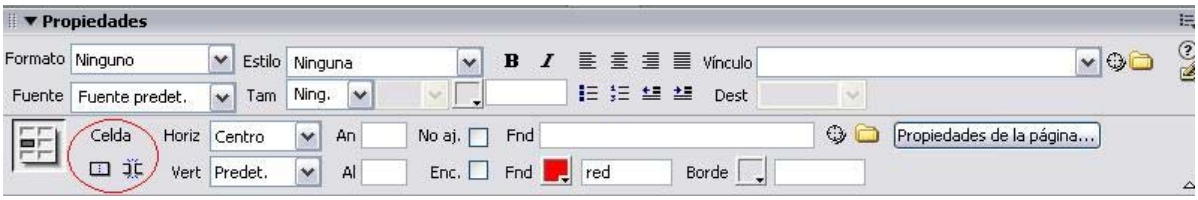


Figura 20. Propiedades de la tabla

En esa celda vamos a colocar un título que explique qué es lo que se tiene que hacer en la tabla, o sea que introduzca su Nick y Password de registro. Pero en el caso de que no este registrado y el Nick o Password estén incorrectos que también sobre esa celda despliegue un mensaje de error. Esto se introduce de la siguiente manera:

- En la parte del código de la página se posiciona en el lugar donde se encuentra el código de la celda.
- Se introduce la siguiente instrucción del lenguaje PHP

```
<td colspan="2"align="center"        ← Alineación del texto
bgcolor=red                          ← Color del fondo
<? if ($_GET["errorusuario"]=="si"){ ?><span style="color:ffffff"><b>Datos incorrectos</b></span>
<? }else{ ?>
bgcolor=#cccccc>Introduce tu Nick y Password que asignaste en tu registro
<? } ?></td>
</td>
```

La primera parte indica que si se recibe una variable por URL con el nombre de errorusuario y que su valor sea igual a “si” entonces escribirá

el texto de “Datos incorrectos” y con el color de fondo rojo. De lo contrario el color del fondo será gris e introducirá el texto “Introduce tu Nick y Password que asignaste en tu registro”. Por último se cierra condición.

En las siguientes celdas se colocarán el texto del lado izquierdo y del lado derecho la caja de texto en la que ingresará sus datos el usuario. En el primero será su Nick y en el segundo su Password. Para el texto solo se escribe dentro de la celda, y para colocar la caja de texto solo se posiciona sobre la otra celda y en el menú *INSERTAR—FORMULARIOS—CAMPO DE TEXTO.*

Por último se convierten las siguientes celdas en una solo como en el primer caso y se coloca un botón de envío de datos yendo al menú *INSERTAR—FORMULARIOS—BOTON*, y debajo de él un texto en el cuál se vincula hacia otra página, que será la de registro, en el caso de que no esté registrado un usuario, y quedará de la siguiente manera nuestro formulario. Ya para terminar, en la parte de las propiedades del formulario en *ACCIÓN* se pondrá ***control.php*** que es donde se dirigirán los valores al pulsar el botón de envío al cual le cambiaremos también sus propiedades y le llamaremos “*Ingresar*”. Además de eso el destino del formulario será sobre la misma ventana en la que tenemos abierta, entonces le pondremos *“_self”*.

Figura 21. Diseño del formulario de ingreso de usuarios

Debajo de este colocamos los botones flash de la siguiente manera:

En el menú en *INSERTAR—MEDIA—BOTON FLASH*. Nos abrirá una ventana y seleccionaremos un estilo de botón, el texto que estará en el botón, el tipo de la fuente, dejamos en blanco el vínculo, el destino, el color de fondo, y le ponemos un nombre.

Así será con los 3 botones, recordando que cada uno va sobre una capa independiente.

Al tener ya los 3 botones nos falta poner el contenido que desplegaran los botones al ser activados; para esto crearemos 3 capas mas y dentro de cada una de las capas el contenido de cada uno de los botones antes creados. Es decir, el primer botón se llama ***"¿Quiénes somos?"***, entonces en otra capa se escribirá la descripción de la empresa para hacer referencia al botón. Así mismo con los otros botones que se llamarán ***"Productos"*** y ***"Servicios"***. Quedarán 3 botones y 3 capas con la información de los botones y estas 3 capas de información en sus propiedades se ponen como ocultas o Hidden en la propiedad *Vis*.

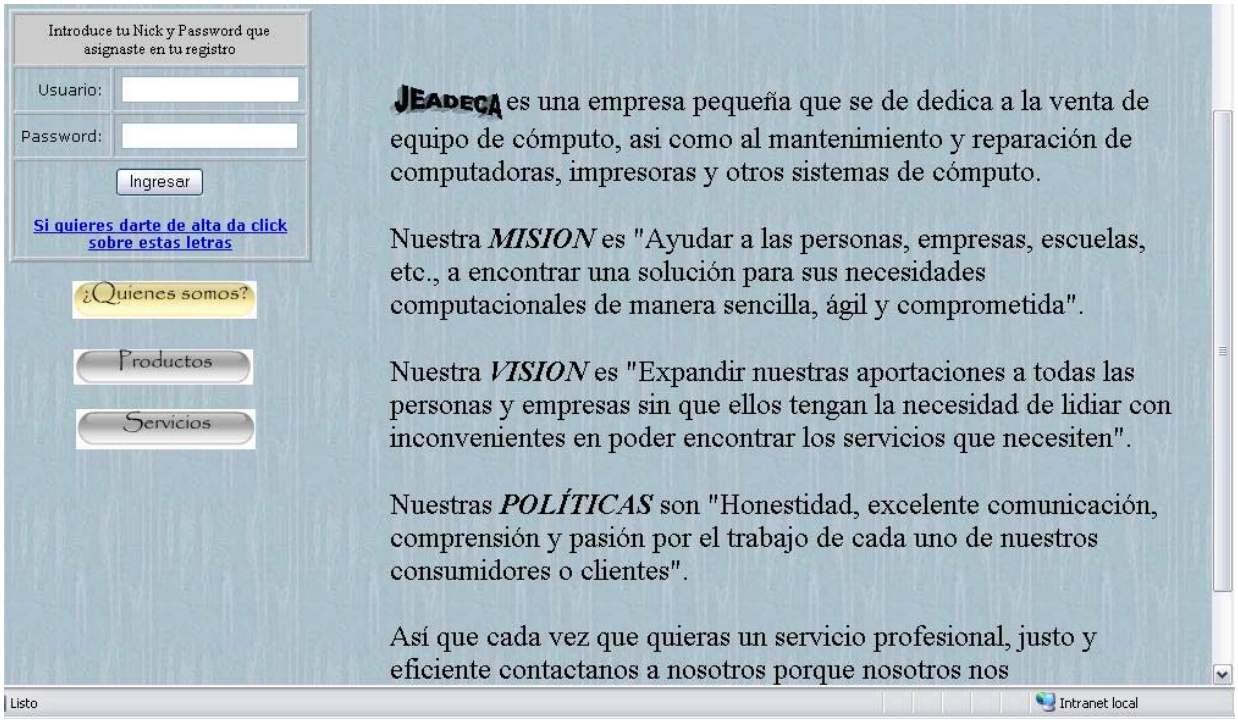


Figura 22. Diseño de los botones Flash

Ya teniendo esto, haremos que se oculte y muestre la información.

En las propiedades de “Etiqueta” que se activa con la combinación Shift+F3, en la opción de comportamientos se va a agregar un comportamiento que muestre y oculte capas. Para darle un comportamiento al botón, se selecciona el botón y se le da en comportamientos en el signo de **+** esta la opción de “Mostrar-ocultar capas”, después se selecciona la capa de información referente al botón y se pone “*MOSTRAR*” y en las demás capas de información de los botones se les pone “*OCULTAR*” y quedará de la siguiente forma:

Figura 23. Ventana de capas en su propiedad “Mostrar-Ocultar”

De la misma forma se va a hacer con los botones y la capa referente a su información.

Todo esto hará que al dar un clic sobre un botón mostrará su información y al dar sobre otro botón ocultará la capa anterior de la información del botón y mostrará la del último botón seleccionado.

El primer botón flash mostrará solo información en una capa.

El segundo botón flash mostrará información sobre los productos también en una capa pero también habrá otra capa pero con sus propiedades de visualizar como *oculta* (hidden), que tenga un botón flash. Este botón mandará hacia otra página en donde estará la lista de los productos.

El tercer botón flash mostrará información sobre los servicios también en una capa, pero también habrá otra capa oculta que tenga un botón flash que así como en el anterior mandará hacia otra página en donde enlistará los servicios.

Entonces, cuando se oculten las capas en la función de *MOSTRAR-OCULTAR CAPAS*, también se tiene que tomar en cuenta las capas de los botones flash. Es decir, cuando se de clic en el botón de ***¿Quiénes somos?***, en los comportamientos de las capas se tienen que ocultar las capas de la información de los botones de ***Productos*** y de ***Servicios*** y las capas de los botones flash que muestra la lista de los productos, y la lista de los servicios.

Asimismo, cuando se de clic sobre la de *Productos*, se ocultan las capas de información de *¿Quiénes somos?* y de *Servicios*, y la capa del botón que manda a la lista de *Servicios*.

Y de igual manera, cuando se de clic sobre la de *Servicios*, se ocultan las capas de información de *¿Quiénes somos?* y de *Productos*, y la capa del botón que manda a la lista de *Productos*.

Ya teniendo todos los elementos de nuestra página principal se crearán los enlaces a los que llevan ciertas partes de la página.

El primero es en el texto del formulario que regresará una variable por URL.

4.2.1.2. control.php

Se creará otra página PHP en la que no tendrá nada de diseño, únicamente código que nos servirá conectar a la base de datos y cerciorarse que los datos de usuario y contraseña en la página index.php sean correctos o mejor dicho, que concuerden con los que existen. Le llamaremos "***control.php***" y tendrá el siguiente código:

Código para Access

```
<?
// Se crean variables para guardar los valores que manda el formulario
$nick = $_POST['usuario'];
$password = $_POST['password'];
// Se crean variables para la conexión
$conexion=odbc_connect("access","","");
$query="select Nick, Password from Usuario where Nick='$nick' and Password='$password';";
$sel= odbc_do($conexion,$query);
$nom= odbc_result($sel,"Nick");
$pass= odbc_result($sel,"Password");
//vemos si el usuario y contraseña son validos
if ($nick==$nom && $password==$pass){
    //usuario y contraseña válidos
    //defino una sesión y guardo datos
        session_start();
        $_SESSION["autentificado"]= "SI";
        $_SESSION["usuario"]=$nick;
    // mando a la página jeadeca.php
        header ("Location: jeadeca.php");
  // ?user=$nick
}else {
    //si no existe le mando otra vez a la principal con la variable = “si”
    header("Location: index.php?errorusuario=si");
}
?>
```

Código para SQL

```
<?
$nick = $_POST['usuario'];
$password = $_POST['password'];

$conexion=odbc_connect("SQL","","");
$query="select Nick,Password from Usuario where Nick='$nick' and
        Password='$password';";
$sel= odbc_do($conexion,$query);
$nom= odbc_result($sel,"Nick");
$pass= odbc_result($sel,"Password");

//vemos si el usuario y contraseña es válido
if ($nick==$nom && $password==$pass){
   //usuario y contraseña válidos
   //defino una sesion y guardo datos
        session_start();
        $_SESSION["autentificado"]= "SI";
        $_SESSION["usuario"]=$nick;
        header ("Location: Jeadeca.php");
}else {
   //si no existe le mando otra vez a la portada
   header("Location: index.php?errorusuario=si");
}
?>
```

Como se muestra en el código recibe los datos del formulario en 2 variables, el contenido de esas variables se va a comparar con el contenido de otras variables que mandan traer la información de la base de datos, si concuerdan los datos entonces da entrada a la página ***jeadeca.php***; si los datos no concuerdan regresa a la pagina ***index.php*** con la variable y despliega en la primera celda del formulario el texto de error sobre el fondo rojo.

Además de eso, crea 2 variables de sesión, la primera con el nombre de *“Autentificado*” y con un valor de “*si*”, y la segunda con el nombre de “*Usuario*” y su valor será el ***nick*** que tiene el Usuario registrado. Por supuesto que estas variables se crean en el caso de que el usuario ya este dado de alta.

En otro enlace es el que se encuentra debajo del botón de envío del formulario, y manda hacia la página ***registrar.php***, únicamente se selecciona el texto, que en este caso es “*Si quieres darte de alta da clic sobre estas letras*” y en las propiedades en la opción de “Vínculo”, se selecciona la página a la que será vinculada.

Antes de hacer los dos enlaces se tienen que crear las páginas a las que se direccionarán, o sea, ***registrar.php*** y ***jeadeca.php***.

4.2.1.3. registrar.php

La página ***registrar.php*** será una página específica para registrar nuevos usuarios en la página. En esta página se le pedirán información personal y se guardará en la base de datos para después poder acceder con los datos registrados a otras páginas.

Y como en todas las demás páginas llevará el logo del encabezado, la descripción o referencia a lo que se dedica la empresa, y la regla horizontal.

Ya teniendo eso quedará el siguiente diseño:

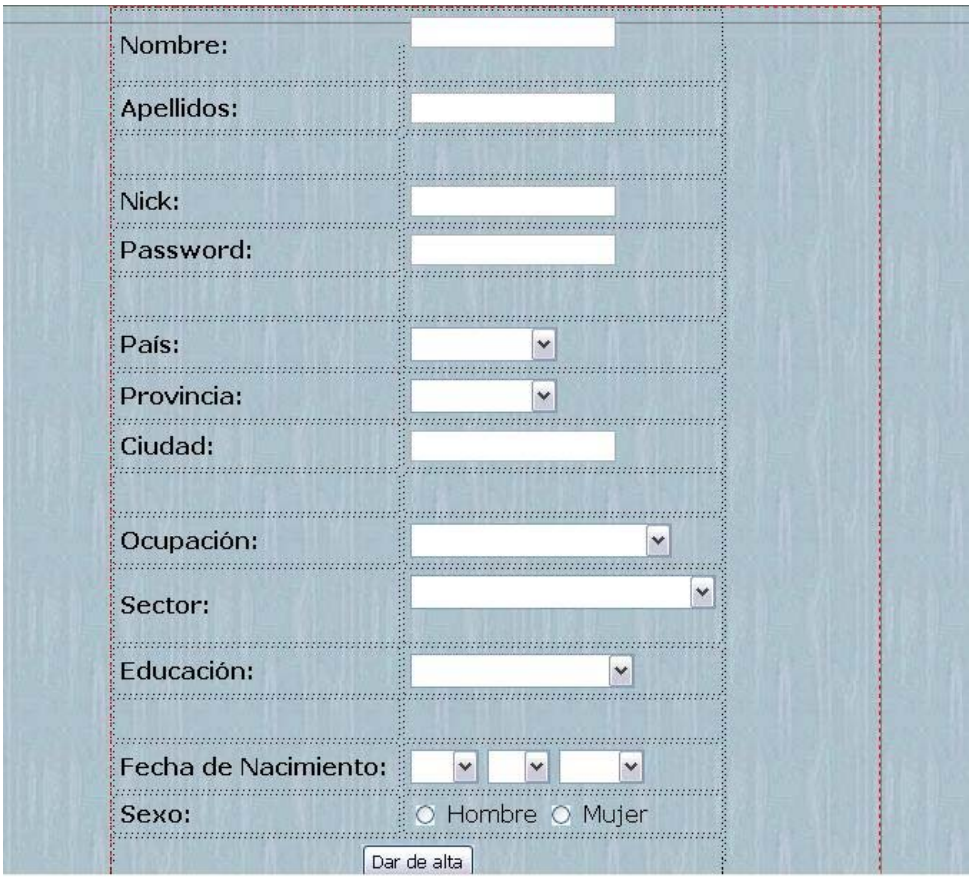


Figura 24. Diseño de *registrar.php*

Como se ve en la imagen, dentro de un formulario está una tabla de 17 filas y 2 columnas, que como en la pagina *index.php* se utilizó una columna para texto y la otra para que el usuario escribiera en una caja de texto. Ahora además de cajas de texto se van a utilizar campos de *lista/menú* y *botón de opción*.

Para insertar los campos de *lista/menú* es en menú *INSERTAR—FORMULARIOS—LISTA/MENÚ*, y para el *botón de opción* en menú *INSERTAR—FORMULARIOS—BOTÓN DE OPCIÓN*.

Y se dejan el texto y los campos como se ven en la imagen.

Como estamos utilizando páginas dinámicas hay cierta información que se nos facilitará tenerla en la base de datos. Y uno de esos casos es en el campo de *PAÍS*, que teniendo en la base de datos los países y sus provincias respectivas haremos que cuando se seleccione un país en el

campo, automáticamente solo despliegue en el campo de *PROVINCIAS* solo las referentes al *PAÍS* elegido.

Esto se llama también como ***CAMPOS DINÁMICOS***.

Para esto tendremos que proseguir con el código PHP para lograrlo, y es el siguiente:

Inmediatamente después de la indicación en el código del *BODY* se declaran unas variables que servirán para la conexión a la base de datos.

Código para Access

```
<body>
<?
$conexion=odbc_connect("Access","","");
$select="select* from Pais order by NomPais";
$select2="select* from Provincia order by idPais";
?>
```

Código para SQL

```
<body>
<?
$conexion=odbc_connect("SQL","","");
$select="select* from Pais order by NomPais";
$select2="select* from Provincia order by idPais";
?>
```

Como en la página *control.php* se hace una variable de conexión, que llama a la base de datos que esta determinada por el ODBC de Windows y que tiene nombre "*Access*" y que por consecuencia es la que estamos utilizando.

Después se crean 2 variables que hacen selects sobre el contenido de los nombre de los países y provincias.

Después en el campo de *PAÍS* se pone el siguiente código:

Código para Access y SQL

```
<select name="pais" onChange="incluir(this.form.pais[selectedIndex].value);">
        <option value="-2">- seleccionar -</option>
        <?php
            $resultado=odbc_do($conexion,$select);
            while(odbc_fetch_row($resultado)){
                if ($row=odbc_result($resultado,"NomPais"))
                {
                    $rowid=odbc_result($resultado,"idPais");
                    echo "<option value=".$rowid.">".$row."</option>";
                }
            }
        ?>
</select>
```

Este código va en el campo de *lista/menú* de *PAÍS* y hace lo siguiente:

- En la primera línea el campo tienen el nombre de *país* y en el evento onChange hace que detecte el valor del formulario del campo *PAÍS* con una función llamada *incluir*[2]
- En la segunda línea delimita un valor como -2 y que aparezca el texto "- seleccionar –". Esto también se puede hacer desde las propiedades del campo y en Valores de Lista. Esto se hace para qué el primer valor que despliegue el campo sea "- *seleccionar –*" por eso se le pone el valor de **-2**.

[2] La función se verá en la página 107

Figura 25. Ventana de valores de la lista “pais”

- Después el código PHP, iniciado desde los signos **<?** Manda llamar los resultados del *select* de los países y los guarda en una variable “*$resultado*”.
- Luego dentro de un ciclo *WHILE* checaremos que mientras tenga valores la variable *$resultado* se guardará en una variable llamada *$row* los resultados del campo del nombre del país en la base de datos (o sea *NomPais*) y en otra variable llamada *$rowid* los resultados del campo del id del país en la base de datos (o sea *idPais*).
- Por último imprimirá dentro de la lista de valores del campo de *LISTA/MENÚ* el nombre del país como Etiqueta de elemento y el id del país como Valor.

Ya entonces se tienen los valores dinámicos en el campo de País, ahora prosigue el de Provincia y es con el siguiente código después de que termine la parte del formulario:

Código para Access y SQL

```
</form>
  <script languaje="jscript">
        function valores(campo1,campo2){
                this.campo1=campo1;
                this.campo2=campo2;
        }
<?
$resultado2=odbc_do($conexión,$select2);
$indice=0;
$pais=0;
while(odbc_fetch_row($resultado2)){
        if ($rowidP=odbc_result($resultado2,"idPais"))    {
                if($pais!=$rowidP){
                        $indice=0;
                        $pais=$rowidP;
                        echo "var mimatriz".$pais."= new Array();\n";
                }
        }
        $nrowid=odbc_result($resultado2,"idProvincia");
        $nrow=odbc_result($resultado2,"NomProvincia");
        Echo"mimatriz".$pais."[".$indice."]=new valores('".$nrow."','".$nrowid."');\n";
        $indice= $indice+1;
}
?>
```

```
var i;
function incluir(array){
        clear();
        array=eval("mimatriz" + array);
        for(i=0; i<array.length; i++){
                var objeto=new Option(array[i].campo1, array[i].campo2);
                main.provincia.options[i]=objeto;
        }
        main.provincia.disabled=false;
}

function clear(){
        main.provincia.length=0;
        main.provincia.disabled=true;
}
main.provincia.disabled=true;

</script>
```

La descripción de este código es la siguiente:

- Se crean dentro del script unas funciones, y es dentro del script de java para que así el programa las detecte con el evento *OnChange*
- La primera función se llama *Valores* y sirve para guardar 2 valores referentes al id de la provincia y al nombre de la provincia.
- Después hay un código similar al utilizado en el campo de *PAÍS*, en el cual se asignan valores de la base de datos a variables, pero en ves de guardarse en la lista de valores se va a guardar en *ARRAY*
- La segunda función se llama *Incluir* y sirve para evaluar el *ARRAY*, y si existe contenido en el *ARRAY*, asigna los valores dinámicamente por medio del script hacia el campo de *PROVINCIA*.
- La última función se llama *Clear* y sirve para deshabilitar el campo de *PROVINCIA* y ponerla en blanco.

Ya con los campos dinámicos ahora se prosigue a colocar los valores de los otros campos de *LISTA/MENÚ*. Como no necesitan ser dinámicos solo se ponen sus valores en sus respectivos *Valores de lista* en las propiedades de cada uno.

Recordando que los nombre de los campos y listas/menú tendrán los mismos nombres que sus textos de lado izquierdo. El único caso en donde va a ser diferente es en el de fecha, ya que el primer campo de lista/menú su nombre será "*dia*", el segundo será "*mes*", y el tercero será "ano*"*, y es *ano*, porque los lenguajes son en ingles y no reconocen la **ñ**.

Como es lógico los valores comienzan desde 1 hasta que terminen los elementos, pero en el campo de *año* se tendrá que poner su *Etiqueta de elemento* y su *valor* iguales, o sea, que el año sea ***1980*** se pone como Etiqueta y como Valor.

Figura 26. Ventana de valores de la lista de "año"

Esto es porque el formulario no manda la etiqueta, sino el valor de esa etiqueta, y al recibir la otra página los datos del formulario no coincidirán los números de los elementos con sus valores. De igual manera si se quiere enviar algún texto, en el valor de la etiqueta se pondrá el texto, ya que ***el valor*** es lo que manda el formulario.

Para los *botones de opción* se les pone a los 2 el mismo nombre, en este caso se llaman "*sexo*", y en su valor uno será "Hombre" y el otro "Mujer". Ya con eso únicamente se podrá mandar uno de los valores por medio del formulario.

Ya teniendo todo listo sólo falta cambiar las propiedades del formulario, el cuál va a tener el nombre de *"main"*, la acción será a ***insertar.php***, y el destino será *_self*.

4.2.1.4. insertar.php

De la misma manera que se creó la página de *control.php*, ahora se creará una página que hará que la información que se insertó en la página *registrar.php* se inserte en la base de datos.

El código es similar a los anteriores, se crean variables a las cuales les asignamos los valores provenientes del formulario de la página *registrar.php* y de la base de datos. La diferencia de la de *control.php* es que aquí si vamos a poner algo de diseño, por lo tanto el código va a ir después del inicio del *BODY* y será el siguiente:

Código para Access

```
<body>
<?
$nombre = $_POST['nombre'];
$apellidos = $_POST['apellidos'];
$nick = $_POST['nick'];
$password = $_POST['password'];
$pais = $_POST['pais'];
$provincia = $_POST['provincia'];
$ciudad = $_POST['ciudad'];
$ocupacion = $_POST['ocupacion'];
$sector = $_POST['sector'];
$educacion = $_POST['educacion'];
$sexo = $_POST['sexo'];
$dia= $_POST['dia'];
$mes= $_POST['mes'];
$ano= $_POST['ano'];
$nacimiento= "$dia/$mes/$ano";
//Conexión con la base
$conexion=odbc_connect("access","","");
//Ejecución de la sentencia SQL
$qpais="select NomPais from Pais where idPais=$pais";
$qprovincia="select NomProvincia from Provincia where idProvincia=$provincia";
$npais= odbc_do($conexion,$qpais);
$Pais= odbc_result($npais,"NomPais");
$nprovincia= odbc_do($conexion,$qprovincia);
$Provincia= odbc_result($nprovincia,"NomProvincia");
$insertar="insert into usuario (NomUsuario,Apellidos,Nick,Password,Pais, Provincia,
        Ciudad,Ocupacion,Sector,Educacion,Nacimiento,Sexo)
        values('$nombre','$apellidos','$nick','$password','$Pais','$Provincia','$ciudad',
        '$ocupacion','$sector','$educacion','$nacimiento','$sexo');";
$resultado=odbc_do($conexion,$insertar);
if ($resultado) {
?>
```

<body>

Código para SQL

```
<?
$nombre = $_POST['nombre'];
$apellidos = $_POST['apellidos'];
$nick = $_POST['nick'];
$password = $_POST['password'];
$pais = $_POST['pais'];
$provincia = $_POST['provincia'];
$ciudad = $_POST['ciudad'];
$ocupacion = $_POST['ocupacion'];
$sector = $_POST['sector'];
$educacion = $_POST['educacion'];
$sexo = $_POST['sexo'];
$dia= $_POST['dia'];
$mes= $_POST['mes'];
$ano= $_POST['ano'];
$nacimiento= "$dia/$mes/$ano";
//Conexión con la base
$conexion=odbc_connect("SQL","","");
//Ejecucion de la sentencia SQL
$N=0; //variable para utilizar como contador
$qpais="select NomPais from Pais where idPais=$pais";
$qprovincia="select NomProvincia from Provincia where idProvincia=$provincia";
$npais= odbc_do($conexion,$qpais);
$Pais= odbc_result($npais,"NomPais");
$nprovincia= odbc_do($conexion,$qprovincia);
$Provincia= odbc_result($nprovincia,"NomProvincia");
$id="select idUsuario from Usuario";
$Mid=odbc_do($conexion,$id);
//aumento en 1 el contador dependiendo de los ID existentes
while(odbc_fetch_row($Mid)){
        $N=$N+1;
}
$insertar="insert into usuario(idUsuario,NomUsuario,Apellidos,Nick,Password,Pais,
            Provincia,Ciudad,Ocupacion,Sector,Educacion,Nacimiento,Sexo) values
            ('$N','$nombre','$apellidos','$nick','$password','$Pais','$Provincia','$ciudad',
            '$ocupacion','$sector','$educacion','$nacimiento','$sexo');";
$resultado=odbc_do($conexion,$insertar);
if ($resultado) {
?>
```

Código para Access y SQL

```
<h1><div align="center">Te haz dado de alta satisfactoriamente</div></h1>
<div align="center"><a href="jeadeca.php">Regresa para ingresar tus datos</a></div>
<?
} else {
        echo "Hubo un problema y no se insertó nada";
}
?>
</body>
```

La diferencia de los códigos es que con Access se hizo la tabla en los id como ***Autonumérico***, y en SQL como ***entero***. Por lo tanto en la tabla de SQL se tiene que ir creando el número consecutivo para cada usuario.

De tal manera que cada uno de los valores los insertará en la base de datos, y ya insertados desplegará en la pantalla lo siguiente

Figura 27. Diseño de *insertar.php* cuando ya insertó la información

Y sino aparecerá una leyenda que “HUBO UN PROBLEMA Y NO SE INSERTÓ NADA”.

Regresando a la página de *index.php*, otros vínculos que se habían puesto son los de los botones que muestran las listas de los productos y servicios.

El botón flash de *Productos* despliega un texto y otro botón flash que se llama *LISTA*. Este botón vinculará la página hacia una nueva que se llamará ***listaprod.php***.

4.2.1.5. listaprod.php

Esta página estará encargada de enlistar los productos que se tienen en la base de datos ordenados por su clasificación. Tendrá el siguiente funcionamiento:

- En un formulario se encontrará un campo *MENU/LISTA* que contendrá la clasificación de los productos y tendrá el nombre de “clasif”.
- Debajo y dentro del formulario se encontrará un campo oculto, esté contendrá en sus propiedades las siguientes características:
    - Nombre: *oculto*
    - Valor: *producto*

  Esto porque luego utilizaremos el valor para diferenciar el campo.
- Debajo y también dentro del formulario se encontrará un botón de envío.
- Una tabla que se creará dependiendo de los valores que contenga la tabla *Productos* en la base de datos en la clasificación que se haya seleccionado.
- Y un botón flash que regrese a la página anterior.

Tendrá el mismo encabezado que la página *index.php*, y quedará el diseño de la siguiente forma:

Figura 28. Diseño de *listaprod.php*

En el diseño se ve que en un formulario se encuentra el campo de *MENÚ/LISTA*, el cuál va a tener un código parecido al de la página de *registrar.php*, seleccionando de la base de datos la información que contendrá el campo. Quedará de la siguiente manera:

En la parte del *BODY* se declararán las siguientes variables de conexión:

```
<?
$conexion=odbc_connect("Access","","");
$select="select * from Clasificación order by NomClasif";
?>
```

Después se en la parte del campo *LISTA/MENU* se pone el siguiente código:

```
<form action="mostrarlista.php" method="post" name="main" id="main">
<p>
    <select name="clasif" id="clasif"
        onchange="incluir(this.form.clasificacion[selectedIndex].value);">
     <option value="-2" selected>- seleccionar -</option>
     <?php
            $resultado=odbc_do($conexion,$select);
```

```
            while(odbc_fetch_row($resultado)){
                if ($row=odbc_result($resultado,"NomClasif")){
                    $rowid=odbc_result($resultado,"idClasif");
                    echo "<option value=".$rowid.">".$row."</option>";
                    $cont=$rowid;
                }
            }
        ?>
    </select>
```

Este código, que es similar al de la página de registro, cambia el nombre del campo *LISTA/MENU*, y en el código cambian las referencias de los nombres de los campos de la tabla *Productos* en la base de datos.

Después de que termina el código del formulario se pone el siguiente código:

```
</form>
<?
    if ($_GET["mandar"]>"0"){
        $clasificacion= $_GET["mandar"];
        $conexion=odbc_connect("access","","");
        $num=0;
        $query="select * from Producto where idClasif=$clasificacion order by
                NomProducto";
        $rquery=odbc_do($conexion,$query);
        while (odbc_fetch_row($rquery)){
            $Prod[$num]=odbc_result($rquery,"NomProducto");
            $Precio[$num]=odbc_result($rquery,"Precio");
            $num=$num+1;
        }
    }
?>
```

Este código recibe una variable por URL llamada “*mandar*”, y si el valor de esta variable es mayor a cero comienza a llenar en unas variables de tipo *Array* la información contenida en la tabla *Productos*. Estas variables se llamarán *$Prod* y *$Precio*.

¿Pero de donde va a provenir esa variable por URL? Principalmente se va a agregar dentro del formulario el campo oculto, y para insertarlo lo vamos a encontrar en el menú en *INSERTAR—FORMULARIO—CAMPO OCULTO*. Como se había mencionado, se pondrán sus propiedades el valor de *producto*, esto para saber que proviene de esta página *listaprod.php*. Después el botón envío va a mandar la información de los campos del formulario hacia una página llamada ***mostrarlista.php***.

4.2.1.6. mostrarlista.php

Esta página no tendrá diseño alguno, solo código. Va a servir para controlar los datos que tengan los campos ocultos que se manden por parte de la página *listaprod.php*, y mandar una variable por URL regresandola a la misma página de donde provino (o sea *listaprod.php*) y también nos servirá para la página que mostrará el listado de los servicios que se tienen.

El código de la página ***mostrarlista.php*** será el siguiente:

```
<?
	$Who = $_POST['oculto'];
	if ($Who=="producto") {
		$clasif= $_POST['clasif'];
		if ($clasif>0){
			header("Location: listaprod.php?mandar=$clasif");
		}else{
			header ("Location: listaprod.php");
		}
	}
	if ($Who=="servicio") {
		$clasif= $_POST['clasif'];
		if ($clasif>0){
			header("Location: listaserv.php?mandar=$clasif");
		}else{
			header ("Location: listaserv.php");
		}
```

```
        }
?>
```

El código primero recibe el valor del *Campo Oculto* y lo guarda en una variable *$Who*, después con unas condiciones verifica el contenido de la variable, si la variable tiene como valor *"producto"* entonces guarda el valor del campo *MENÚ/LISTA* de la clasificación del producto en otra variable. Si el valor de esta variable es mayor a cero, devuelve el valor por la variable URL "*mandar*" a la página de *listaprod.php*. Pero si el valor de la variable de clasificación no satisface la condición *IF* entonces solo nos regresará a la página de *listaprod.php*.

En el código también aparece una condición *IF* que es para cuando el valor de la variable *$Who* contiene como valor "*servicio*", esto es cuando en una página que nos propondremos a crear mande la información por medio del *Campo Oculto*.

Todo este rollo de crear páginas y de recibir variables de tipo *Array* nos servirá para crear una tabla, que para reconocerla la llamaremos *Tabla Virtual*. Esta se va a crear dinámicamente dependiendo de los campos que contenga la tabla de la base de datos.

Para poder crear esta tabla se pondrá el siguiente código en una capa.

```
<div id="Layer5" style="position:absolute; left:304px; top:244px; width:253px; height:303px; z-index:6; visibility: visible;">
  <div align="center"><span class="Estilo2"><strong><span class="Estilo11">Lista de productos <br>
    <br>
    <br>
  </span></strong></span> </div>
  <table align=center border=1 cellpadding="1">
    <?
        echo "<tr><td align=\"center\" bgcolor=#cccccc><span class=\"cabezatablas\">PRODUCTOS</td><td align=\"center\" bgcolor=#cccccc><span class=\"cabezatablas\">PRECIOS</td></tr>\n";

    for ($i=0;$i<=$num-1;$i++){
```

```
      echo "<tr><td>$Prod[$i]</td><td align=\"center\">$Precio[$i]</td></tr>\n";
    }
?>
  </table>
</div>
```

El código va creando celdas dependiendo de la cantidad de elementos que contengan las variables *Array* mientras coloca la información de cada una de estas variables.

4.2.1.7. listaserv.php

La otra página que se había mencionado para el listado de los servicios se llamará ***listaserv.php*****,** que nos servirá, como ya se había mencionado, para mostrar la lista de servicios que se tienen en la base de datos ordenados por clasificación de servicio. Esta página será activada por un botón llamado *Mantenimiento* que se visualizará al pulsar el botón flash *Servicios* de la página index.php, y será de diseño exactamente igual a la de *listaprod.php*, lo único que cambia es el código para las variables de conexión, el valor del *Campo Oculto* que será "servicio" y los textos que llevará la tabla Virtual.

Variables de conexión:

```
<?
$conexion=odbc_connect("Access","","");
$select="select * from ClasSer order by NomClasSer";
?>
<form action="mostrarlista.php" method="post" name="main" id="main">
   <p>
     <select name="clasif" id="clasif">
       <option value="-2" selected>- seleccionar -</option>
       <?php
              $resultado=odbc_do($conexion,$select);
              while(odbc_fetch_row($resultado)){
```

```
                    if ($row=odbc_result($resultado,"NomClasSer")){
                        $rowid=odbc_result($resultado,"idClasSer");
                        echo "<option value=".$rowid.">".$row."</option>";
                        $cont=$rowid;
                    }
                }
        ?>
      </select>
      <br>
      <input name="oculto" type="hidden" id="oculto" value="servicio">
    </p>
    <p align="center">
      <input type="submit" name="Submit" value="Ver">
</p>
  </form>

<? if ($_GET["mandar"]>"0"){
        $clasificacion= $_GET["mandar"];
        $conexion=odbc_connect("access","","");
        $num=0;
        $query="select * from Servicio where idClasSer=$clasificacion order by
                NomServicio";
        $rquery=odbc_do($conexion,$query);
        while (odbc_fetch_row($rquery)){
                $Serv[$num]=odbc_result($rquery,"NomServicio");
                $Precio[$num]=odbc_result($rquery,"Precio");
                $num=$num+1;
                }
        }
?>
```

Ya que terminamos estas páginas, se ha de pensar, ¿y con qué propósito se hicieron? El propósito es dar permiso de entrar libremente a unas páginas y de permitir entrar a otra de las páginas principales, la cuál solo los usuarios registrados podrán acceder. La diferencia de esta nueva página y la de *index.php* es que se podrán hacer compras de productos desde la página. Esta manera el administrador de la página va a tener

acceso a la base de datos y checar qué usuario requiere qué cantidad de productos sin necesidad de comunicarse directamente con él. La forma es muy sencilla, es como un carrito de compras, se selecciona el producto, la cantidad y se va ir llenando un pedido de productos.

El administrador le comunicará directamente al encargado de ventas de la empresa el pedido, y se le enviará directamente al domicilio del usuario.

Ya se podrían plantear ciertas medidas de pago, ya sea al recibir el producto, o antes de enviarlo comunicarse con el usuario para que haga un depósito electrónico, etc.

Entonces ya que se detecte el usuario pasará a una página llamada ***jeadeca.php*** que es la que mandaba la pagina *control.php*. Esta página deberá de estará protegida con las variables de sesión antes creadas, por eso tendremos que crear una página que detecte que estén activas esas variables. Le llamaremos ***seguridad.php***.

4.2.1.8. seguridad.php

Esta página contendrá solo un código de chequeo, y si es que están activas las variables de sesion permite permanecer en la página, sino hace que regrese a la página de *index.php*. Su código es el siguiente:

```
<?
//Inicio la sesión
session_start();
//COMPRUEBA QUE EL USUARIO ESTA AUTENTIFICADO
if ($_SESSION["autentificado"] != "SI") {
    //si no existe, envío a la página de autentificación
    header("Location: index.php");
    //además salgo de este script
    exit();}
?>
```

Ya con estas partes de seguridad esenciales nos propondremos a crear la página de ***jeadeca.php***.

4.2.1.9. jeadeca.php

Esta página servirá como página principal para los usuarios que ya se registraron y tienen una sesion abierta. El formato es casi idéntico a la de *index.php*, la única cosa que se le va a quitar es el formulario de ingreso. Además de eso se le agregarán nuevos elementos:

1. Texto de bienvenida al Usuario registrado
2. Botón para cerrar la sesión del Usuario registrado
3. Botón para ingresar a la página de compras

Figura 29. Diseño de *jeadeca.php*

Al inicio de la página vamos a incluir el contenido de la página *"seguridad.php"* para lo ya mencionado, y es con el siguiente código

```
<? include ("seguridad.php"); ?>
```

Para el texto de bienvenida se va agregar una parte de código en una capa y será la siguiente:

```
<div id="Usuario" style="position:absolute; left:55px; top:139px; width:334px; height:24px; z-index:13">
<span class="Estilo7">Bienvenido:
```

```
<?
echo $_SESSION["usuario"];
 ?>
</div>
```

Como se ve en el código se asigna el nombre a la capa como "*Usuario*", después se le escribe la palabra "*Bienvenido:*" y el código PHP, el cual escribe el valor de la variable de sesión *Usuario* (que es el *Nick*) y listo.

El botón de cierre de sesión sirve para que otra persona no pueda entrar con su cuenta de usuario en su ventana, o simplemente para evitar problemas de seguridad. Esto se utiliza mucho en páginas como **HOTMAIL** y similares.

Utilizaremos un botón de formulario y en sus propiedades, su nombre será "*salirsesion*" y en la parte de Acción seleccionaremos *"Ninguno*". Después en los *Comportamientos* en el panel de *Etiqueta* le damos un evento de ***OnClick*** de ***Ir a URL***. La URL será otra página alterna que crearemos, se llamará ***salirsesion.php***.

4.2.1.10. salirsesion.php

En esta página sólo contendrá un código con el cual haremos que se destruya la sesión y que mande a la página de *index.php*. El código es el siguiente:

```
<?
session_start();
session_destroy();
header ("Location: index.php");
?>
```

Ahora sólo falta el botón que te manda a la página de compras. Será en un botón flash la cuál tendrá como Vinculo la página ***comprar.php***.

Esta página de comprar va a tener 2 páginas incluidas que será la de *seguridad.php* y otra que vamos a crear ahora que se va a llamar ***mostrarcarrito.php***.

4.2.1.11. mostrarcarrito.php

Esta página solo llevará un código que buscará los elementos que están agregados en una tabla de la base de datos llamada *CompraVirtual*, que es en la cuál se van a guardar todas las compras virtuales que van a ser los usuarios. Para esto se buscan únicamente los resultados que sean del día en que se esta consultando la página y por parte del usuario que esta haciendo la consulta, y estos datos los mandarán llamar en la página de compra. Quedará el código de la siguiente manera:

```
<? include("seguridad.php");


$fecha2=date("Y-m-d 00:00:00");
$conexion=odbc_connect("access","","");
$nom=$_SESSION["usuario"];
$num=0;
$query="select idCompra, NomProducto, Cantidad, Fecha from CompraVirtual where
    NomUsuario='$nom' order by NomProducto";
$rquery=odbc_do($conexion,$query);
while (odbc_fetch_row($rquery)){
        $date=odbc_result($rquery,"Fecha");
        $cant=odbc_result($rquery,"cantidad");
        if (($date==$fecha2) && ($cant>0)){
                $idCompra[$num]=odbc_result($rquery,"idCompra");
                $Producto[$num]=odbc_result($rquery,"NomProducto");
                $cantidad[$num]=odbc_result($rquery,"cantidad");
                $num=$num+1;
        }
}
?>
```

Los valores los guardará en 3 *Arrays* que se van a utilizar mas adelante.

Empezaremos a crear la pagina *comprar.php.*

4.2.1.12. comprar.php

Esta página servirá para comprar productos desde la página, y mostrar estos productos que se llevan comprados en una lista. Tendrá el mismo encabezado que todas, tendrá un comentario similar al de bienvenida como en la página *jeadeca.php*, tendrá un formulario con una tabla fija insertada en la que estarán los elementos a seleccionar como son: Categoría del producto, Nombre del producto y Cantidad del producto. Además de un botón de envío.

Habrá una capa invisible, la cuál se hará visible en el caso de que los datos del formulario estén incompletos o erróneos. Además pondremos una tabla que se generará dependiendo de los elementos que estén insertados en la tabla *CompraVirtual* de la base de datos.

Y por último un botón flash que mandará hacia otra página que nos servirá para quitar productos que se hayan insertado.

El diseño será de la siguiente manera:

Figura 30. Diseño de *comprar.php*

Para poner el mensaje de "*selecciona tus productos*" y la variable de sesión de *Usuario* es igual que en el de bienvenida de *jeadeca.php*.

Para poner el formulario con la tabla y los elementos es igual que en la de *registrar.php* nada más acomodando el código de la siguiente manera para el campo de *CATEGORÍA DEL PRODUCTO* como si fuera la de *PAIS*:

```
<select name="clasificacion" id="clasificacion" onchange=
  "incluir(this.form.clasificacion[selectedIndex].value);">
      <option value="-2" selected>- seleccionar -</option>
      <?php
        $resultado=odbc_do($conexion,$select);
        while(odbc_fetch_row($resultado)){
              if ($row=odbc_result($resultado,"NomClasif")){
                    $rowid=odbc_result($resultado,"idClasif");
                    echo "<option value=".$rowid.">".$row."</option>";
              }
        }
      ?>
</select>
```

Y acomodando el siguiente como si fuera la parte de de *PROVINCIA*:

```
</form>
  <script languaje="jscript">
        function valores(campo1,campo2){
              this.campo1=campo1;
              this.campo2=campo2;
        }
        <?
        $resultado2=odbc_do($conexion,$select2);
        $indice=0;
        $clasif=0;
        while(odbc_fetch_row($resultado2)){
              if ($rowidC=odbc_result($resultado2,"idClasif")) {
                    if($clasif!=$rowidC){
                          $indice=0;
```

```
                        $clasif=$rowidC;
                        echo "var mimatriz".$clasif."= new Array();\n";
                }
        }
        $nrowid=odbc_result($resultado2,"idProducto");
        $nrow=odbc_result($resultado2,"NomProducto");
        echo "mimatriz".$clasif."[".$indice."]=new
                valores('".$nrow."','".$nrowid."');\n";
        $indice= $indice+1;
}
?>
var i;
function incluir(array){
        clear();
        array=eval("mimatriz" + array);
        for(i=0; i<array.length; i++){
                var objeto=new Option(array[i].campo1, array[i].campo2);
                main.producto.options[i]=objeto;
        }
        main.producto.disabled=false;
}

function clear(){
        main.producto.length=0;
        main.producto.disabled=true;
}
main.producto.disabled=true;
</script>
```

Para la capa que será invisible hasta que ocurra un error primero se genera una capa vacía, y dentro de ella se agregará en la parte de código lo siguiente:

```
<div id="Layer4" style="position:absolute; left:17px; top:238px; width:155px; height:83px; z-index:5; visibility: visible;">
<? if ($_GET["errorproductos"]=="si"){ ?>
        <div align="center"><span style="color:#CC3300; font-family: Georgia, &amp;quot;Times New Roman&amp;quot;, Times, serif; font-size: 16px;"><b>Faltan datos. Comprueba que hayas seleccionado un producto de una categoría y con una cantidad diferente a cero </b></span>
```

```
    <? } ?>
    </div>
</div>
```

Esto hace que cuando mande los datos del formulario a una página que se llamará ***controlcarrito.php***. Esta verificará que estén o no completos, en el caso de que no estén completos, manda una variable por URL con nombre *errorproductos* con un valor de "*si*" de nuevo a la página *comprar.php* y en el código de la capa se tendrá una condición, si es la variable tiene el valor despliega el texto como aparece en la imagen.

4.2.1.13. controlcarrito.php

El código que se va a utilizar en la página ***controlcarrito.php*** nos va a servir para tener un control de los datos que nos mande el formulario. Como ya se vio en la capa de despliegue de error, si no están completos, *controlcarrito.php* nos regresa una variable de error; pero si están completos y correctos los datos nos va a insertar estos datos a la tabla de *CompraVirtual* (ya mencionada anteriormente).

En el caso de que alguno de los productos ya hayan sido insertados, solamente se va a hacer un update sobre las cantidades de los productos. Para que se entienda mejor, este es el código que se estará utilizando.

```
<? include("seguridad.php");
$clasificacion = $_POST['clasificacion'];
$producto = $_POST['producto'];
$cantidad = $_POST['cantidad'];
$fecha=date("d/m/Y");
$fecha2=date("Y-m-d 00:00:00");

if ($clasificacion!=-2){
        $conexion=odbc_connect("access","","");
        $nom=$_SESSION["usuario"];

        $qclasif="select NomClasif from Clasificación where idClasif=$clasificacion";
        $qproducto="select NomProducto from Producto where idProducto=$producto";
```

```
        $nclasif= odbc_do($conexion,$qclasif);
        $Clasif= odbc_result($nclasif,"NomClasif");
        $nproducto= odbc_do($conexion,$qproducto);
        $Producto= odbc_result($nproducto,"NomProducto");
}
if (($clasificacion!=-2) && ($cantidad>0)){
        $query="select idCompra, Fecha from CompraVirtual where
            NomProducto='$Producto' and NomUsuario='$nom'";
        $rquery=odbc_do($conexion,$query);
        $id=0;
        while(odbc_fetch_row($rquery)){
                if ($fecha2==$date=odbc_result($rquery,"fecha")){
                        $id=odbc_result($rquery,"idCompra");
                        $update="update CompraVirtual set cantidad=cantidad+$cantidad
                                where idCompra=$id";
                        $reupdate=odbc_do($conexion,$update);
                        header ("Location: comprar.php");
                }
        }
        if ($id==0){
                $insert="insert into CompraVirtual
                    (NomUsuario,NomClasificacion,idProducto,NomProducto,
                    Cantidad,Fecha) values ('$nom','$Clasif','$producto',
                    '$Producto','$cantidad','$fecha')";
                $resinsert=odbc_do($conexion,$insert);
                header ("Location: comprar.php");
        }
} else {
        header("Location: comprar.php?errorproductos=si");
}
?>
```

Lo único que podría ser confuso aquí es porqué se utilizaron 2 variables de fecha (***$fecha*** y ***$fecha2***) y esto es porque la de *$fecha2* se va a utilizar para comparar los resultados que nos arroja en un select el lenguaje SQL con la fecha actual pero con el mismo formato, por eso se asigna el formato ***date("Y-m-d 00:00:00")***. Y la de *$fecha* es para insertar

directamente a la tabla, ya que de esta manera no tiene ningún problema en reconocer el lenguaje SQL ***date("d/m/Y")***.

Ya sabiendo esto, volvemos donde nos habíamos quedado en la página de *comprar.php.*

Prosigue colocar la tabla que se va a ir generando dependiendo de los elementos que tenga la tabla *CompraVirtual* en la base de datos y que sean del Usuario activo, y en la fecha en que se esta visitando la página.

De nuevo se colocará el código dentro de una capa a la cuál dentro solo se escribirá en la parte de diseño un título "***Lista de productos que llevas***", después nos vamos a la parte de código e insertaremos el siguiente código:

```
<div id="Layer5" style="position:absolute; left:701px; top:213px; width:253px; height:160px; z-index:6; visibility: visible;">
  <div align="center"><span class="Estilo5"><strong><span class="Estilo11">Lista de
     productos <br>
     que llevas<br>
     <br>
     </span></strong></span>
  </div>
  <table align=center border=1 cellpadding="1">
  <?
      echo "<tr><td align=\"center\" bgcolor=#cccccc><span
class=\"cabezatablas\">PRODUCTOS</td><td align=\"center\" bgcolor=#cccccc><span
class=\"cabezatablas\">CANTIDADES</td></tr>\n";

   for ($i=0;$i<=$num-1;$i++){
     echo "<tr><td>$Producto[$i]</td><td align=\"center\">$cantidad[$i]</td>
          </tr>\n";
   }
  ?>
 </table>
</div>
```

Al crear la tabla el código se genera igual que cuando la insertas desde la parte de diseño, solamente que aquí queremos que vaya generando las celdas conforme a los valores que nos brindará la base de datos. Primero se genera un encabezado de la tabla, que va a permanecer visible aunque no haya valores en la tabla. Después en un ciclo se van a ir insertando los valores dependiendo de los valores que contengan los *Arrays* en la página *mostrarcarrito.php*[3]. El del nombre del producto del lado izquierdo y el de la cantidad del lado derecho. Ya corriendo la página se verá de la siguiente manera:

Tabla sin datos *Tabla con datos*

Figura 31. Diseño de las tablas virtuales

Por último se crean 2 botones flash, el primero con un vínculo a una página llamada ***borrar.php*** y su destino *_self* como todas las demás, y el otro con un vínculo hacia la página *jeadeca.php*.

Ahora sigue crear la página de ***borrar.php***.

[3] Ver código de la página 123

4.2.1.14. borrar.php

Esta página nos va a servir para que en el caso de que el usuario se hubiera equivocado en las cantidades que compró, o en algún producto, se borren de la lista, y por consecuencia de la base de datos.

El diseño es prácticamente igual al de la página de *comprar.php*, solo se quitan algunos campos en la tabla del formulario y se cambia una parte del código. Quedará el diseño de la siguiente manera:

Figura 32. Diseño de *borrar.php*

Lo que va a cambiar es que ya no va a estar la categoría del producto, solo va a estar su nombre, por lo tanto el código va a cambiar.

Como ahora solo queremos saber los productos que están en la lista para poder modificar solo esos, los datos no los vamos a recopilar desde una conexión a la base de datos directamente, sino que los vamos a recopilar desde los *Arrays* que contienen los datos de la tabla virtual.

En la parte del código del campo del Nombre del producto se le agregará el siguiente código:

```
<select name="producto" id="producto">
  <option value="-2" selected>- seleccionar -</option>
    <?php
       for ($i=0;$i<=$num-1;$i++){
              echo "<option value=".$idCompra[$i].">".$Producto[$i]."</option>";
       }
    ?>
</select>
```

Entonces ya con este código se generarán los elementos de los *valores de lista* del campo *LISTA/MENÚ producto*, y que serán los mismos elementos que se encuentran en la tabla virtual.

El campo de cantidad y el botón de envío quedan iguales, solo se le cambian los textos como se ve en la imagen. Y en las propiedades del formulario en la acción se mandará a la página ***borrarcarrito.php*** y Destino *_self*.

4.2.1.15. borrarcarrito.php

La página de ***borrarcarrito.php*** va a servir para borrar los datos de la base de datos, para esto la página recibe los datos de la tabla *CompraVirtual* que sean de la fecha en que se esta visitando la página y del usuario que esté activándola en el ese momento.

Específicamente hace referencia por el id de Compra (*idCompra*) de la base de datos. El *idCompra* esta almacenado en el *Array* de los productos del carrito, por lo tanto, si se borra una cantidad o producto de la base de datos, en la tabla virtual ya no aparecerá. El código queda de la siguiente manera:

```
<? include("seguridad.php");

$producto = $_POST['producto'];
$cantidad = $_POST['cantidad'];
$conexion=odbc_connect("access","","");
if (($clasificacion!=-2) && ($cantidad>0)){
        $query="select cantidad from CompraVirtual where idCompra=$producto";
        $rquery=odbc_do($conexion,$query);
        $Cantidad=odbc_result($rquery,"cantidad");
        if ($cantidad>$Cantidad){
                header("Location: borrar.php?errorproductos=si");
        } else {
                $update="update CompraVirtual set cantidad=cantidad-$cantidad where
                    idCompra=$producto";
                $reupdate=odbc_do($conexion,$update);
                header ("Location: borrar.php");
        }
} else {
        header("Location: borrar.php?errorproductos=si");
} ?>
```

De similar acción hace este código que cuando los datos mandados por el formulario no sean correctos se regrese la variable de sesión *errorproductos* con valor igual a "*si*" a la página de *borrar.php*, y esto hará que se despliegue el mensaje de error.

En la página también se colocan 2 botones flash, uno que servirá para vincular a la página *comprar.php*, y el otro para vincular a la de *jeadeca.php* (se puede utilizar el mismo botón que en la de *comprar.php*)

El código para la tabla virtual es exactamente igual a la de *comprar.php.*

Y esto es todo lo que contendrán las páginas, y para visualizar bien cuál va a ser el manejo de direccionamiento o vínculos al navegar sobre ellas lo veremos sobre el siguiente esquema:

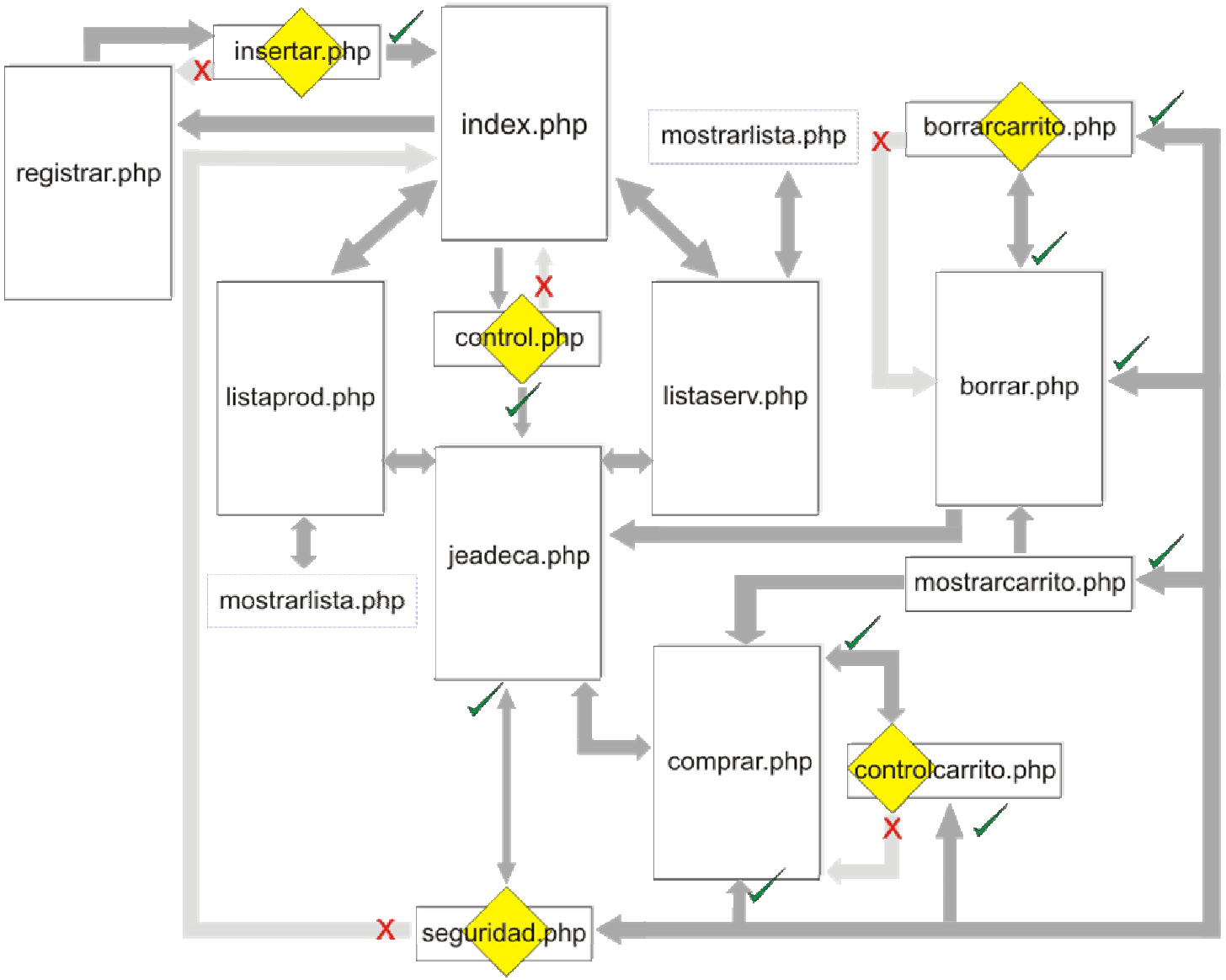


Figura 33. Esquema de vinculación de las páginas creadas en PHP

4.2.2. Access / SQL – ASP.NET – Visual Studio .NET 2003

Para desarrollar la página en asp.net voy a utilizar como lenguaje de programación Visual Basic, la razón es poder mostrar otro de los lenguajes de programación con que se pueden crear los recursos necesarios para las páginas dinámicas con mayor facilidad y versatilidad.

Como primer paso se crea el proyecto. Este proyecto se alojará en el LOCALHOST de la computadora, y en una carpeta de Windows que se alojara en "\Mis documentos\Visual Studio Projects" y al ser cargado al servidor se tendrán los elementos en la carpeta de *Inetpub*.

Se le pondrá un nombre, en este caso le daré el de "EjemploWeb" y la plantilla va a ser de *Aplicación Web ASP.NET* (como ya se había mencionado anteriormente). Al hacer esto Visual Studio creará los accesos al servidor y los elementos predeterminados del proyecto.

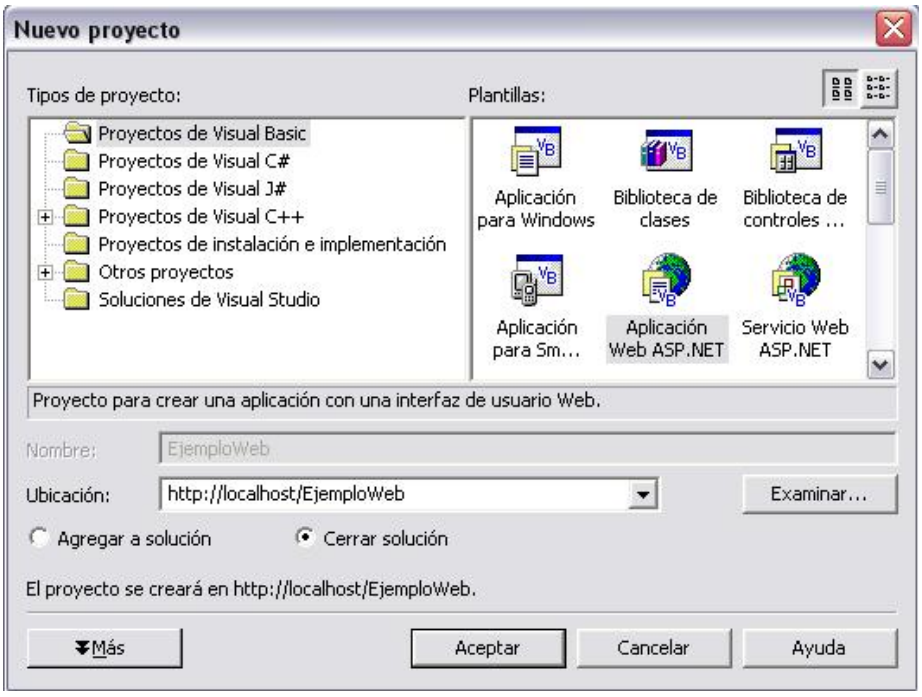


Figura 34. Ventana de creación de proyectos de Visual Studio

Todas las páginas que vamos a utilizar van a ser las mismas que las páginas creadas en PHP con Dreamweaver, esto para ver las diferencias entre uno y otro.

El elemento que vamos a utilizar es el llamado ***WebForm***, este es simplemente la página con la extensión .aspx.

De manera predeterminada ya tiene un elemento llamado WebForm1.aspx, solo hay que dar un *clic* sobre el elemento para poder cambiar el nombre y darle el nombre de nuestra página principal que será ***index.aspx.***

4.2.2.1. index.aspx

Esta página nos sirve como página principal a la cuál van a entrar todos y cada uno de los usuarios visitantes.

Sobre el centro de la página se le da clic derecho al Mouse para que saque las propiedades de la página. Ahí tiene las opciones de ponerle el título a la página y el fondo.

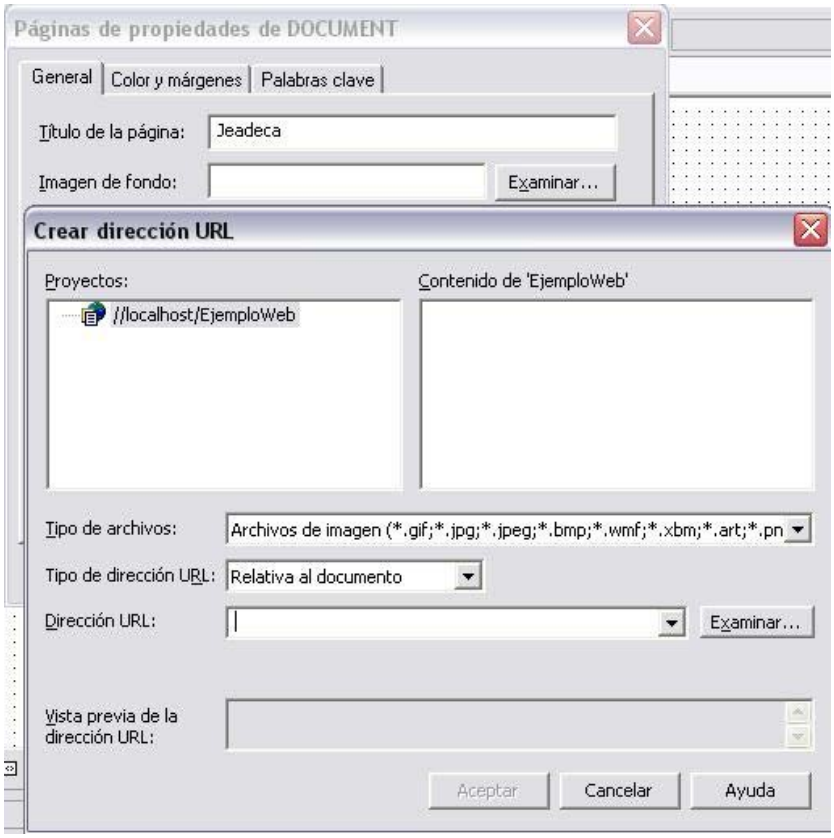


Figura 35. Ventana de selección de ruta de archivo

En donde dice *Examinar* al lado del apartado de *imagen de fondo* se seleccionará. Abrirá otra ventana en la cual al lado de *Dirección URL* muestra otra parte para *Examinar*, se selecciona y abrirá otra ventana en la cual se seleccionará la imagen que se quiere como fondo.

Ya teniendo el fondo, en el cuadro de herramientas, en la cual están todos los elementos. Aquí existen tipos de elementos de *Web Forms*, y *HTML*. La diferencia entre estos es que los de Web Forms tienen la propiedad de que se ejecuta como control de servidor automáticamente y los de HTML no. Esta propiedad debe de estar activa si es que se requiere que el elemento realice una opción o instrucción específica para mandar al servidor.

Los elementos se seleccionan y se dibujan sobre la página. Aquí no es necesario colocarlo dentro de una capa para poder posicionarlo en cualquier parte de la página como se hace en el Dreamweaver, solo arrastra con el Mouse y listo.

Dos de los primeros elementos que se van a poner es la imagen del logo de la empresa y el texto al lado de ella. En el cuadro de herramientas se encuentra el elemento de *"Image"* se selecciona y se hace un recuadro en la página en donde estará el elemento. Al colocarse en la parte de propiedades del elemento en *ImageUrl* se seleccionará la ruta de donde se encuentra la imagen. Para el texto se coloca un *"Label"* y en sus propiedades en la parte de *Text* se escribe el texto que va a aparecer dentro del Label.

Para colocar la tabla, se selecciona la tabla HTML, esto porque es mas fácil de manipular. Se dibuja la tabla, pero aquí no se puede manipular como en Dreamweaver, para esto tenemos que irnos al código HTML, que se encuentra en la parte inferior, al lado de DISEÑO.

Figura 36. Submenú de selección de vista de la página

Si queremos tener nuestra tabla como la tenemos hecha en Dreamweaver se tendrá de la siguiente manera en el código HTML:

```
<table cellPadding="5" width="272" border="2">
    <tr>
        <td align="center" colSpan="2"></td>
     </td></tr>
    <tr>
        <td style="HEIGHT: 38px"> </td>
        <td style="HEIGHT: 38px"> </td>
    </tr>
    <tr background="Fondos/azul08.jpg">
        <td></td>
        <td></td>
    </tr>
    <tr background="Fondos/azul08.jpg">
        <td align="center" colSpan="2">
        </td>
    </tr>
</table>
```

Ya teniendo la tabla igual, solo se escribe el texto que queremos que aparezca dentro de la tabla. Para esto vamos a crear 2 label, uno se llamará “texto” y otro “textoerror”. En el label *texto* escribiremos “Introduce tu Nick y Password que asignaste en tu registro” que es lo que aparecerá cada ves que se inicie la página, y en el de *textoerror* escribiremos “Datos incorrectos” pero con su propiedad Visible en *false*, esto para cuando los datos que se manden a la base de datos si no son correctos nos regrese el mensaje de error. Estos 2 los colocaremos dentro de la primera celda.

También insertar las cajas de texto, el botón y el link de texto como se tenían respectivamente. Las cajas de texto se llamarán “usuario” y “password” (respectivamente), el botón se llamará “ingresar”, el link será un *Linkbutton* y se llamará “registrarse”.

Después de esto se colocarán 3 linkbutton que servirán para mostrar las características de la empresa, como ya lo habíamos puesto antes (*¿Quiénes somos?, Productos, Servicios*), y en donde se va a mostrar la información será en *paneles*. Estos se sacan del cuadro de herramientas, se selecciona el panel y se escribe la información. Un panel para cada uno

de los linkbutton. Pero además de los paneles habrá 2 botones, uno para cuando se muestre la lista de los productos que se llamará “btnlista”, y otro para la de los servicios que se llamará “bntmantenimiento”.
Se verá de la siguiente manera:

Figura 37. Diseño de los linkbutton

Ya que se tienen todos los elementos pasamos a la parte del lenguaje de programación. Al dar doble clic sobre la página nos desplegará otra parte de la página pero esta es la parte del lenguaje Visual Basic con el que programaremos. Comenzaremos con declarar la variable de conexión.
Esta variable la vamos a poner pública debajo de `Código generado por el Diseñador de Web Forms` para que se aplique a toda la página y se declara de la siguiente manera:

```
Public Cn As New System.data.OleDb.OleDbConnection
```

Ya declarada la variable de conexión vamos a aplicar el evento de conexión cuando se presione el botón de ingresar, y también que haga la búsqueda del usuario y password en la base de datos, si el usuario y el password son correctos lo mande a otra a la pagina *jeadeca.aspx* sino, que regrese a la misma página pero marcando un mensaje de error.
Se le da doble clic al botón de ingresar y aparecerá el *Private sub* del evento click del botón. Ahí colocaremos lo siguiente

```
Private Sub ingresar_ServerClick(ByVal sender As System.Object, ByVal e As
                            System.EventArgs) Handles ingresar.ServerClick
        Try
            Cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
                            ID=Admin;Data Source=C:\Ejemplo.mdb;"
            Cn.Open()
            Dim commSQL As New System.Data.OleDb.OleDbCommand
            Dim datRead As System.Data.OleDb.OleDbDataReader

            commSQL = New OleDb.OleDbCommand("Select Nick, Password From
                     Usuario ", Cn)

            datRead = commSQL.ExecuteReader
            While datRead.Read()
                If datRead.Item("Nick") = usuario.Text And
                   datRead.Item("Password") = password.Text Then
                    Session("conectado") = "si"
                    Session("USR") = datRead.Item("Nick").ToString
                    Response.Redirect("jeadeca.aspx")
                End If
            End While

            datRead.Close()
            Cn.Close()

            textoerror.Visible = True
            texto.Visible = False
            usuario.Text = ""
            password.Text = ""
        Catch
            Response.Write("Imposible realizar conexión")
        End Try
End Sub
```

Explicando el anterior código, la conexión a la base de datos se da en la ruta que marca el cn.connectionstring. Aquí se pone la ruta de donde se encuentra la base de datos, ya sea el archivo de .mdb de Access o la ruta de la base de datos que controla SQL.

ACCESS

```
Cn.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;User
ID=Admin;Data Source= ruta y nombre de la base de datos;"
```

SQL

```
Cn.ConnectionString = "Provider=SQLOLEDB;User
Id=nombre de inicio de sesion;Password=contraseña;Initial
Catalog=nombre de la base de datos;Data Source=servidor activo en
SQL;"
```

En versiones anteriores de Visual Basic se contaba con un objeto llamado RecordSet, el cual permitía alojar el resultado de la ejecución de un comando. En ADO .NET el mismo no existe, por lo que debemos pensar entonces en un objeto capaz de realizar tareas similares. El nuevo modelo de clases para acceso a datos provee un objeto llamado DataReader, el cual permite ejecutar una consulta almacenada por un comando, y posteriormente acceder a su resultado (este es ReadOnly - ForwardOnly).
DataReader ofrece la forma más eficiente de acceso a un conjunto de filas, y es el único provisto por ADO .NET para consumir una conexión durante todo el tiempo.
El primer paso es crear los siguientes objetos de tipo DataReader y el de tipo Comando:

```
Dim datRead As System.Data.OleDb.OleDbDataReader
Dim commSQL As New System.Data.OleDb.OleDbCommand
```

El segundo paso es asignar al comando la consulta y la conexión:

```
commSQL = New OleDb.OleDbCommand("Select Nick, Password From Usuario ", Cn)
```

El enlace con el objeto de comando se hace:

```
datRead = commSQL.ExecuteReader
```

Para consultar el contenido de cada registro, existen dos formas. Ambas modalidades utilizan la propiedad Item para obtener el valor de la columna, y lo hacen en el formato original definido por la columna de la base de datos. La primera consiste en especificar el nombre del campo mediante una cadena de caracteres:

```
While datRead.Read()
    If datRead.Item("Nick") = usuario.Text And
      datRead.Item("Password") = password.Text Then
```

```
            Session("conectado") = "si"
            Session("USR") = datRead.Item("Nick").ToString
            Response.Redirect("jeadeca.aspx")
        End If
    End While
```

La segunda consiste en acceder al mismo mediante su ordinal, siendo la primera posición siempre 0:

```
    While datRead.Read()
        If datRead.Item(0) = usuario.Text And datRead.Item(1) =
                            password.Text Then
            Session("conectado") = "si"
            Session("USR") = datRead.Item("Nick").ToString
            Response.Redirect("jeadeca.aspx")
        End If
    End While
```

Para mejor entendimiento en las siguientes páginas se utilizará la primera opción para identificar mejor lo que queremos traer como resultado.

Como se ve, en las 2 formas se declararon 2 variables de sesión, las cuales se llaman “*Conectado*” y “*USR*”. Estas tendrán un valor con el cuál vamos a manipular para poder acceder a las demás páginas, es decir, como ya lo habíamos visto, si el usuario no esta registrado, y no ha iniciado una sesión, no podrá acceder a las otras páginas, por lo tanto en cada página vamos a checar que la variable de sesión *conectado* tenga el valor de “si”, si no es así regresará a la página de *index.aspx*. En este casó, si está coinciden los valores de usuario y password dados en las cajas de texto permitirá ir a la página *jeadeca.aspx*.

Entonces, si es que coinciden entra a la condición en la que asigna las variables de sesión y direcciona a la otra página automáticamente, por lo tanto, si no coinciden el usuario y contraseña escritos en las cajas de texto es que no entra a la condición *if* y no podrán entrar a las demás páginas. Para esto se coloca después de que se cierre la conexión del dataRead y de la conexión total (Cn), la propiedad visible del label “texto” en *false* y la de “textoerror” en *true*.

```
        textoerror.Visible = True
        texto.Visible = False
```

Entonces como se ve, se ocultará lo que esta escrito en la label “texto” y visualizará el contenido de la label “textoerror”. Después limpiaremos las cajas de texto asignándoles un valor en blanco.

```
usuario.Text = ""
password.Text = ""
```

Todo esto lo colocamos en una instrucción *TRY* la cual sirve para asegurar un código que si marca error lo atrape y realice una instrucción que le asignemos. Dentro de ***Try*** colocaremos el código de lo que queremos realizar, y dentro de ***Catch*** colocaremos el código de lo que queremos que se haga si es que marca un error y se finaliza el *Try* con ***End Try***.

Al hacer el clic en el link de registrarse se redireccionará a la página de *registrar.aspx*, esto dando doble clic sobre el *Linkbutton* “registrarse” y colocando en el código lo siguiente:

```
Private Sub registrarse_Click(ByVal sender As System.Object, ByVal e
                    As System.EventArgs) Handles registrarse.Click
  Response.Redirect("registrar.aspx")
End Sub
```

Antes de pasar a la página de registrar falta poner el código de los linkbuttons y sus respectivos botones, entonces después del código que ya acabo de mencionar pondré el siguiente:

```
Private Sub somos_Click(ByVal sender As System.Object, ByVal e As
                    System.EventArgs) Handles somos.Click
    Panel1.Visible = True
    Panel2.Visible = False
    Panel3.Visible = False
    btnlista.Visible = False
    btnmantenimiento.Visible = False
End Sub

Private Sub productos_Click(ByVal sender As System.Object, ByVal e As
                        System.EventArgs) Handles productos.Click
    Panel2.Visible = True
    Panel1.Visible = False
    Panel3.Visible = False
    btnlista.Visible = True
    btnmantenimiento.Visible = False
End Sub

Private Sub servicios_Click(ByVal sender As System.Object, ByVal e As
                    System.EventArgs) Handles servicios.Click
    Panel3.Visible = True
    Panel1.Visible = False
    Panel2.Visible = False
```

```
        btnlista.Visible = False
        btnmantenimiento.Visible = True
    End Sub

    Private Sub btnlista_Click(ByVal sender As System.Object, ByVal e As
                               System.EventArgs) Handles btnlista.Click
        Response.Redirect("listaprod.aspx")
    End Sub


    Private Sub btnmantenimiento_Click(ByVal sender As System.Object, ByVal
                       e As System.EventArgs) Handles btnmantenimiento.Click
        Response.Redirect("listaserv.aspx")
    End Sub
```

La explicación de este código es que cuando se de clic en los linkbutton de “somos”, “productos” o “servicios”, en cada uno pondrá visible el panel y botón que le corresponde y pondrá invisible los demás paneles y botones.

Además cuando se de clic sobre el botón de lista o mantenimiento estos redireccionarán a la página correspondiente.

Ahora crearé la página *registrar.aspx*.

4.2.2.2. registrar.aspx

Esta página sirve para poder registrar nuevos usuarios y que estos puedan acceder, ya después registrarse, a las demás páginas restringidas. Comenzaré colocando el mismo encabezado que tendrán todas las páginas (Logo, Texto y regla de separación).

Luego voy a poner los cuadros de texto y labels. Además las lista/menú y los botones de opción que pusimos en Dreamweaver se llaman diferentes pero de igual manera aparecen, la lista/menú se llama “*DropDownList”* y los botones de opción como los requerimos se llaman “*RadioButtonList*”. Y el texto de error también va a se una *Label*, como en la página anterior. Todo lo voy a colocar exactamente igual que en la de *registrar.php*, y agregaré otras cosillas.

Estas cosillas son precisamente unos mensajes de error que se habilitarán si los cuadros de texto están vacíos. Se llaman "*RequiredFieldValidator*" y "*RangeValidator*" va a ir uno para cada caja de texto.

Quedará de la siguiente manera:

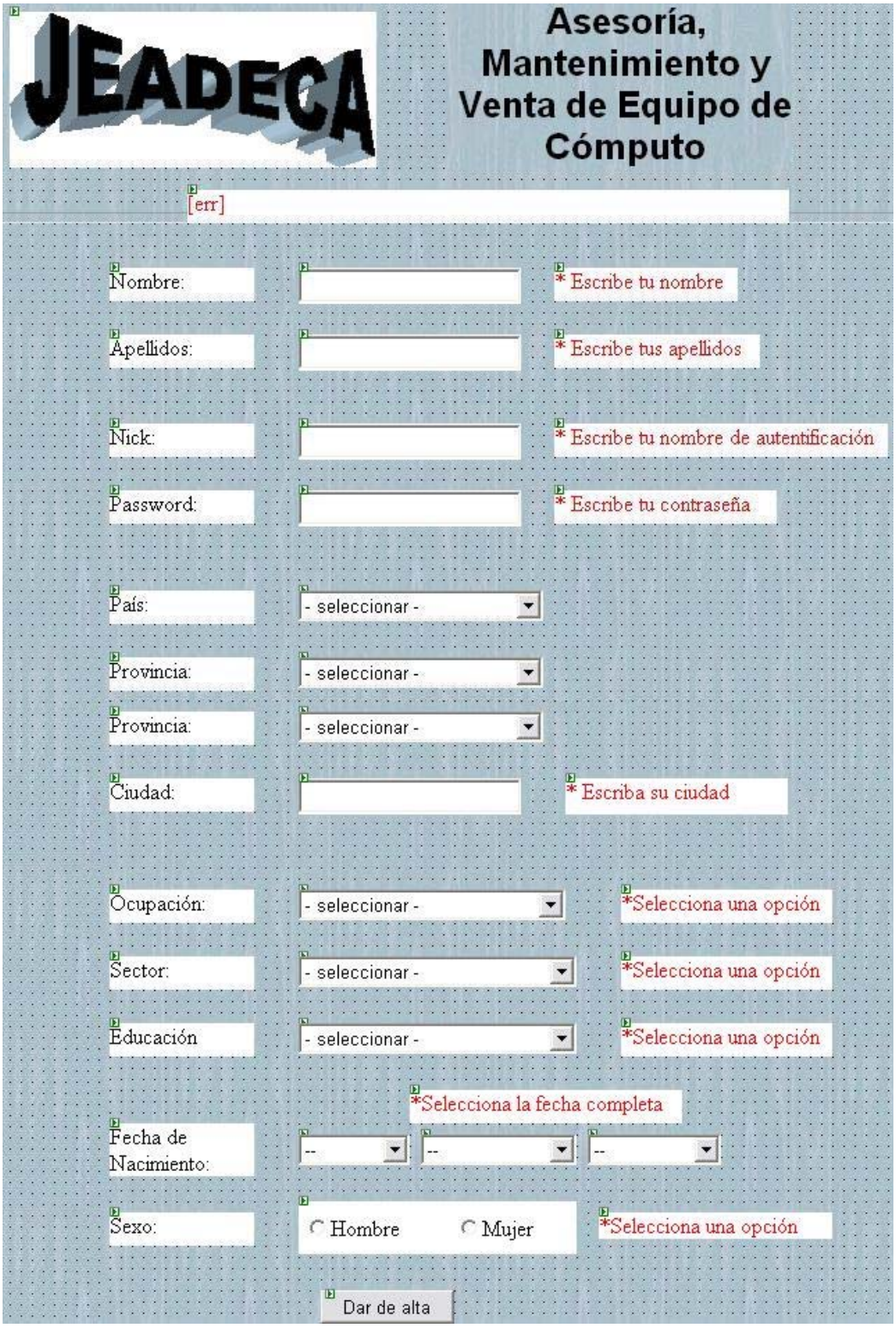


Figura 38. Diseño de *registrar.aspx*

Como se ve en la imagen queda todo ordenado. Los nombres de las cajas de texto serán respectivos a las etiquetas (Label) que le corresponden (pero recordando que todos los nombres los estoy poniendo en minúsculas).

El DropDownList de “País” y “Provincia” le llamaré *combopais* y *comboprov* respectivamente. Los demás DropDownList llevarán los nombres respectivos a sus etiquetas, con excepción de las de fechas, porque como lo puse en la página php, uno marcará los días, otro los meses y el otro los años, entonces colocaré sus nombres como *dia, mes* y *ano* respectivamente. Todos estos DropDownList llevarán datos en su lista, y estos datos se colocan en la parte de *“Ítems”* de sus propiedades, con excepción de combopais y comboprov que los datos se llenaran con los datos de la tabla de país y provincia. El RadioButtonList también tendrá su nombre respectivo a su etiqueta.

Ya nombrados prosigue colocar en las propiedades de cada uno de los RequiredFieldValidator su nombre y el control que se va a validar. Entonces en sus opciones en ID se pone el nombre, en ControlToValidate se selecciona el control que va a validar, y en ErrorMessage se escribe el texto que va a aparecer. Estos RequiredFieldValidator solo servirán para los cuadros de texto y el RadioButtonList.

Para los Dropdownlist se utilizarán RangeValidator, y estos sirven para identificar qué valores están permitidos como mínimo y máximo, es decir, a cada Dropdownlist en sus ítems se les puso un texto y su valor correspondiente, aquí solo basta con poner en cada Rangevalidator el mínimo valor aceptado, que será 1, y el máximo que será el valor del último elemento. Estos valores se ponen sus propiedades *MinimumValue* y *MaximumValue*.

De igual manera se colocará el texto aparezca cuando ocurra el error en la propiedad *ErrorMessage* y control que va a validar en *ControlToValidate*.

En los de fecha coloqué los 3 RangeValidator uno sobre otro con el mismo mensaje de error, esto para unificar que se el error por fecha completa, y no por cada una de sus partes, pero en cada uno de los rangevalidator se ponen sus valores máximos y mínimos referentes a cada dropdownlist.

Ahora viene la parte del código.

Para poder cargar los datos del país sus las provincias en los dropdownlist correspondientes hay que traer los datos de las tablas de la base de datos (como ya habíamos mencionado).

Para Asp no necesitaremos utilizar javascript, solamente tendremos que utilizar el lenguaje de Visual Basic para traer esos datos, y una de sus opciones del lenguaje Visual Basic es la función *DataSet*.

Se crean las variables para controlar los datos con el Dataset y para la conexión a la base de datos, y serán las siguientes:

```
Dim ds As New DataSet
Dim commSQL As New OleDb.OleDbDataAdapter
Dim commSQL2 As New OleDb.OleDbDataAdapter

Public Cn As New System.data.OleDb.OleDbConnection
```

Cuando cargue la página se necesita que traiga los datos de los países y de las provincias, para esto en la parte del código de *Page_Load* se pondrá el siguiente código:

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
        System.EventArgs) Handles MyBase.Load

    Cn.ConnectionString = Ruta de conexión Access o SQL
    Cn.Open()

    If Not IsPostBack Then

        commSQL = New OleDb.OleDbDataAdapter("Select idpais, nompais
                                    From Pais order by nompais ", Cn)
        commSQL.Fill(ds, "Pais")


        combopais.DataSource = ds.Tables("Pais").DefaultView
```

```
            combopais.DataTextField = "NomPais"
            combopais.DataValueField = "idPais"
            combopais.DataBind()
            BindProvincia()
        End If
        Cn.Close()

    End Sub
```

La explicación del código anterior es la siguiente: en la parte inicial se crea la conexión y se abre. Luego hago una condición en la que muestra *si no hay un regreso de página* (postback) a la variable ***commSQL*** le asigna el DataAdapter con la información de qué búsqueda debe hacer a la base de datos. Después se hace la conexión Dataset con la variable commSQL y se deja lista la variable commSQL en su propiedad *Fill* (llenar) para que guarde los datos, asignándole un nombre a la tabla interna que tendrá la variable, a esta se le puede poner cualquier nombre, en este caso yo le pondré “Pais”. Después se le asignará al Drowpdownlist “*combopais*” en sus propiedades los siguientes datos:

- DataSource: el Dataset con su tabla interna en su vista inicial.
- DataTextField: el texto que aparecerá en los elementos del dropdownlist, que en este caso serán los resultados de los nombres de los países.
- DataValueField: el valor de los elementos del dropdownlist, que en este caso serán los resultados de los id de los países.
- Databind: hace que se ligue el Dropdownlist con los resultados del Dataset.

Cada id y nombre estarán relacionados en la tabla interna del Dataset.

Después está ***BindProvincia*** que es una subrutina que realizará el llenado del Dropdownlist comboprov. Se termina la condición y finalmente se cierra la conexión.

Después se creará la subrutina ***BindProvincia*** con el siguiente código:

```
Public Sub BindProvincia()

  commSQL2 = New OleDb.OleDbDataAdapter("select idProvincia,
            NomProvincia from Provincia Where idPais=" &
            combopais.SelectedItem.Value & " order by NomProvincia ",
            Cn)
  commSQL2.Fill(ds, "Provincia")

  comboprov.DataSource = ds.Tables("Provincia").DefaultView
  comboprov.DataTextField = "NomProvincia"
  comboprov.DataValueField = "idProvincia"
  comboprov.DataBind()

End Sub
```

Es igual al anterior, claro, con la diferencia de la búsqueda a la base de datos, y donde la referencia de la integridad entre las 2 tablas (país y provincia) es el valor del combopais cuando este se seleccione.

Cuando se cambie de elemento el combopais tendrá que cambiar automáticamente el comboprov, para esto se tienen que hacer 2 cosas. La primera es que en la propiedad *SelectedIndexChanged* del combopais se pone el siguiente código:

```
Private Sub combopais_SelectedIndexChanged(ByVal sender As System.Object,
            ByVal e As System.EventArgs) Handles
            combopais.SelectedIndexChanged
      BindProvincia()

End Sub
```

Para poder acceder a esta propiedad sólo basta con dar doble clic sobre el Dropdownlist combopais.

Y como segunda cosa es poner en la vista de diseño sobre el combopais, en sus propiedades, activar ***autopostback*** en *true*. Esto es para cuando cambie, recargue la página pero con la modalidad postback. Entonces así cuando inicie la página cargará el país por orden alfabético y sus respectivas provincias, pero si se cambia el nombre del país, pasará a la forma postback y cargará las provincias del país seleccionado.

Por último se pone el código para que guarde todos los datos en la base de datos en la tabla *Usuario*, esto será cuando se le active el botón de enviar, y será el código siguiente:

```
Private Sub enviar_ServerClick(ByVal sender As System.Object, ByVal e As
                               System.EventArgs) Handles enviar.ServerClick
        Dim insertcommand As System.Data.OleDb.OleDbCommand
        Dim datread As System.Data.OleDb.OleDbDataReader
        Dim querycommand As System.Data.OleDb.OleDbCommand
        Dim maxusua As Integer
        Dim insertdatos As String
        Dim nacimiento As String
        Cn.Open()
        querycommand = New OleDb.OleDbCommand("select idusuario from
                                              Usuario", Cn)
        datread = querycommand.ExecuteReader
        While datread.Read()
            maxusua = datread.Item("idusuario").ToString + 1
        End While
        datread.Close()
        Cn.Close()
        nacimiento = dia.Value + "/" + mes.Value + "/" + ano.Value

        insertdatos = "insert into Usuario values (" & maxusua.ToString &
        ",'" + nombre.Text + "','" + apellidos.Text + "','" + nick.Text +
        "','" + password.Text + "','" & nacimiento.ToString & "','" &
        sexo.SelectedValue & "','" & combopais.SelectedItem.ToString & "','"
        & comboprov.SelectedItem.ToString & "','" + ciudad.Text + "','" &
        ocupacio.SelectedItem.Value & "','" & secto.SelectedItem.Value &
        "','" & educacio.SelectedItem.Value & "')"

        insertcommand = New System.Data.OleDb.OleDbCommand(insertdatos, Cn)
        insertcommand.Connection.Open()
        Try
            insertcommand.ExecuteNonQuery()
            insertcommand.Connection.Close()
            Response.Redirect("registrado.aspx")
        Catch Exp As System.data.OLEDB.OleDbException
            err.Text = "ERROR: no se pudo agregar el registro, compruebe
                        que los campos están rellenos correctamente"
        End Try
        insertcommand.Connection.Close()
    End Sub
```

Se cuenta también con otro tipo de consultas, las cuales no retornan filas después de la ejecución de la misma. Este tipo de sentencias se utiliza muy comúnmente para crear objetos en la base de datos (tablas, permisos), así como también para modificar la información de una o varias filas (Insert, Update o Delete). Para dicho fin, los objetos de comando cuentan con un método denominado ExecuteNonQuery, que

permite al mismo optimizar el proceso, partiendo de la base de que la instrucción no retornará filas.

Por esta razón se coloca este comando para insertar los datos, lo demás del código es similar. Si ocurre, escribirá el texto de error en el Label de error, y si es que se inserta se redireccionará a la página *registrado.aspx*.

4.2.2.3. registrado.aspx

Esta página solo servirá para redireccionar a la página principal. Va a tener, además del mismo encabezado que las demás páginas, un label y un linkbutton que regresará a la página *index.aspx*. Por lo tanto en el linkbutton en su evento clic tendrá el siguiente código:

```
Response.Redirect("index.aspx")
```

Figura 39. Diseño de *registrado.aspx*

Regresando a la página de index.aspx están los botones que mostrarán la lista de los productos y servicios. Cada uno tendrá en su evento clic el redireccionamiento a otra página.

El botón “lista” mandará a la página ***listaprod.aspx***, la cual mostrará la lista de los productos que tiene la empresa (como ya se había mencionado

en la parte de php) y el botón “mantenimiento” mostrará la lista de servicios que se tienen.

4.2.2.4. listaprod.aspx y listaserv.aspx

Estas páginas servirán para mostrar el listado de productos y servicios existentes en la base de datos ordenados por su clasificación.

Tendrá el mismo encabezado, un dropdownlist llamado “comboclasif”, un ***datagrid*** y un botón llamado “regresar”.

El datagrid es un control que muestra los datos en formato de tabla y, de manera opcional, admite la selección, ordenación, paginación y edición de los datos[4]. De tal manera que utilizaré este control para mostrar los resultados que nos mande de la base de datos sobre los productos o servicios.

Quedará de la siguiente manera:

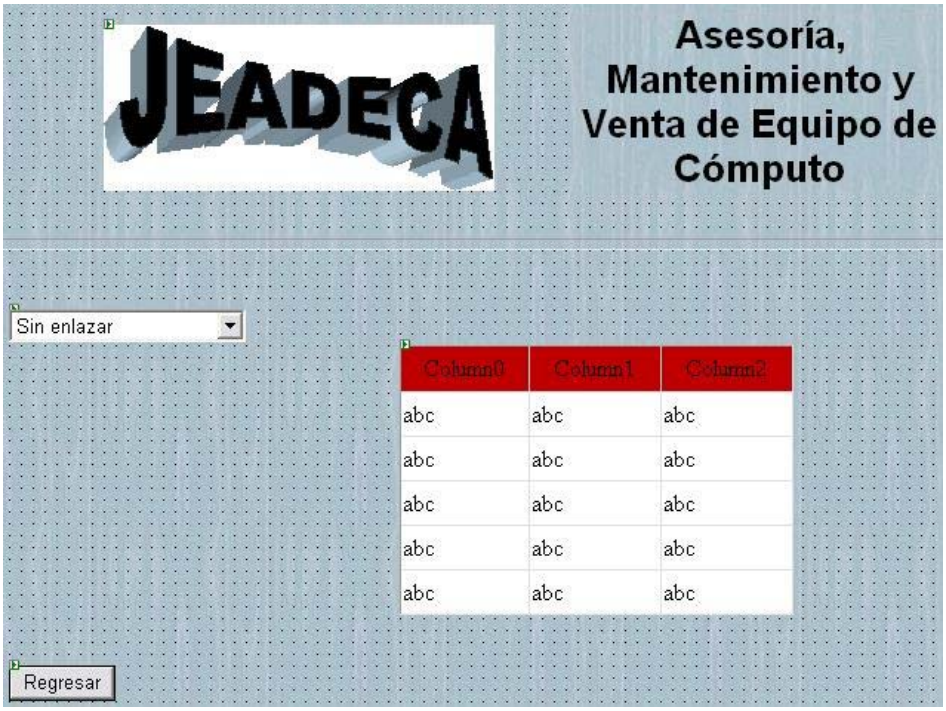


Figura 40. Diseño de *listaprod.aspx* y *listaserv.aspx*

[4] En http://es.gotdotnet.com/quickstart/aspplus/samples/webforms/ctrlref/webctrl/datagrid/doc_datagrid.aspx

Entonces para hacer la conexión a la base de datos y traer los datos requeridos utilizo el siguiente código para la página *listaprod.aspx*:

```
Dim adaptador As System.Data.OleDb.OleDbDataAdapter
Dim ds, ds2 As New DataSet
Dim commSQL As OleDb.OleDbDataAdapter

Public Cn As New System.data.OleDb.OleDbConnection

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
                      System.EventArgs) Handles MyBase.Load
    Cn.ConnectionString = Ruta de conexión Access o SQL
    Cn.Open()
    If Not IsPostBack Then
        commSQL = New OleDb.OleDbDataAdapter("Select idClasif,
                  NomClasif from Clasificación", Cn)
        commSQL.Fill(ds, "Clasif")
        comboclasif.DataSource = ds.Tables("Clasif").DefaultView
        comboclasif.DataTextField = "NomClasif"
        comboclasif.DataValueField = "idClasif"
        comboclasif.DataBind()
        tabla()
    End If
    Cn.Close()
End Sub

Public Sub tabla()
    adaptador = New System.Data.OleDb.OleDbDataAdapter("select
                NomProducto as [Producto], Precio from Producto where
                idClasif= " & comboclasif.SelectedValue & " order by
                NomProducto", Cn)
    ds2 = New System.Data.DataSet
    adaptador.Fill(ds2, "NomTab")
    Datagrid1.DataSource = ds2.Tables("NomTab").DefaultView
    Datagrid1.DataBind()
End Sub
 Private Sub comboclasif_SelectedIndexChanged(ByVal sender As Object,
                            ByVal e As System.EventArgs) Handles
                            comboclasif.SelectedIndexChanged
    tabla()
End Sub
```

Como se ve, el código tiene partes similares a la página de registrar, solo que aquí tiene un solo dropdownlist. Este controla la clasificación de los productos, y cuando se selecciona alguna el datagrid toma el valor del comboclasif para hacer el llamado de los datos de la tabla productos correspondientes a esa clasificación. Todo esto se hace en la subrutina ***tabla***, en donde el adaptador guarda la instrucción de la búsqueda que se va a hacer, después la aplica al dataset y manda llenar el datagrid con esos datos.

Y el código para página *listaserv.aspx* será similar, solo cambiarán las instrucciones de búsqueda a la base de datos y es el siguiente:

```
Dim adaptador As System.Data.OleDb.OleDbDataAdapter
Dim ds, ds2 As New DataSet
Dim commSQL As OleDb.OleDbDataAdapter

Public Cn As New System.data.OleDb.OleDbConnection

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
                      System.EventArgs) Handles MyBase.Load
    Cn.ConnectionString = Ruta de conexión Access o SQL
    Cn.Open()
    If Not IsPostBack Then
        commSQL = New OleDb.OleDbDataAdapter("Select idClasSer,
                 NomClasSer from ClasSer", Cn)
        commSQL.Fill(ds, "Clasif")
        comboclasif.DataSource = ds.Tables("Clasif").DefaultView
        comboclasif.DataTextField = "NomClasSer"
        comboclasif.DataValueField = "idClasSer"
        comboclasif.DataBind()
        tabla()
    End If
    Cn.Close()
End Sub

Public Sub tabla()
    adaptador = New System.Data.OleDb.OleDbDataAdapter("select
              NomServicio as [Servicio], Precio from Servicio where
              idClasSer= " & comboclasif.SelectedValue & " order by
              NomServicio", Cn)
    ds2 = New System.Data.DataSet
    adaptador.Fill(ds2, "NomTab")
    Datagrid1.DataSource = ds2.Tables("NomTab").DefaultView
    Datagrid1.DataBind()
End Sub

Private Sub comboclasif_SelectedIndexChanged(ByVal sender As
           System.Object, ByVal e As System.EventArgs) Handles
           comboclasif.SelectedIndexChanged
    tabla()
End Sub
```

Por último, el botón “regresar” redireccionará a la página anterior que se había accesado, pero aquí cómo saber a cual si va a haber 2 páginas desde las cuales se podrá entrar a “listaprod.aspx” o “listaserv.aspx”, una será de index.aspx y la otra de jeadeca.aspx. Entonces primero explicaré la creación de *jeadeca.aspx* y luego explicaré el comportamiento del botón.

Ya teniendo la página de registro y la redireccionamiento podremos acceder a las demás páginas en las cuales solo se permiten para usuarios registrados.

4.2.2.5. jeadeca.aspx

La página principal es la de ***jeadeca.aspx***, y que como en la de PHP será en general igual a la de index pero haciendo algunas correcciones. Primero se quita la tabla de ingreso, luego se agregan 2 botones y una etiqueta de bienvenida para el usuario registrado.

Como ya expliqué, la página no se abrirá si no está registrado el usuario, entonces voy a poner una condición en el evento page_load para realizar esto. Si entra a la página, la etiqueta de bienvenida muestra el saludo "Bienvenido:" con el Nick del usuario que entró, todo esto con el siguiente código:

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
                  System.EventArgs) Handles MyBase.Load
    If Session("conectado") <> "si" Then
        Response.Redirect("index.aspx")
    Else
        Bienvenida.Text = "Bienvenido: " & Session("USR")
    End If
End Sub
```

Ahora en un botón que llamaré "desconectar", pondré un código para cerrar la sesión activa y por lo tanto que regrese a la página index.aspx:

```
Private Sub desconectar_Click(ByVal sender As System.Object, ByVal e As
                          System.EventArgs) Handles desconectar.Click
    Session.Abandon()
    Response.Redirect("index.aspx")
End Sub
```

Entonces para retomar el asunto del botón de regresar. Si a la página jeadeca.aspx solo se puede acceder si esta activa la variable de sesión, entonces la solución la brindará la variable de sesion "conectado". Si redirecciono a jeadeca.aspx y no esta activa la variable de sesión me

regresará a index.aspx, pero si esta activa me mandará a jeadeca.aspx, por lo tanto así podré saber de cuál página accedí.

Entonces a la página *listaprod.aspx* y *listaserv.aspx* en la propiedad clic del botón “regresar” se le pondrá:

```
Private Sub regresar_Click(ByVal sender As System.Object, ByVal e As
                        System.EventArgs) Handles regresar.Click
    Response.Redirect("jeadeca.aspx")
End Sub
```

Ya entendido cómo quedaron las páginas anteriores vuelvo a regresar a la de jeadeca.aspx. Todo el código de los controles similares a index.aspx va a quedar igual en la de jeadeca.aspx.

Ya delimité la seguridad en la página y también cómo abandonar la sesión, ahora falta el otro botón que llamaré “btncomprar”. Este botón solo tendrá función el redireccionar a la página ***comprar.aspx***.

```
Private Sub btncomprar_Click(ByVal sender As System.Object, ByVal e As
                          System.EventArgs) Handles btncomprar.Click
    Response.Redirect("comprar.aspx")
End Sub
```

Terminando con jeadeca.aspx paso a crear la página *comprar.aspx*.

4.2.2.6. comprar.aspx

Esta página nos servirá para seleccionar productos que estén en existencia y poder comprarlos, y se guardará la compra en una tabla en la base de datos.

Esta página tendrá el encabezado, la etiqueta de bienvenida, 3 etiquetas de error, otras etiquetas de texto, 2 DropDownList, 1 caja de texto, 1 RequiredfielValidator, 1 DataGrid y 3 botones repartidos de la siguiente manera:

Figura 41. Diseño de *comprar.aspx*

La etiqueta de bienvenida es exactamente igual a la página anterior, y una etiqueta de error llamada “err” va a ir en la parte de arriba.

Se pone una etiqueta como título de “Categoría del Producto” para el DropDownList que se llamará “comboclasif” que mostrará la lista de clasificaciones para los productos. Después el otro dropdownlist se llamará “comboprod” que es el que desplegará la lista de productos dependiendo de qué clasificación se haya seleccionado.

Debajo la caja de texto que servirá para colocar la cantidad del producto que se va a comprar, y debajo de esta el RequiredFieldValidator que controle a la caja de texto.

Al lado de la caja de texto colocaré 2 etiquetas, una mostrará el texto “En existencia:” y la otra estará en blanco pero la llamaré “prod”. Debajo de esto esta el botón que se llamará “btncomprar”.

Un botón que se llamará “terminar” que redireccionará la página de vuelta a *jeadeca.aspx*, y otro botón arriba del Datagrid que se llamará “borrar” que redireccionará la página a la página ***borrar.aspx***.

Ahora va la explicación de cómo va a ser el proceso de código de la página.

En esta página se van a juntar todas las funciones que se habían programado en las demás páginas, y para darle entendimiento del proceso que hace lo explicaré de la siguiente forma:

Se crean las diferentes variables que se van a utilizar.

```
Dim datRead As System.Data.OleDb.OleDbDataReader
Dim ds, ds2 As New DataSet
Dim commSQL, commSQL2 As New OleDb.OleDbDataAdapter
Dim Adaptador As System.Data.OleDb.OleDbDataAdapter
Dim usr As String
Dim fecha1 As Date
Dim max As Integer

Public Cn As New System.data.OleDb.OleDbConnection
```

Después se va a la parte de cuando carga la página (Page_Load).

```
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As
                      System.EventArgs) Handles MyBase.Load
    fecha = DateTime.Now.ToString("dd/MM/yyyy")
    If Session("conectado") <> "si" Then
        Response.Redirect("index.aspx")
    Else
        Bienvenida.Text = "Bienvenido: " & Session("USR")
        Cn.ConnectionString = Ruta de conexión Access o SQL
        Cn.Open()
        usr = Session("USR")
        tabla()
         If Not IsPostBack Then
             commSQL = New OleDb.OleDbDataAdapter("Select idClasif,
                       NomClasif From Clasificación", Cn)
             commSQL.Fill(ds, "Clasificacion")

             comboclasif.DataSource =
                            ds.Tables("Clasificacion").DefaultView
             comboclasif.DataTextField = "NomClasif"
             comboclasif.DataValueField = "idClasif"
             comboclasif.DataBind()
             BindProducto()
             Cn.Close()
             bindexist()
         End If
         Cn.Close()
    End If
End Sub
```

Cuando se carga la página se le asigna a la variable “fecha” la fecha actual. Se hace la restricción de seguridad con la variable de sesión “conectado”.

Si pasa, da la bienvenida, se hace la conexión y se le asigna el valor de la variable de sesión “USR” a la variable “*usr*” y manda llamar a la subrutina ***tabla***. Esta subrutina es la que llena los datos del datagrid. Estos datos van a provenir de la base de datos traídos de la tabla *CompraVirtual*, que es donde se almacenará lo que se ha comprado por cada usuario.

Si no es *Postback* hace el llenado del dropdownlist “comboclasif” que es el que llevará los datos de la clasificación de los productos. Después de llenar este dropdownlist se manda llamar la subrutina ***BindProducto*** que es la que llenará el dropdownlist “comboprod” con los datos de los productos referentes a la clasificación activa en el *comboclasif*. Se cierra la conexión y manda llamar a la subrutina ***bindexist***. Esta subrutina trae la *existencia* de la cantidad de producto activo en el *comboprod* y le asigna el dato a la etiqueta “prod” para que la muestre en pantalla. Y finalmente cierra la conexión total.

La subrutina ***tabla*** es la siguiente:

ACCESS

```
Public Sub tabla()
        Adaptador = New System.Data.OleDb.OleDbDataAdapter("select
                    NomProducto as [Producto],Cantidad from
                    CompraVirtual where NomUsuario= '" & usr & "' and
                    Fecha= ‘" & fecha1 & "’ order by NomProducto", Cn)
        ds2 = New System.Data.DataSet
        Adaptador.Fill(ds2, "NomTab")
        DataGrid1.DataSource = ds2.Tables("NomTab").DefaultView
        DataGrid1.DataBind()
End Sub
```

SQL

```
Public Sub tabla()
        Adaptador = New System.Data.OleDb.OleDbDataAdapter("select
                    NomProducto as [Producto],Cantidad from
                    CompraVirtual where NomUsuario= '" & usr & "' and
                    Fecha= #" & fecha & "# order by NomProducto", Cn)
        ds2 = New System.Data.DataSet
        Adaptador.Fill(ds2, "NomTab")
        DataGrid1.DataSource = ds2.Tables("NomTab").DefaultView
        DataGrid1.DataBind()
End Sub
```

La diferencia entre la de Access y la SQL es la línea de búsqueda que se va a hacer en la parte de *Fecha*. La restricción de fecha para Access se tiene que hacer con el signo **#**, y para SQL es con el signo **‘**. Todo lo demás es igual.

La subrutina ***BindProducto*** es la siguiente:

```
Public Sub BindProducto()

    commSQL2 = New OleDb.OleDbDataAdapter("select
               idProducto,NomProducto,Existencia from Producto Where
               idClasif=" & comboclasif.SelectedItem.Value, Cn)
    commSQL2.Fill(ds, "Producto")

    comboprod.DataSource = ds.Tables("Producto").DefaultView
    comboprod.DataTextField = "NomProducto"
    comboprod.DataValueField = "idProducto"
    comboprod.DataBind()
End Sub
```

La subrutina ***bindexist*** es la siguiente:

```
Public Sub bindexist()
    Dim queryexistencia As System.Data.OleDb.OleDbCommand
    Dim querycommand As New System.Data.OleDb.OleDbCommand

    Cn.Open()
    queryexistencia = New OleDb.OleDbCommand("Select existencia from
                      Producto where idProducto= " &
                      comboprod.SelectedItem.Value, Cn)
    datRead = queryexistencia.ExecuteReader
    While datRead.Read
        prod.Text = datRead.Item("existencia")
    End While
    datRead.Close()
End Sub
```

Cuando se selecciona otro valor en el *comboclasif* entra a propiedad *SelectedIndexChanged* del dropdownlist y manda llamar la subrutina *BindProducto*, la subrutina *bindexist* y la subrutina ***limpiar***. Esta última hace invisibles las etiquetas de error y le asigna el menor valor a la caja de texto, o sea 1.

```
Private Sub comboclasif_SelectedIndexChanged(ByVal sender As
               System.Object, ByVal e As System.EventArgs) Handles
               comboclasif.SelectedIndexChanged
    BindProducto()
    bindexist()
    limpiar()
End Sub
```

La subrutina ***limpiar*** es la siguiente:

```
Public Sub limpiar()
    exist.Visible = False
    exist2.Visible = False
    err.Visible = False
    cantidad.Text = 1
End Sub
```

De igual manera, cuando se selecciona otro valor en el *comboprod* entra a la propiedad *SelectedIndexChanged*, en la cual manda llamar a las subrutinas *bindexist* y *limpiar*.

```
Private Sub comboprod_SelectedIndexChanged(ByVal sender As Object, ByVal
                e As System.EventArgs) Handles
                comboprod.SelectedIndexChanged
    bindexist()
    limpiar()
End Sub
```

Cuando se da clic sobre el botón “btncomprar” manda llamar a la subrutina ***verifexist*** y a la subrutina ***insertar***.

```
Private Sub btncomprar_Click(ByVal sender As System.Object, ByVal e As
                        System.EventArgs) Handles btncomprar.Click
    verifexist()
    insertar()
End Sub
```

La subrutina ***verifexist*** hace una búsqueda en la base de datos para conocer la existencia del producto que esta activo en el *comboprod* y checa que esta cantidad no sea menor a la cantidad escrita en el cuadro de texto.

Si es la cantidad escrita en la caja de texto es mayor a la que hay en existencia pone visible la etiqueta de error “exist”. Esta etiqueta tendrá el siguiente texto: *Estas escribiendo una cantidad mayor de la que hay en existencia, corrige la cantidad por favor*.

Si no es mayor el valor en la caja de texto a la de existencia deja esta etiqueta invisible. Y es la siguiente:

```
Public Sub verifexist()
    Dim queryexistencia As System.Data.OleDb.OleDbCommand
    Cn.Open()
    queryexistencia = New OleDb.OleDbCommand("Select existencia from
                  Producto where idProducto= " &
                  comboprod.SelectedItem.Value, Cn)
    datRead = queryexistencia.ExecuteReader

    While datRead.Read
        If cantidad.Text > datRead.Item("existencia") Then
              exist.Visible = True
         Else
              exist.Visible = False
         end if
    End While
    datRead.Close()
    Cn.Close()
End Sub
```

La subrutina ***insertar*** es la que va a hacer que se inserten los datos seleccionados en la página.

```
Public Sub insertar()
    Dim insertcommand As System.Data.OleDb.OleDbCommand
    Dim querycommand As New System.Data.OleDb.OleDbCommand
    Dim insertdatos As String
    Dim id As Integer
    Dim suma As Integer

    If exist.Visible = False Then
       id = 0
       suma = 0
```

ACCESS

```
querycommand = New OleDb.OleDbCommand("select idCompra,
                        cantidad from CompraVirtual where
                        NomUsuario= '" & usr & "' and Fecha=
                        #" & fecha1 & "# and NomProducto= '"
                        & comboprod.SelectedItem.ToString &
                        "' order by idCompra", Cn)
```

SQL

```
querycommand = New OleDb.OleDbCommand("select idCompra,
                        cantidad from CompraVirtual where
                        NomUsuario= '" & usr & "' and Fecha=
                        ‘" & fecha1 & "’ and NomProducto= '"
                        & comboprod.SelectedItem.ToString &
                        "' order by idCompra", Cn)

       Cn.Open()

       datRead = querycommand.ExecuteReader
       While datRead.Read
           id = datRead.Item("idCompra")
           suma = datRead.Item("cantidad")
       End While
```

```
        suma = suma + cantidad.Text
        datRead.Close()
        Cn.Close()
        If suma > prod.Text Then
            exist2.Visible = True
        Else
            If id = 0 Then
                Cn.Open()
                querycommand = New OleDb.OleDbCommand("select idCompra
                               from CompraVirtual", Cn)

                datRead = querycommand.ExecuteReader
                While datRead.Read()
                    max = datRead.Item("idCompra").ToString + 1
                End While
                datRead.Close()
                Cn.Close()

                insertdatos = "insert into
                 CompraVirtual(NomUsuario,NomClasificacion,idProducto,Nom
                 Producto,Cantidad,Fecha) values ('" & usr & "','" &
                 comboclasif.SelectedItem.ToString & "'," &
                 comboprod.SelectedValue & ",'" &
                 comboprod.SelectedItem.ToString & "'," + cantidad.Text +
                 ",'" & fecha1 & "')"
                insertcommand = New
                          System.Data.OleDb.OleDbCommand(insertdatos, Cn)
                insertcommand.Connection.Open()
                Try
                    insertcommand.ExecuteNonQuery()
                    insertcommand.Connection.Close()
                    Page.Response.Redirect("comprar.aspx")
                Catch Exp As System.data.OLEDB.OleDbException
                    err.Visible = True
                    err.Text = "ERROR: no se pudo agregar el registro,
                                compruebe que los campos están rellenos
                                correctamente"
                End Try
                insertcommand.Connection.Close()
            Else
                insertdatos = "update CompraVirtual set
                             cantidad=cantidad+ " & cantidad.Text & "
                             where idCompra= " & id
                insertcommand = New
                          System.Data.OleDb.OleDbCommand(insertdatos, Cn)
                insertcommand.Connection.Open()
                Try
                    insertcommand.ExecuteNonQuery()
                    insertcommand.Connection.Close()
                    Page.Response.Redirect("comprar.aspx")
                Catch Exp As System.data.OLEDB.OleDbException
                    err.Visible = True
                    err.Text = "ERROR: no se pudo agregar el registro,
                                compruebe que los campos están rellenos
                                correctamente"
                End Try
                insertcommand.Connection.Close()
            End If
        End If
    End If
End Sub
```

Como se ve en el código condiciona la acción, si no esta visible la etiqueta de error “exist” entra. Primero asigna a las variables *id* y *suma* los resultados, si es que ya existieran, del número de control de compra del producto seleccionado y la cantidad. Esto es por si ya se había seleccionado un producto con cierta cantidad, y después de haber seleccionado ese producto volviera seleccionar el mismo, entonces se sumaría la cantidad que ya se había seleccionado más la que se esta poniendo en la caja de texto. Entonces si la suma de estas cantidades es mayor a la cantidad en existencia (que es la cantidad que tendrá como valor la etiqueta “prod”) pondrá visible la etiqueta de error “exist2”. Esta etiqueta tendrá el siguiente texto: *Estas tratando de comprar más productos de lo que hay en existencia, corrige la cantidad por favor*. Si la cantidad no es mayor a la de existencia podrá realizar 2 cosas. La primera, si es que el valor de la variable “*id*” sigue en “0” es que no se había comprado el producto seleccionado, por lo tanto se va a hacer un *INSERT*. Primero se hace una búsqueda del *idCompra* mayor para aumentarle 1 y asignárselo a la variable max para que vaya insertando los números consecutivos en la tabla de la base de datos (recordando que en Access se puso autonumérico pero en SQL no, por lo tanto hay que unificarlos).

Ya que se tienen todos los valores que se van a insertar, se hace la inserción. Si no ocurre ningún error se redirecciona a sí misma la página, y si ocurre un error muestra el mensaje de error en la etiqueta “err”.

Pero si fue el caso de que el producto seleccionado ya había sido comprado entra al *else* para hacer un *UPDATE*. Se hace la suma de las cantidades y se ejecuta la inserción. De la misma manera que con el insert, si no ocurre ningún error se redirecciona a si misma y si ocurre muestra el mensaje de error.

Sólo falta ver lo que tiene el código de los otros 2 botones, que como se explicó, solo redirecciona a sus respectivas páginas.

```
Private Sub Borrar_Click(ByVal sender As System.Object, ByVal e As
                         System.EventArgs) Handles Borrar.Click
    Response.Redirect("borrar.aspx")
End Sub

Private Sub Terminar_Click(ByVal sender As System.Object, ByVal e As
                         System.EventArgs) Handles Terminar.Click
    Response.Redirect("jeadeca.aspx")
End Sub
```

4.2.2.7. borrar.aspx

La página ***borrar.aspx*** servirá para borrar cantidades de producto de la base de datos.

Tendrá el encabezado, la etiqueta de bienvenida, 2 etiquetas de error, etiquetas para identificar los elementos, 1 Drowdownlist, 1 caja de texto, 1 RequiredFieldValidator, 1 DataGrid y 3 botones. Quedará con el siguiente diseño:

Figura 42. Diseño de *borrar.aspx*

La etiqueta de bienvenida y la etiqueta “err” es exactamente igual a la página anterior. Se pone una etiqueta como título de “Selecciona el Producto” para el DropDownList que se llamará “comboprod” que mostrará la lista de los productos que van comprados. Debajo la caja de

texto que servirá para colocar la cantidad del producto que se va borrar, y debajo de esta el RequiredFieldValidator que controle a la caja de texto.

Debajo de esto esta el botón que se llamará “btnborrar”. Un botón que se llamará “terminar” que redireccionará la página de vuelta a *jeadeca.aspx*, y otro botón arriba del Datagrid que se llamará “borrar” que redireccionará la página a la página *comprar.aspx*.

El código es similar al de *comprar.aspx*, primero se crean las diferentes variables que se van a utilizar.

```
Dim commSQL As OleDb.OleDbDataAdapter
Dim ds, ds2 As New DataSet
Dim fecha As Date
Dim usr As String
Dim cantcero As Integer
Dim adaptador As System.Data.OleDb.OleDbDataAdapter
Dim datRead As System.Data.OleDb.OleDbDataReader

Public Cn As New System.data.OleDb.OleDbConnection
```

En la parte de Page_Load se pone la fecha actual en la variable fecha y se hace la condición de seguridad, si entra da la bienvenida, hace la conexión a la base de datos, coloca el Nick del usuario en la variable “usr” y llena el Datagrid con la subrutina ***tabla*** que es igual a la anterior.

```
Public Sub tabla()
```

ACCESS

```
adaptador = New System.Data.OleDb.OleDbDataAdapter("select
                    NomProducto as [Producto],Cantidad from
                    CompraVirtual where NomUsuario= '" & usr & "'
                    and Fecha= #" & fecha & "# order by
                    NomProducto", Cn)
```

SQL

```
adaptador = New System.Data.OleDb.OleDbDataAdapter("select
                    NomProducto as [Producto],Cantidad from
                    CompraVirtual where NomUsuario= '" & usr & "'
                    and Fecha= '" & fecha & "' order by
                    NomProducto", Cn)
```

```
    ds2 = New System.Data.DataSet

    adaptador.Fill(ds2, "NomTab")
    Datagrid1.DataSource = DS2.Tables("NomTab").DefaultView
    Datagrid1.DataBind()
End Sub
```

Si no es *Postback* hace el llenado del dropdownlist “comboprod” que es el que llevará los datos de los productos que se han comprado. Después de llenar este dropdownlist se cierra la conexión.

Al presionar el botón “btnborrar” manda llamar las subrutinas ***verifexist*** y ***eliminar***.

```
Private Sub btnborrar_ServerClick(ByVal sender As System.Object, ByVal
                    e As System.EventArgs) Handles btnborrar.ServerClick
    verifexist()
    eliminar()
End Sub
```

La subrutina ***verifexist*** checa que la cantidad escrita en la caja de texto no sea mayor a la cantidad comprada. Si es mayor, pone visible la etiqueta de error “exist”, y si no la deja invisible.

```
Public Sub verifexist()
    Dim queryexistencia As System.Data.OleDb.OleDbCommand
    Cn.Open()
    queryexistencia = New OleDb.OleDbCommand("Select cantidad from
                       CompraVirtual where idCompra= " &
                       comboprod.SelectedValue, Cn)
    datRead = queryexistencia.ExecuteReader
    While datRead.Read
        cantcero = datRead.Item("cantidad")
        If cantidad.Text > datRead.Item("cantidad") Then
              exist.Visible = True
        Else
              exist.Visible = False
        End if
    End While
    datRead.Close()
    Cn.Close()
End Sub
```

La subrutina ***eliminar*** es la que borra las cantidades de producto que se escriban en la página, y si es que es la totalidad de las cantidades, elimina la compra del producto seleccionado.

```
Public Sub eliminar()
    Dim deletecommand As System.Data.OleDb.OleDbCommand
    Dim querycommand As New System.Data.OleDb.OleDbCommand
    Dim deletedatos As String
    Dim diferencia As Integer

    diferencia = cantcero - cantidad.Text
    If exist.Visible = False Then
```

```
            If diferencia = 0 Then
                deletedatos = "delete From CompraVirtual where idCompra=" &
                              comboprod.SelectedValue
                deletecommand = New
                              System.Data.OleDb.OleDbCommand(deletedatos, Cn)
                deletecommand.Connection.Open()
                Try
                    deletecommand.ExecuteNonQuery()
                    deletecommand.Connection.Close()
                    Page.Response.Redirect("borrar.aspx")
                Catch Exp As System.data.OLEDB.OleDbException
                    err.Visible = True
                    err.Text = "ERROR: no se pudo agregar el registro,
                               compruebe que los campos están rellenos
                               correctamente"
                End Try
                deletecommand.Connection.Close()
            Else
                deletedatos = "update CompraVirtual set cantidad= cantidad
                               - " & cantidad.Text & " where idCompra=" &
                               comboprod.SelectedValue
                deletecommand = New
                              System.Data.OleDb.OleDbCommand(deletedatos, Cn)
                deletecommand.Connection.Open()
                Try
                    deletecommand.ExecuteNonQuery()
                    deletecommand.Connection.Close()
                    Page.Response.Redirect("borrar.aspx")
                Catch Exp As System.data.OLEDB.OleDbException
                    err.Visible = True
                    err.Text = "ERROR: no se pudo agregar el registro,
                               compruebe que los campos están rellenos
                               correctamente"
                End Try
                deletecommand.Connection.Close()
            End If
        End If
    End Sub
```

Como se ve en el código, asigna a la variable “diferencia” la resta de la cantidad que se ha comprado del producto menos la cantidad escrita en la caja de texto. Luego condiciona la entrada al código, si no esta visible la etiqueta de error “exist” entra.

Luego condiciona de nuevo el accionar del código, si la variable “diferencia” es igual a cero, quiere decir que se va a borrar todo el producto, por lo tanto va a hacer un *DELETE*.

Borra la compra hecha en donde se encuentra el producto seleccionado. Si no ocurre algún error se redirecciona la página a sí misma, y si ocurre algún error muestra el mensaje de error en la etiqueta “err”.

Pero si la variable “diferencia” no es igual a cero entonces hace un *UPDATE*. En donde solamente restará la cantidad escrita en la caja de texto con la cantidad que ya se había comprado. De igual manera si no ocurre algún error se redirecciona a sí misma, y si ocurre error muestra el mensaje de error en la etiqueta “err”.

Falta poner el código de redireccionamiento de los botones correspondientes a cada una de las páginas que va a mandar.

```
Private Sub btncomprar_Click(ByVal sender As System.Object, ByVal e As
                             System.EventArgs) Handles btncomprar.Click
    Response.Redirect("comprar.aspx")
End Sub

Private Sub Terminar_Click(ByVal sender As System.Object, ByVal e As
                             System.EventArgs) Handles Terminar.Click
    Response.Redirect("jeadeca.aspx")
End Sub
```

CONCLUSION

Al concluir con el desarrollo la página Web dinámica se pudo confirmar lo fácil que se puede acceder a la información que se localiza en la base de datos de la empresa, y que ésta información además de poder visualizarla, se le puede dar el mejor uso.
Como se vio, se pudieron hacer compras desde la página Web, lo único que tiene que decidir el negocio o empresa es cómo el cliente va a realizar el pago por las compras que se hagan en la página. Pueden ser varias opciones para pago: las puede pagar directamente con una línea bancaria, o por medio de depósitos, o inclusive en efectivo directamente con el negocio, de igual manera para cada uno de los casos el dueño del negocio tiene el acceso a la información de las compras que hizo cada cliente, y éstas puede identificarlas en su sistema de ventas por medio de informes, o como se tenga el negocio configurado para ver los resultados del contenido de su base de datos.

Lo que no se pudo cerciorar con esta hipótesis es el resultado específico de la ganancia (cantidad monetaria) que tuvo la empresa al implementar esta tecnología, ya que para poder tener ese tipo de información se debería de echar a andar la página de la empresa, y para esto se tienen que hacer otras cuestiones, como subir la página en un servidor de Internet (HOST) y que la gente comenzara a comprar los productos para poder monitorear las ganancias.
Pero así como se ha visto en los diferentes casos de mejoras de tecnologías, como Cajeros automáticos o Puntos de Venta en tienditas y supermercados, siempre han ayudado a la mejora en el trabajo, disminuyendo tiempos y aumentando ganancias.

Realizar una página Web dinámica para el programador va a ser tan sencillo como él lo decida, ya que puede haber páginas con requerimientos muy simples que no tenga gran chiste desarrollar, pero también la suposición de que una empresa con grandes requerimientos informáticos pueda solicitar una página Web muy compleja implica que el

trabajo de realizarla va a ser muy tardado y con más funciones y elementos programables en la página. Por estas razones es que confirmo que el trabajo de un programador Web es muy importante y a veces nada sencillo de hacer.

Me costó mucho tiempo y esfuerzo realizar todo el trabajo de investigación y desarrollar esta tesis, muchas horas y muchas semanas que para mi valen la pena, porque con este tipo de información espero ahorrarle tiempo a otra persona que apenas este descubriendo la tecnología Web.
Es decisión tuya que lees esta tesis comenzar a utilizar estas bases de programación Web, te ayudará mucho a simplificar tu trabajo y más si eres principiante en la creación de páginas Web dinámicas.

BIBLIOGRAFIA

COSENTINO, Christopher, ***PHP***, Editorial Pearson Educación, Madrid, España, 2001, 188 pp.

DELGADO, Albert, ***Microsoft SQL Server 2000. Edición especial***, Editorial Pearson Educación, España, 2001, 926 pp.

DUTHIE, G. Andrew, ***Microsoft ASP Programming with Microsoft Visual Basic .NET Version 2003 Step By Step***, Editorial Microsoft Press, Estados Unidos, 2003, 600 pp.

PÉREZ, Cesar, ***Dreamweaver MX 2004. Desarrollo de páginas Web dinámicas con PHP y MYSQL***, Editorial Ra-Ma, México, D.F., 2004, 528 pp.

R. BÜHLER, Erich, ***Visual Basic .NET***, Editorial McGraw-Hill, España, 2001, 950 pp.

STOPFORD, Andrew, ***Guía avanzada programación de PHP para Windows***, Editorial Pearson Educación, Madrid, España, 2002, 340 pp.

OTRAS FUENTES

http://cdec.unican.es
http://es.wikipedia.org
http://es.gotdotnet.com
http://www.aspfacil.com
http://www.aui.es
http://www.aunmas.com
http://www.cursodephp.com
http://www.desarrolloweb.com
http://www.enterate.unam.mx
http://www.estrucplan.com.ar
http://www.forosdelweb.com
http://www.itlp.edu.mx
http://www.maestrosdelweb.com
http://www.marketalia.com
http://www.programacion.com
http://www.wampserver.com