UNIVERSIDAD NACIONAL AUTONOMA DE NICARAGUA, LEON

Facultad de Ciencias y Letras

INTEGRACIÓN DE BASES DE DATOS EN EL WEB

MONOGRAFÍA PARA OPTAR AL TÍTULO DE LICENCIADO EN COMPUTACIÓN

ELABORADO POR:

- **Br. Eduardo Santiago Molina Poveda.**

TUTOR:

- **Ing. Ricardo Espinoza**

2001

AGRADECIMIENTO

Te agradezco a ti Señor y Dios mío, autor de todo bien, quien con tu sabiduría y fortaleza me has permitido culminar mis estudios universitarios, y lograr así alcanzar la meta propuesta desde hace mucho tiempo, gracias por tu presencia en mi vida, por estar en los momentos de éxito y los momentos difíciles. También te doy gracias a ti María Santísima, por tu presencia incondicional en mi vida.

Doy gracias también a mi papá, Felipe Santiago Molina Vásquez, a mi mamá, Nohemí Poveda de Molina. A mi papá, porque ha sido una persona motivadora, por haber dado lo mejor de sí para mi formación, porque ahora puedes, papá, ver alcanzado uno de tus sueños. A mi mamá, quien me ha enseñado el sentido de la perseverancia, por apoyarme y acompañarme en mi vida, en todo.

Al profesor, Ing. Ricardo Espinoza, puesto que él me impulsó a terminar este proyecto y fue quien me hizo iniciarlo, ya que antes de que yo pensara cambiar el tema, hizo que lo retomara teniendo su contribución como tutor.

A mis amigos y compañeros de estudio, a los que me ayudaron con algunas ideas sobre el proyecto.

DEDICATORIA

A mi papá, a mi mamá y hermanos, Rommel, Jorge y Omayra,
especialmente a estos últimos, ellos que
vienen detrás y que desean también
un día alcanzar sus metas propuestas.

INDICE

APÉNDICE

Apéndice A: Variables recibidas por un CGI desde su entorno

Basado en parte en el documento http://hoohoo.ncsa.uiuc.edu/cgi/env.html.

SERVER_SOFTWARE
Formato: Nombre/versión

El nombre y versión del servidor de información (Servidor Web) que está atendiendo al visitante y que es el padre del proceso actual (CGI).

SERVER_NAME

El nombre del host (computador donde está el CGI), su nombre en el DNS, o la dirección IP del mismo si el CGI ha sido llamado con este.

GATEWAY_INTERFACE
Formato: CGI/revision

La revisión de la especificación para CGIs con la que cumple el servidor.

SERVER_PROTOCOL
Formato: protocolo/revisión

El nombre y revisión del protocolo de información con que se ha generado la llamada al CGI.

SERVER_PORT

El número de la puerta con que fue hecha la llamada CGI.

REQUEST_METED

El método con que fue hecha la llamada. Para el protocolo http contiene “GET”, “HEAD”, “POST”, etc.

PATH_INFO

Información del path extendida. Los scripts (CGIs) pueden ser llamados usando su path virtual seguido de información extra. Esta información es enviado como PATH_INFO. La misma debe ser decodificada por el servidor si viene desde una URL antes de ser pasada al CGI.

PATH_TRANSLATED

El servidor entrega una versión traducida del PATH_INFO, que toma el path y hace las traducciones correspondientes, entregando al CGI

la dirección correspondiente en el sistema de archivos en que está alojado.

SCRIPT_NAME

El path virtual con que el CGI está siendo ejecutado.
Usado para autoreferenciarse.

QUERY_STRING

La información que sigue al ? en las llamadas a un CGI.
Esta es la información de consulta o parámetros, no está decodificada y es labor del CGI hacerlo.

REMOTE_HOST

El nombre del host que hace la solicitud. Si el servidor no tiene la información, entrega la variable REMOTE_ADDR.

REMOTE_ADD

La dirección IP del host que hace la solicitud.

AUTH_TYPE

Si el servidor soporta autentificación de un usuario y el CGI está protegido, contiene el protocolo específico de autentificación usado para validar al usuario.

REMOTE_USER

Si el servidor soporta autentificación de usuario y el CGI está protegido, contiene nombre del usuario que está accediendo al cgi.

REMOTE_IDENT

Si el Servidor Web soporta identificación RFC 931, está variable entrega el nombre del usuario remoto tomado desde el servidor.

CONTENT_TYPE

Para llamadas que llevan información agregaa, como en el caso de las llamadas con POST o PUT, contiene el tipo de los datos.

CONTENT_LENGTH

El largo de los datos enviados por el cliente.

Adicionalmente el cliente puede enviar líneas extra en la forma de encabezados, los cuales son puestos en variables de ambiente con el prefijo HTTP_ seguido por el nombre del header.
Cualquier carácter – en el encabezado es convertido en _.
El servidor puede excluir los encabezados que ya han sido procesados, como Authorization o Content_type. También podría excluir algunos encabezados si estos no cumplieran en el espacio reservado para variables de ambiente.

Algunos ejemplos recibidos comúnmente son (no hay garantía de que se reciban siempre, ya que dependen del cliente):

HTTP_ACCEPT
Formato: tipo/subtipo[, tipo/subtipo[...]]
El tipo de datos MIME que el cliente (HTTP) puede aceptar.

HTTP_USER_AGENT
Formato (general): software/version librería/versión
El navegador que está haciendo la solicitud.

HTTP_COOKIE
Las cookies que han sido grabadas por el previamente.

HTTP_REFERER
El documento desde el cual se llamó al CGI.

HTTP_UA_CPU
El tipo de procesador usado por el cliente.

HTTP_UA_COLOR
Indica cómo el usuario visualiza colores.

HTTP_UA_PIXELS
Indica la resolución de pantalla del usuario.

Apéndice B: Métodos de Clases DBI

Métodos Generales de Clases DBI

Nombre	Uso	Descripción
Connect()	$dbHandle = DBI->connect(DBI: $driverName, $dataSource, $usrName, $password); $dbHandle=DBI->connect($dataSource, $usrName, $passwrord, $driverName);	Este método establece una conexión a la fuente de datos especificada y retorna un objeto identificador (manejador a la base de datos). Hay que notar que si se define el nombre del driver (controlador) como cuarto parámetro no se necesitará la marca DBI:$driverName al lado de $dataSource.
Available_drivers()	@drivers=DBI->available_drivers() ; @drivers=DBI->vailable_driver ($warnFlag);	Como se ve, este método devuelve un arreglo (matriz) de controladores que el módulo puede ver. Obtiene la lista mediante una búsqueda a través del arreglo @INC para módulos DBI. Si pasa un valor verdadero como parámetro, los avisos acerca de controladores ocultos serán omitidos.
data_sources()	@dataSources=DBI->data_sources ($driver);	Devuelve una lista de fuente de datos disponibles para un controlador determinado, en caso que el controlador soporte el método.
Trace()	DBI->trace($leve); DBI->trace($level, $file);	Provee la ruta functional de como el archivo es abierto. Un nivel 0 deshabilita la ruta y un nivel 2 provee la ruta detallada.

Neat()	$neatvalue=DBI::neat($value,$MaxLength);	Formatea el valor especificado como cadena de aracteres entre comillas, los valores no definidos los reemplaza con “undef”, aracteres no imprimibles con una “.”, y las cadenas son truncadas y finalizadas con “...” si es más grande que $MaxLength
Neat_list()	$delimitedList=DBI::neat_list(\@lisRef, $maxLength, $delimited);	Formatea un arreglo por cada elemento llamando a neat() y formando una cadena uniendo los elementos con el delimitador especificado.
Dum_results()	$dbRows=DBI::dump_results($statementHandle, $maxLength, $lineSeparator, $fieldSeparator, $fileHandle);	Obtiene todas las filas desde el objeto manejador de declaración, llama a neat_list() en cada fila e imprime el resultado hacia el manejador de archivo ($fileHandle) que es por defecto.

Métodos Generales de Manejadores

Nombre	Uso	Descripción
Error()	$result=$handle->error();	Obtiene el código de error devuelto desde la base de datos.
Errstr()	$result=$handel->errstr();	Obtiene el mensaje de erro devuelto desde la base de datos.
State()	$state=$andel->state();	Obtiene el código de error del estado del Sql.

Métodos de Manejadores de Base de Datos

Nombre	Uso	Descripción
Prepare()	$statementHandle=$dataSourceHandle->prepare ($sql);	Prepara consulta para ejecución.
Do()	$rowCount=$databaseHandle->do();	Prepara y ejecuta una sentencia SQL y devuelve el número de filas retornadas.
Commit()	$rowCount=$databaseHandle->commit();	Hace permanente el último cambio realizado a la base de datos en caso de soportar el método.
Rollbacl()	$rowCount=$databaseHandle->rollback();	Deshace el cambio hecho por commit, si es soportado.
Disconnect()	$rowCount=$databaseHandle->disconnect();	Desconecta de la base de datos.
Ping()	$rowCount=$databaseHandle->ping();	Prueba si la base de datos en el servidor está disponible.
Quote()	$sql=$databaesHandle->quote($string);	Excelente cadena para una consulta SQL.

Método para Manejadores de Consultas

Nombre	Uso	Descripción
Execute()	$value=$statementHandle->execute();	Ejecuta una consulta preparada.
Fetchrow_arrayRef()	$arrayRef=$statementHandle-> fetchrow_arrayRef();	Obtiene la siguiente fila de datos como una referencia a un arreglo, teniendo los valores de las columnas.
Fetchrow_array()	@array=$statementHandle- >fetchrow_array();	Obtiene la siguiente fila de datos en el cual las claves a las que referencia son los nombres de las columnas y los valores son los valores de las columnas.
Fetchall_arrayref()	$reference=$statementHandle- >fetchall_array ();	Obtiene todas las filas como referencias en un arreglo referenciado.

Apéndice 3: Métodos, Propiedades y Colecciones de los Objetos Recordset, Connection, Field, Parameter, Property y Error.

INTRODUCCIÓN

En este Apéndice se analizan todos los métodos y propiedades de los objetos ADO.

MODALIDADES DEL OBJETO RECORDSET

Un objeto RecordSet se puede abrir con muchos cursores diferentes. Cada cursor permite realizar un acceso distinto a los datos del objeto RecordSet, para no limitar a un simple acceso secuencial. Para utilizar esta característica del objeto RecordSet se debe crear primero una instancia del objeto y después abrirlo con el cursor que deseado.

Al crearlo, antes de darle ningún valor o asignarlo a una tabla, podemos decir las restricciones que queremos aplicarle.

Una de las primeras restricciones que podemos aplicar son los cursores. Un cursor nos indica de qué forma podemos recorrer un objeto RecordSet. Es de este modo como conseguimos utilizar un RecordSet de otra forma que no sea la secuencial. Existen cuatro cursores para abrir un RecordSet

- **AdOpenForwandOnly**: Este cursor sólo nos permite recorrer el objeto de forma secuancial. Con este cursor nos podemos filtrar la tabla. Este es el que aparece por defecto.

- **AndOpenKeySet**: Con este cursor nos podemos mover para adelante y para atrás en el objeto RecordSet. Además, si en algún momento otro usuario modifica o borra algún registro de la tabla, se reflejará un en el objeto RecordSet. No así si añade un nuevo registro.

- **AnOpenDinamic**: Nos podemos mover en cualquier dirección y además veremos cualquier modificación, borrado o inserción de registros que haya realizado un nuevo usuario.

- **AnOpenStatic**: Nos podemos mover en todas direcciones, pero, no se verá ningún cambio que haya realizado otro usuario sobre la tabla.

Con estos cursores del objeto RecordSet podremos recorrer la tabla de a nuestro gusto. Si solamente vamos a recorrer la tabla secuencialmente, es mejor utilizar el cursor por defecto, ya que es el más rápido. En cualquier otro caso utilizaremos el que mejor de adapte a nuestras exigencias.

La forma de aplicar cualquiera de estos cursores es creando dos objetos, uno del tipo Connection y otro RecordSet. Abrimos primero el objeto Connection con su método open, indicándole la base de datos a la que se va a referir. Por último, utilizaremos el método open del objeto RecordSet para abrir éste, es decir, para darle un valor. Deberemos indicar la selección que queremos hacer sobre la base de datos y el objeto Connection que nos une con la base de datos. Además, podremos fijar de forma opcional algunas restricciones. Entre ellas está el tipo de cursor a utilizar.

```
Ob_RS.Open Sql Ob_Conn.cursor.
```

Otro atributo que podemos utilizar a la hora de abrir un objeto RecordSet es el cerrojo. El cerrojo nos permite controlar el acceso que van a tener el resto de los usuarios de una base de datos mientras yo lo estoy utilizando. Será muy útil cuando utilicemos bases de datos muy concurridas, ya que evitaremos la inconsistencia de los datos.

Como en el caso de los cursores, tendremos una serie de cerrojos que nos impondrán una mayor o menor restricción. Siempre es conveniente utilizar el más restrictivo que se adapte a nuestras exigencias. Los posibles cerrojos son los siguientes:

- **adLockReadOnly**: Indica que el usurario no podrá modificar los registros de la tabla. Una vez que abrimos la atabla, ésta no podrá ser modificada. Este es el que aparece por defecto.
- **adLockPessimistic**: Indica que cuando el usuario abra una base de datos, nadie más podrá hacerlo al mismo tiempo. Cuando la cierre se abrirá el cerrojo de los demás usuarios. De este modo nos aseguramos la integridad de los datos.
- **adLockOptimistic**: Indica que la base de datos será cerrada a los demás usuarios cuando se esté realizando la operación Update del objeto RecordSet, De este modo la base de datos quedará bloqueada mucho menos tiempo. Si nos aseguramos que todas las páginas Web tienen esta restricción, entonces conseguiremos el mismo efecto que con el cerrojo anterior.
- **adLockBatchOptimistic**: Indica que el registro se actualizará en modo batch.

Para aplicar estos cerrojos utilizaremos la misma técnica que cuando utilizábamos cursores. Al abrir el objeto RecordSet aplicaremos esta restricción para asegurarnos de la integridad de nuestros datos en casos de concurrencia.

```
Ob_RS.OpenSql.Ob_Conn.cursor,cerrojo
```

Cuando abrimos el objeto RecordSet, entre otras cosas, debemos indicar la tabla con la que queremos trabajar. Esto lo podemos decir de varias formas, con el nombre de la tabla, con la sentencia SQL o con un procedimiento almacenado. Si no indicamos nada será el objeto quien averigüe lo que estamos pidiendo.

Para facilitar el trabajo a este objeto existe el parámetro Option. Este parámetro que fijaremos al abrir dicho objeto, nos sirve para decir de qué forma indicamos la tabla que queremos abrir. De este modo facilitaremos el trabajo a nuestra aplicación y aumentaremos la velocidad de respuesta. Existen tres formas de indicar qué tabla queremos, las cuales están representadas por las siguientes constantes:

- **adCmdTable**: Lo usaremos cuando indiquemos la tabla que queremos utilizar, con el nombre de la misma.

- **adCmdText**: La utilizaremos cuando escribamos una sentencia SQL para recoger los datos de la tabla.

- **adCmdStoredProc**: lo utilizaremos cuando indiquemos los datos que necesitemos con un procedimiento almacenado.

- **AdCmdUnknown**: Ésta es la opción por defecto e indica que el objeto RecordSet se debe encargar de averiguar por sí mismo el significado de la expresión.

Dependiendo de lo que queramos obtener utilizaremos una u otra expresión. De esta forma, si queremos utilizar una tabla entera, lo más cómodo es llamarla por su nombre. Sin embargo, si queremos filtrar esa información antes de utilizarla deberemos indicarlo con una sentencia SQL. Si deseamos operaciones más complicadas tendremos que utilizar procedimientos almacenados.

METODOS Y PROPIEDADES PARA MOVERNOS EN UN RECORDSET

Ya hemos visto cómo podemos forzar un RecordSet que nos permita movernos a nuestro gusto por él. Ahora vamos a analizar los métodos y propiedades que utilizaremos para movernos. Los métodos nos servirán para realizar el movimiento en sí y las propiedades para controlar los límites del movimiento. Los métodos que tenemos son los siguientes:

- **Move**: Nos traslada un número de registros tanto para arriba como para abajo. Ese número lo indicaremos a continuación. Si este es negativo subiremos en el objeto RecordSet, si es positivo bajaremos. Ej Ob_RS.Move 3

- **MoveFirst**: Nos lleva al primer registro. Nos servirá cuando necesitemos hacer más de un recorrido por la tabla.

- **MoveNext**: Mueve el puntero del objeto RecordSet una posición hacia el final de mismo. Est método es el que hemos venido utilizando normalmente. Nos será útil cccuando recorramos secuencialmente la tabla.

- **MoveLast**: Nos lleva al último registro de la tabla. Será de utilidad cuando queremos recorrer la tabla de forma inversa o simplemente para ver el último registro.

- **MovePrevious**: Mueve el puntero del objeto RecordSet una posición hacia el principio de mismo. Es el contrario de MoveNext, ya que nos será útil cuando recorramos de abajo a arriba la tabla.

Estos métodos no nos servirán de nada si no tuviéramos unas propiedades que nos controlaran su uso. Las propiedades de este objeto nos servirán para controlar la situación del puntero, es decir, cuál es el registro que vamos a leer. De este modo podremos recorrer un objeto RecordSet sin salirnos de sus límites. Las propiedades de este objeto son las siguientes:

- **AbsolutePosition**: Nos devuelve el número de registro en el que nos encontramos, es decir, a qué registro apunta el puntero del objeto RecordSet. También lo podemos utilizar para fijar la posición en la que nos encontremos, es decir, si decimos que Ob_RS.AbsolutePOSITION=5, estaremos moviendo el puntero del objeto RecordSet al quinto registro.

- **Bof**: Esta propiedad es cierta cuando nos encontramos al principio del RecordSet, es decir, si el puntero señala antes del primer registro. Cuando estamos en el primer registro, Bof es igual a False.

- **Eof**: Esta propiedad es cierta cuando nos encontramos al final del RecordSet, es decir, cuando el primero esté después del último registro. Si estamos leyendo el último esta propiedad nos dará False.

- **RecordCount**: Indica el número de registro que hay dentro del objeto RecordSet.

Aunque estas propiedades y métodos son fáciles de comprender, se muestra el siguiente ejemplo para aportar una mayor claridad y eliminar cualquier tipo de duda con respecto a la función de cada una de estas facilidades.

METODOS, PROPIEDADES Y COLECCIONES DEL OBJETO RECORDSET

Para terminar de ver todas las posibilidades que nos ofrece el objeto RecordSet, vamos a enumerar todos sus métodos, propiedades y colecciones, de modo que el lector pueda consultar este capítulo en caso de tener alguna duda. El objeto RecordSet tiene los siguientes métodos.

- **AddNew** [campos], [valores]: Crea un nuevo registro en la tabla a la que representa el objeto RecordSet. Este nuevo registro no se hará efectivo hasta que no apliquemos en método Update al objeto RecordSet.

- **CancelBatch** [Registros_afectados]: Cancela una actualización en modo batch. En Registros_afectados se aplican los registros que han sido borrados. Se pueden utilizar las siguientes constantes:
    - adAffectCurrent: Sólo afecta al registro actual..
    - adAffectGroup: Sólo afecta a los registros que hayan superado el filtro.
    - AdAffectAll: Afecta a todos los registros. Éste es el valor por defecto.

- **CancelUpdate**: cancela una actualización. Utilizaremos este método cuando no deseemos que se lleve a cabo una actualización que estamos realizando, ya sea insertar o modificar un registro.

- **Clone**: crea una copia del objeto RecordSet. Tendremos que usar la sentencia Set para asignar esta copia a una variable, ya que estamos tratando con un objeto.

- **Close**: cierra el objeto RecordSet al que se le aplique. Este método será útil cuando modifiquemos dicho objeto, ya que puede darse el caso de que se pierdan las modificaciones. Es recomendable utilizarlo siempre para añadir claridad al código y poder diferenciar de dónde a dónde utilizamos el objeto.

- **Delete** [Registros_afectados]: Elimina el registro del objeto RecordSet apuntado por el puntero de dicho objeto, es decir, el registro que estemos tratando en ese momento. En Registros_afectados se especifican los registros que han sido borrados. Se pueden utilizar las siguientes constantes:

    - adAffectectCurrent: Sólo borra el registro actual. Ésta es la opción por defecto.
    - AdAffectGroup: Borra todas las que hayan superado el filtro.

- **GetRows** ([filas], [inicio], [campos]): Devuelve un array con el contenido del objeto RecordSet. Lo podremos asignar a una variable para utilizarlo como si se tratara de una tabla normal. El primer índice representará la columna y el segundo la fila. Podemos añadir tres parámetros a este método. El primero será el número de filas que queremos que devuelva. El segundo, a partir de qué fila queremos que empiece. Y por último los campos que queremos que muestre.

- **Move** Num_Registros, [inicio]: Mueve el puntero del RecordSet un determinado número de registros. Podemos poner números positivos o negativos, indicando con un número positivo que avanzamos en la tabla y con uno negativo, que retrocedemos. Podemos indicar el registro de inicio para que empiece a contar desde él.

- **MoveFirst**: Nos lleva al primer registro de la tabla

- **MoveLast**: Nos lleva al último registro de la tabla

- **MoveNext**: Avanza un registro hacia debajo en la tabla

- **MovePrevious**: Avanza un registro hacia arriba en la tabla

- **NextRecordSet** ([registros_afectados]): Cuando una consulta Sql devuelve más de un RecordSet utilizaremos este método para borrar el RecordSet actual y pasar al siguiente.

- **Open** [origen], [ActiveConnection], [cursor], [cerrojo],[opciones]: Abre un objeto RecordSet uniendo éste con una tabla o con varias. Le podemos pasar como parámetro el origen de datos, que será una sentencia Sql, un procedimiento almacenado o el nombre de una tabla. El objeto Connection será quien relacione el RecordSet con una base de datos, el cursor y el cerrojo cuyas constantes están en el apartado anterior. Además podemos incluir una opción en la que especificamos el tipo de origen. Las constantes de los cursores y las constantes de los cerrojos se han detallado en la sección *"Modalidades del objeto RecordSet"*.

Por último las constantes para el parámetro Option son las siguientes:

  - adCmlText: Es una definición textual de una sentencia SQL.
  - AdCmlTable: Es el nombre de una tabla
  - AdCmlStoredProc:Representa a un procedimiento almacenado
  - AdCmlUnknown: No se puede decir qué es. Ésta es la opción por defecto

- **Requery**: Es un método de refresco. Vuelve a ejecutar el comando que generó el RecordSet para actualizar los posibles cambios que se hayan producido en la tabla a la que hace referencia.

- **Resync**: [registros_afectados]: Este método también sirve de refresco del RecordSet, pero en este caso lo que hace es simplemente traer los registros nuevos que tenga la tabla. Podemos indicar los campos que queremos que sean afectados con el parámetro registros_afectados. Podemos utilizar una de las siguientes constantes:

  - adAffectCurrent: Sólo afecta el registro actual
  - adAffectGroup: Sólo afecta a los registros que hayan superado el filtro
  - adAffecctAll: Afecta a todos los registros. Éste es el valor por defecto

- **Supports** (método): devuelve True o False dependiendo de si el RecordSet soporta o no el método de función que pasamos como parámetro. Es decir, dependiendo del cerrojo que hayamos utilizado, el objeto RecordSet soportará unos métodos u otros. Con este método podremos saber si soporta un método en concreto. Tenemos las siguientes posibilidades:

  - adAddNew: El RecordSet soporta el método AddNew.

  - adApproxPoxition: El RecordSet soporta las propiedades Absolute Position y AbsolutePage.

- adBookmark: El RecordSet soporta la propiedad Bookmark.

- adDelete: El RecordSet soporta el método Delete.

- adHoldRecords: El objeto RecordSeet puede recuperar un valor modificado sin anular toda la modificación.

- adMovePrevious: El RecordSet soporta los métodos MovePrevious y Move.

- adResync: El RecordSet soporta el método Resync

- adUpdate: El RecordSet soporta el método Update.

- adUpdateBatch: El RecordSet soporta el método UpdateBatch.

- **Update** [campos], [valores]: Almacena los cambios que hemos realizado en un RecordSet. Sin este método los cambios o inserciones de registros que realizáramos no tendrían efecto en la base de datos física. Podemos además especificar un array de campos con sus respectivos valores.

- **UpdateBatch** [registros_afectadis]: Almacena los cambios que hemos realizado en un objeto RecordSet mediante un proceso de tipo batch.

    - adAffectCurrent: Sólo afecta al registro actual
    - adAffectGroup: Sólo afecta a los registros que hayan superado el filtro
    - adAffecAll: Afecta a todos los registros. Éste es el valor por defecto

Vistos los métodos del objeto RecordSet, vamos a enumerar sus propiedades. Éstas nos ayudarán a saber el estado del objeto en cada momento para poder controlar la ejecución de nuestro código. Las propiedades del objeto RecordSet son las siguientes:

- **AbsolutePage**: Nos devuelve o fija la página en la que nos encontramos actualmente. Utilizaremos esta propiedad para situarnos en una página determinada cuando estemos paginando, o bien para saber dónde nos encontramos. Podemos utilizar las siguientes constantes:

    - adPosUnknown: El RecordSet está vacío y la página es desconocida o soporta la propiedad AbsolutePage
    - adPosBOF: La propiedad Bof es True
    - adPosEOP: La propiedad Eof es True

- **AbsolutePosition**: Nos devuelve o fija el registro en el que nos encontramos. Este registro vendrá especificado por su número de orden. Podemos utilizar las constantes:

- adPosUnknown: El registro está vacío y la página es desconocida o no soporta la propiedad AbsolutePage
- adPosBOF: La propiedad Bof es True
- adPosEOF: La propiedad Eof es True

- **ActiveCeonnetion**: Nos permite o fija los parámetros del objeto Connection al que hace referencia el objeto RecordSet. Devuelve el Dsn, la ruta de la base de datos, el servidor de la base de datos, etc. Si éste ya está abierto sólo debemos utilizar para leer el nombre, ya que si fijáramos otra conexión perderíamos la actual

- **Bof**: Devuelve True ccuando nos encontramos al principio de una tabla. Esto no significa cccuando estamos en el primer registro, sino antes del mismo. Nos será útil cuando recorramos una tabla en sentido inverso, ya que sabremos cuándo detenernos.

- **Bookmark**: Devuelve o fija un identificador para el registro actual.

- **CacheSize**: Fija o devuelve el número de registros de un RecordSet que están almacnados en la memoria caché.

- **CursorLocation**: Especifica la librería de cursores que vamos a utilizar. Podemos utilizar los valores.

  - adUseClient: librería de cliente.
  - adUseServer: librería de servidor.

- **CursorType**: Devuelve o fija el cursor que vamos a utilizar en el objeto RecordSet. Puede utilizar cualquiera de los vistos en apartados anteriores de este capítulo. Si queremos especificar un cursor debemos hacerlo antes de abrir el objeto. Las constantes a utilizar son las siguientes:

  - adOpenForwardOnly: Este cursor sólo nos permite recorrer el objeto de forma secuencial. Con este cursor no podremos filtrar la tabla. Este es el que se pone por defecto.

  - AdOpenKeySet: Con este cursor nos podemos mover para adelante y para atrás en el objeto RecordSet. Además si en algún momento otro usuario modifica o borra algún registro de la tabla. Se reflejará en el objeto RecordSet. No así si añade un nuevo registro.

  - AdOpenDinamic: Nos podemos mover en cualquier dirección y además veremos cualquier modificación, borrado o inserción de registros que haya realizado otro usuario.

  - AdOpenStatc: Nos permite mover en todas direcciones, pero no se verá ningún cambio que haya realizado otro usuario sobre la tabla

-

- **EditMode**: indica cuál es estado de edición del registro actual. Al decir edición nos referimos a cuando lo estamos modificando. Tenemos tres posibilidades:

    - adEditNone: No estamos editando el registro.

    - adEditInProgress: Hemos modificado el registro pero no lo hemos salvado.

    - adEditAdd: Hemos llamado al método AddNew

- **Eof**: Devuelve el valor True cuando hemos llegado al final del objeto RecordSet, este es, después del último registro.

- **Filter**: Filtra un RecordSet dejando visibles en el mismo, solo los campos que superen el filtro. Podemos filtrar el objeto RecordSet con una de las siguientes constantes:

    - adOpenForwardOnly: Este cursor sólo nos permite recorrer el objeto de forma secuencial. Con. Con este cursor no podemos filtrar la tabla. Éste es el que se pone por defecto.

    - adOpenKeySet: Con este cursor nos podemos mover para adelante y para atrás en el objeto RecordSet. Además si en algún momento otro usuario modifica o borra algún registro de la tabla, se reflejará en el objeto RecordSet. No así si añade un nuevo registro.

    - adOpenDinamic: Nos podemos mover en cualquier dirección y además veremos cualquier modificación, borrado o inserción de registros que haya realizado otro usuario.

    - adOpenStatic: Nos podemos mover en todas direcciones, pero no se verá ningún cambio que haya realizado otro usuario sobre la tabla.

- **LockType**: Devuelve o fija el cerrojo que se va a utilizar en el objeto RecordSet. Si lo queremos fijar lo demos hacer antes de abrir el objeto. Los cerrojos que podemos utilizar son los siguientes:
    - adLockReadOnly: Indica que el usuario no podrá modificar los registros de la tabla. Una vez que abrimos la tabla, ésta no podrá ser modificada. Éste es el que aparece por defecto.

    - adLockPessimistic: Indica que cuando se abra una base de datos nadie más podrá hacerlo al mismo tiempo. Cuando se cierre se abrirá el cerrojo a los demás usuarios. De este modo nos aseguramos la integridad de los datos.

    - adLockOptimistic: Indica que la base de datos será cerrada a los demás usuarios cuando se esté realizando la operación Update del objeto

RecordSet. De este modo la base de datos quedará bloqueada mucho menos tiempo. Si me aseguro de que todas las páginas Web tienen esta restricción, entonces conseguiré el mismo efecto que con el cerrojo anterior.

- adLockBatchOptimistic: Indica que el registro se actualizará en modo batch.

- **MarshalOption**: Especifica los registros que se deben actualizar en el servidor. Puede utilizar las siguientes constantes:

  - adMarshalAll: Actualiza todos los registros. Es la opción por defecto.

  - adMarshalModifiedOnly: Sólo actualiza los registros que hayan sido modificados.

- **MaxRecords**: Fija o devuelve el número de registros máximo que va a tener el objeto RecordSet. Si ponemos a esta propiedad el valor 0, el cual está por defecto, no tendrá ningún tope.

- **PageCount**: Devuelve el número de páginas que tiene un objeto RecordSet. Este número dependerá del tamaño de página, es decir, del número de registros que pueda tener una página.

- **PageSize**: Fija o devuelve el número de registros que tiene una página de un objeto RecordSet. Por defecto son diez.

- **RecordCount**: Devuelve el número de registros que tiene un objeto RecordSet. Si devuelve –1 significa que dicho número no se puede determinar.

- **Source**: Indica al objeto RecordSet las características de la tabla que va a representar. Se podrá especificar con una tabla, una sentencia Sql o un procedimiento almacenado.

- **State**: devuelve el estado del objeto RecordSet que puede estar abierto o cerrado. Para estos estados utiliza las siguientes constantes:

  - adStateClosed: el objeto está cerrado.
  - adStateOpen: el objeto está abierto.

- **Status**: devuelve el estado del objeto RecordSet cuando se está actualizando en modo batch. Puede utilizar las siguientes constantes:

  - adRecOK: El registro se ha actualizado correctamente.

  - adRecNew: El registro es nuevo.

  - adRecModified: El registro ha sido modificado.

- adRecDelete: El registro ha sido borrado.
- adRecInvalid: El registro no se ha salvado por no ser válido.
- adRecMultipleChanges: El registro no se ha salvado porque afectaría a más de un registro.
- adRecPendingChanges: El registro no se ha salvado porque hace referencia a una inserción pendiente.
- adRecCanceled: El registro no se ha salvado porque no se ha anulado la operación.
- adRecCantRelase: El registro no se ha salvado porque existen cerrojos que lo impiden.
- adRecConcurrencyViolation: El registro no se ha salvado porque se está produciendo concurrencia en modo Optimistic.
- adRecIntegrityViolation: El registro no se ha salvado para no violar la integridad de los datos.
- adRecMaxChangesExceeded: El registro no se ha salvado porque hay demasiados registros sin actualizar.
- adRecObjectOpen: El registro no se ha salvado porque se ha producido un conflicto con un objeto de almacenamiento abierto.
- adRecOutOfMemory: El registro no se ha salvado porque se ha producido un desbordamiento de pila.
- adRecPermissionDenied: El registro no se ha salvado porque no tiene permisos para ello.
- adRecSchemaViolation: El registro no se ha salvado porque violaría la estructura interna de la base de datos.
- adRecDBDeleted: El registro ya había sido borrado anteriormente.

Por último, veamos las dos colecciones que tiene este objeto, Fields y Properties, ambas formadas por objetos. Sólo vamos a ver su cometido, ya que todo lo referente a los objetos que contiene se explicará más adelante:

- **Fields**: Representa el conjunto de campos de la tabla. Es decir, cuando tengamos un registro podremos acceder a un campo del mismo a través de esta colección. Lo haremos de la siguiente forma:

  Ob_RS.Fields(nombre_o_número_de_campo).

- **Properties**: Contiene las propiedades del objeto RecordSet. Estas dependerán de los valores por defecto que ofrece cada proveedor y de las modificaciones que nosotros realicemos. Accederemos a ellas de la siguiente forma:Ob_RS.Properties (propiedad)

TRANSACCIONES, OPERACIONES SOBRE VARIAS TABLAS

Hasta el momento hemos trabajado con una sola base de datos y realizado una única operación. Pero esto normalmente no es así. Lo más común es que al actualizar una tabla o base de datos tengamos que actualizar varias que están relacionadas con ella. Esta situación nos crea el problema de que si una de las actualizaciones falla o no puede llevarse a cabo, debemos cancelar el resto. A este tipo de operaciones es a las que llamaremos transacciones.

El objeto Connection nos permite realizar este tipo de operaciones a través de una serie de métodos. Estos métodos, que a continuación veremos, nos permitirán abrir o definir una transacción, para una vez definida ejecutarla o anularla según sea conveniente.

En primer lugar nos encontramos con el método BeginTrans que indica al objeto Connection que vamos a comenzar a definir una transacción. Una vez que ejecutemos ese método, todas las operaciones que realicemos sobre bases de datos no tendrán un efecto real sobre las mismas.

Para que dichas operaciones se hagan efectivas en la base de datos, utilizaremos el método Commit.Trans. Este método será el que realice las operaciones que hemos definido sobre la base de datos.

Pero, ¿qué pasa si en un momento de la transacción nos damos cuenta de que lo que estamos haciendo no es válido por cualquier motivo? En teoría, todas las modificaciones que hasta ahora hayamos realizado habrán quedado ya actualizadas. Esto no sucede en las transacciones, gracias al método RollBackTrans. Este método anula todas las modificaciones que hayamos realizado dentro de la transacción, es decir, desde que utilizamos el método BeginTrans. Este método, como es lógico, habrá que invocarlo antes de cerrar la transacción con CommitTraans, ya que si no los cambios serán irreversibles.

Un ejemplo sencillo será el siguiente. Tenemos una base de datos con tres tablas que contienen distintos datos de una persona. Queremos introducir un nuevo registro en las tres tablas siempre y cuando el nuevo dato no esté repetido en ninguna de las tablas. La forma más sencilla de hacerlo sería crear una transacción en la que introdujésemos el nuevo dato, de una en una, en todas las tablas. Si en alguna de ellas está repetido anulamos la transacción para que no se inserte en el resto, y si no aceptamos la transacción insertado el nuevo registro en todas las tablas.

METODOS, PROPIEDADES Y COLECCIONES DEL OBJETO CONNECTION

Al igual que hemos hecho con el objeto RecordSet vamos ahora a enumerar los métodos, propiedades y colecciones del objeto Connection. Muchos de éstos ya se han visto anteriormente, pero aun así se van a mostrar para dar una visión más completa de las prestaciones de este objeto. Empezaremos con los métodos:

- **BeginTrans**. Abre una transacción. Todas las operaciones que realicemos a partir de ese momento no se harán efectiva hasta que no cerremos la transacción
- **Close**: Cierra el objeto Connection
- **CommitTrans**: Cierra una transacción, haciendo efectivos todos los cambios que se han realizado dentro ella.

- **Execute** Comando, [registros_afectados], [opciones] o **Execute** (Comando, [registros_afectados], [opciones]): Ejecuta una sentencia Sql. Devuelve el resultado en forma de un objeto RecordSet. Tendremos que utilizar la sentencia Set para asignar el resultado de esta operación a una variable. Si la instrucción no devuelve nada, los parámetros irán sin paréntesis y en caso contrario con paréntesis. Podemos saber el número de registros que han sido afectados con registros_afectados, así como especificar el tipo de sentencia Sql con las siguientes constantes:

    - adCmdText: Es una definición textual de una sentencia SQL.
    - AdCmdTable: Es el nombre de una tabla
    - AdCmdStoredProx: Representa a un procedimiento almacenado
    - AdCmdUnknown: No se puede decir qué es. Esta es la opción por defecto

- **Open** [referenciaADO], [usuario], [clave]: Abre un objeto Connection. Esto quiere decir que crea un enlace entre dicho objeto y una base de datos física. Como argumento le pasaremos la referencia a la base de datos, que puede ser un Dsn u otra frase que determine el camino a la base de datos. Por elemplo: driver=(MixrosoftAccessDriver(*.mdb), donde indicamos que el proveedor de datos va a ser Access. Además podemos indicar con uid el usuario y con pwd la clave. También podemos pasar el nombre y password necesarios para acceder a la base de datos como parámetros del método Open. Además le podemos pasar los siguientes argumentos.

- **Provider**: Nombre del proveedor.

    - Data Source: nombre de la fuente de datos.
    - User: usuario.
    - Password: clave.
    - File Name: nombre del archivo con la información de la conexión.
    - Dbq: ruta de la base de datos.

Por ejemplo, si tenemos una base de datos creada con Acces en el directorio bases, lo indicaremos de la siguiente forma:

“driver = { Microsoft Acces Driver (*.mdb) } ; dbq= c:/bases/nombre _base.mdb”

- **OpenSchema** (typo_de cuestion), [criterio]) : Devuelve informacion sobre la base de datos que estamos utilizando
- **RollbackTrans**:Anula una transacción . Todos los cambios que se hayan realizado dentro de ella se pierden

Una vez vistos los métodos de ese objeto, vamos a revisar las propiedades del mismo. Las propiedades nos servirán para saber y fijar ciertos valores utilizados por el objeto. El objeto Connection tiene las siguientes propiedades:

- **Attributes**: Fija o devuelve los atributos de una transacción. Puede tener dos valores:

    - AdXactCommitRetaining: indica que al llamar al método CommitTrns se creará automáticamente una nueva transacción.

    - AdXactAbortRetaining: indica que al llamar al método RollbackTrans se empezará automáticamente una nueva transacción.

- **CommandTimeOut**: Especifica el tiempo en segundos que un objeto dependa de un Connection (Command y RecordSeet) va a esperar cuando realiza una solicitud. Por defecto son treinta segundos, es decir, a los treinta segundos deja de intentarlo y devuelve un error. Si le asignamos el valor 0 el tiempo espera será infinito, con lo que si el proveedor no responde, el explorador se quedará bloqueado.

- **ConnectionString**: Representa una referencia a una base de datos. Esta puede ser un Dsn o una referencia directa a la base de datos. Puede tener por ejemplo esta sintaxis: dsn = ruta_de_la_base,uid = nombre_usuario,pwd,clave. Además podemos utilizar los siguientes atributos:

    - Provider: Donde indicaremos el nombre del proveedor de la base de datos (en nuestros ejemplos Access).

    - Data Source: Donde indicaremos el nombre de la fuente de datos.

    - User: Donde indicamos el nombre del usuario.

    - Password: Donde indicaremos la clave de acceso a los datos.

    - File Name: Donde especificaremos el nombre del archivo de documentación del proveedor.

- **ConnectionTimeOut**: Indica el tiempo en segundos que va a esperar un objeto Connection la respuesta de un proveedor de datos al ejecutar el método Open. Por defecto es de 15 segundos. Podemos hacer indefinido este tiempo poniendo el valor 0.

- **CursorLocation**: Especifica la librería de cursores que se está utilizando. Se pueden utilizar dos valores:

    - AdUseClient: Para usar la librería de cliente.

    - AdUseServer: Indica que se usa la librería del servidor. Es el valor por defecto.
- **DefaulDatabase**: Indica la base de datos que se está utilizando.

- **IsolationLevel**: Indica el nivel de aislamiento de la base de datos. Los cambios de esta propiedad no se hacen efectivos hasta que o se llega a un BeginTrans.

- **Mode**: Contiene los permisos de la base de datos a la que hace referencia el objeto Connection. Puede tener los siguientes valores:

    - adModeUnknown: Valor indeterminado, se utiliza por defecto.

    - adModeRead: Solo lectura.

    - adModeWrite: Solo escritura.

    - adModeReadWrite: Lectura y escritura.

    - adModdeShareDenyRead: No permite que otros abran la base de datos con permiso de lectura.

    - adModeShareDenyWrite: Impide que otros abran la base con permiso de escritura.

    - adModeShareExclusive: Evita que otros abran la base de datos con permiso de lectura y/o escritura.

    - adModeShareDenyNone: No permite que otros abran la base de datos.

- **Provider**: Especifica el nombre del proveedor de la conexión.

- **State**: Devuelve el estado del objeto. Puede tener dos valores:

    - adStateClose: objeto cerrado.
    - adStateOpen: objeto abierto.

- **Versión**: Devuelve la versión del ADO.

Por último vamos a ver las dos colecciones que tiene el objeto Connection: Errores y Properties. Ambas colecciones están compuestas por objetos Error y Property, de modo que no se van a

explicar los métodos de los mismos, sino que remitimos al lector a los siguienes apartados de este capítulo donde se detallarán dichos objetos.

- **Errors**: Contiene los errores que se producen en la ejecución de algún método del objeto Connection. Accederemos a ellos de la siguiente forma: Ob_Conn.Errors (índice).

- **Properties**: Contiene las propiedades del objeto Connection. Vendrán en función de los parámetros por defecto de cada proveedor y de las modificaciones que nosotros hayamos realizado. Accederemos a ellas de la siguiente forma: Ob_Conn.Properties (propiedad).

EL OBJETO FIELD, METODOS, PROPIEDADES Y COLECCIONES

El objeto Field es un objeto secundario que representa una columna en un objeto RecordSet. Al decir secundario nos referimos a que no puede existir por sí mismo, no podemos instanciarlo directamente. El objeto Field depende del objeto RecordSet. Cuando creamos un objeto RecordSet y lo asignamos a una tabla estamos creando también objetos del tipo Field que corresponderán a cada uno de los campos de esa tabla. Sus métodos son los siguientes:

- **AppendChunk** Datos: Sirve para añadir grandes cantidades de texto a un campo. Ese texto irá como parámetro del método. La primera vez que lo utilicemos, el texto se insertará en el campo borrando su anterior contenido. Si lo volvemos a utilizar en el mismo campo, el texto se añadirá al anterior. Para utilizar este método el atributo adFieldLong debe tener el valor True.

- **GetChunk** (Número): Devuelve un número de caracteres de un campo. Lo utilizaremos para grandes cantidades de texto. La primera vez que lo hacemos empieza por el principio. Si lo volvemos a utilizar en el mismo campo, sigue desde donde lo dejó.

Una vez visto los métodos vamos a pasar a las propiedades.

- **ActualSize**: Devuelve el tamaño de un campo. Si no lo puede averiguar devuelve –1 que coincide con la constante adUnkown.

- **Atributes**: Colección de objetos. Atributos que contiene contantes que nos indican el estado de un campo u objeto Field. Puede tener los siguientes valores:

    - adFDMayDefer: El valor del campo no es devuelto con el resto del registro. Hay que acceder directamente a él.

    - adFldUndatable: Se puede escribir en el campo.

    - adFldUnknownUpdatable: No se sabe el tipo de datos que se debe escribir en el campo.

- adFldFixed: El campo contiene datos de tamaño fijo.
- adFldlsNullable: Se puede escribir Null en el campo.
- adFldMayBeNull: Se puede llegar a leer el valor Null en el campo.
- adFldLong: Se pueden usar los métodos AppendChunk y GetChunk.
- adFldRowlD: El campo es el identificador de cada registro.
- adFldRowVersion: El campo representa una columna temporal.
- adFldCacheDererred: El dato del campo está almacenado en caché.

- **DefinedSize**: Devuelve el tamaño definido para el campo.
- **Name**: Devuelve el nombre del campo.
- **NumericScale**: Indica la cantidad de dígitos a la izquierda de la coma que puede representar un campo numérico.
- **OriginalValue**: Devuelve el valor que tenía el campo antes de ser actualizado. Si no ha sido modificado devolverá el valor actual del campo.
- **Precision**: Devuelve el número de dígitos que un campo numérico puede representar.
- **Type**: Devuelve el tipo de datos del campo. Puede devolver los siguientes constantes/valores:
    - adBigInt (20): entero de 8 bytes
    - adBinary (128): binario
    - adBoolean (11): booleano
    - adBSTR (8): Null
    - adChar (129): texto
    - adCurrency (6): Currency
    - adDate (7): Fecha
    - adDBDate (133): fecha en formato aaaammdd
    - adBDTime (134): hora en formato hhmmss
    - addDBTimeStamp (135): fecha y hora en formato yyymmddhmmss
    - adDecimal (14): decimal

- adDouble (5): entero de precisión doble
- adEmpty (0): Empty
- adError (10): código de error de 32 hits
- adGUID (72): identificador de grupo
- adInteger (3): entero de 4 bytes
- adIUnknown (13): puntero a una interfaz desconocida
- adNumeric (131): numérico de precisión exacta
- adSingle (4): entero de precisión simple
- adSmallInt (2): entero corto de dos bytes
- adTinyInt (16): entero de un byte
- adUnsignedBigInt (2): entero sin signo de 8 bytes
- adUnsignedInt (19): entero sin signo de 4 bytes
- adUnsignedSallInt (18): entero sin signo de 2 bytes
- adUnsignedTinyInt (17): entero sin signo de 1 byte
- adUserDefined (132): variable definida por el usuario
- adVariant (12): variable de un objeto OLE
- adWChar (130): (Null)

- **UnderlyngValue**: Devuelve el valor actual del campo en la base de datos.
- **Value**: Devuelve el valor del campo.

Por último vamos a ver la única colección que tiene:

- **Properties**: Esta colección contiene las propiedades del objeto Fields. Estas tendrán distintos valores en función del proveedor y de los cambios que realicemos posteriormente.

EL OBJETO PARAMETER, METODOS, PROPIEDADES Y COLECCIONES

Este objeto representa un parámetro utilizado en una sentencia Sql o en un procedimiento almacenado. También se utilizara con los valores que devuelva un procedimiento almacenado. Podremos instanciarlo directamente utilizando el método CreateObject del objeto Server. Le pasaremos el parámetro "ADOB.Parameter" .

Este objeto sólo tiene un método que es **AppendChunk**.

- **AppendChunk** Datos: Lo utilizaremos para añadir una gran cantidad de datos, ya sean de texto o en forma binaria. Para poder utilizar este método el atributo adParamLong de la propiedad Atibutes debe ser True. La primera vez que se llama a este método los datos se sobreescriben en el parámetro. El resto de veces que se llame a este método sobre el mismo parámetro los parámetros se añaden al final del parámetro.

  Propiedades:

- **Attributes**: Colección que indica mediante constantes si el parámetro soporta ciertas propiedades. Estas pueden ser las siguientes:

  - adParamSigned: acepta valores con signo
  - adParamNullable: acepta el valor Null
  - adParamLong: acepta el metodo AppendChunk

- **Direction**: Determina la dirección hacia la que circulan los datos por ese parámetro. Las constantes/valor que determina esta dirección son las siguientes:

  - adParamInput (1): parámetro de entrada
  - adParamInput (2): parámetro de salida
  - adParamInputOut (3): parámetro de entrada y salida
  - adParamReturnValue (4): parámetro devuelto por un procedimiento

- **Name**: Nombre del objeto Parameter.

- **NumericScale**: Indica la cantidad de dígitos a la izquierda de la coma que puede representar un parámetro numérico.

- **Precision**: Devuelve el numero de dígitos que un parámetro numérico puede representar.

- **Size**: Tamaño máximo que puede almacenar.

- **Type**: Devuelve el tipo de datos del parámetro. Puede devolver los siguientes constantes valores:

- adBigInt (20): entero de 8 bytes
- adBinary (128): binario.
- adBooleam (11): booleano
- adBSTR (8): Null
- adChar (129): texto
- adCurrency (6): Currency
- adDate (7): fecha
- adDBDate (133): fecha en formato aaaamndd
- adDBDTime (134): Hora en formato hhmmss
- adDBTimeStamp (135): fecha y hora en formato yyyymmddhhmmss
- adDecimal (14): decimal
- adDouble (5): entero de precisión doble
- adEmpty (0): Empty
- adError (10): código de errores de 32 bits
- adGUID (72): identificador de grupo
- asIDspatch (9): puntero a una interfaz Idispatch en un objeto OLE
- adInteger (3): entero de 4 bytes
- adIUnknow (13): puntero a una interfaz desconocida
- adLongVarBinary (208): valor binario largo
- adLongVarChar (201): valor de texto largo
- adLongVarWChar (203): texto terminado en Null
- adNumeric (131): numerico de precision exacta
- adSingle (4): entero de precision simple
- adSmallInt (2): entero corto de dos bytes
- adTinyInt (16): entero de un byte
- adUnsignedBigInt (21): entero sin signo de 8 bytes
- adUnsignedInt (19): entero sin signo de 4 bytes
- adUnsignedSmallInt (18): entero sin signo de 2 bytes
- adUnsignedTinyInt (17): entero sin signo de 1 byte
- adUserDefined (132): variable definida por el usuario
- adVarBinary (204): valor binario

- adVarChar (200): texto
- adVariant (12): Variable de un objeto OLE
- adVarWChar (202): Null
- adWChar (130): Null

- **Value**: Devuelve el valor del parámetro.

Por ultimo veamos la única colección del objeto Parameter, la colección Properties.

- **Properties**: esta formada por objetos del tipo Property. Esta colección contiene características del objeto Parameter que dependerán del proveedor de datos y de las modificaciones que hayamos realizado nosotros.

El objeto Property: Propiedades

Este objeto contiene las propiedades que determinan el funcionamiento por defecto del resto de los objetos. Estas propiedades dependerán del proveedor de datos, ya que cada uno fija su modo de actuar.

Este objeto sólo tiene propiedades y son las siguientes:

- **Atributes**: Representa atributos sobre la propiedad a la que representa el objeto. Puede tener los siguientes valores:
  - AdPropNotSupported: el proveedor no soporta esta propiedad.
  - AdPropRequired: la fuente de datos necesita esta propiedad antes de ser inicializada.
  - AdPropOptional: la fuente de datos no necesita esta propiedad antes de ser inicializada.
  - AdProRead: esta propiedad puede ser leída por el usuario.
  - AdPropWrite: esta propiedad puede ser modificada por el usuario.

- **Name**: Nombre de la propiedad.

- **Type**: Devuelve el tipo de datos de la propiedad. Puede devolver los siguientes constantes/ valores:
  - adBigInt (20): entero de 8 bytes.

- adBinary (128):binario
- adBoolean(11):booleano
- adBSTR (8):Null
- adChar (129): texto
- adCurrency (6): Currency
- adDate (7): fecha
- adDBDate (133): fecha en formato aaaammdd
- adBDTime (134): hora en formato hhmmss
- adDBTimeStamp (135): fecha y hora en formato yyyymmddhhmmss
- adDecimal (14): decimal
- adDouble (5): entero de precisión doble
- adEmpty (0): Empty
- adError (10):código de error de 32 bits
- adGUID (72): identificador de grupo
- adInteger (3): entero de 4 bytes
- adINumeric (131): numérico de precisión exacta
- adSingle (4): entero de precisión simple
- adSmallInt (2): entero corto de 2 bytes
- adTinyInt (16): entero de un byte
- adUnsignedBigInt (21): entero sin signo de 8 bytes
- adUnsignedInt(19): entero sin signo de 4 bytes
- adUnsignedSmallInt (18): entero sin signo de 2 bytes
- adUnsignedTinyInt (17): entero sin signo de 1 byte
- adUserDefined (132): variable definida por el usuario
- adVariant (12):variable de un objeto OLE
- adWChar (130): Null

- **Value**: valor del objeto Property

EL OBJETO ERROR

Este objeto contiene los errores que se producen durante la ejecución de paginas con ADO. Sólo tiene propiedades y son las siguientes:

- **Description**: descripción del error.
- **HelpContext**: tema del archivo de ayuda para el error
- **HelpFile**: Archivo de ayuda para el error
- **NativeError**: muestra el error desde el proveedor
- **Number**: Número de error
- **Source**: objeto que ha provocado el error
- **SQLState**: código de error de cinco caracteres generado por SQL

I. INTRODUCCIÓN

El Web es un medio para localizar/enviar/recibir información de diversos tipos, aun con las bases de datos. En el ámbito competitivo, es esencial ver las ventajas que esta vía electrónica proporciona para presentar la información, reduciendo costos, el almacenamiento de la información, y aumentando la rapidez de difusión de la misma.

Internet provee de un formato de presentación dinámico para ofrecer campañas y mejorar negocios, además de que permite acceder a cada sitio alrededor del mundo, con lo cual se incrementa el número de personas a las cuales llega la información.

Hasta el año 2000 alrededor de 14 millones de personas alrededor del mundo hacen uso de Internet, lo cual demuestra el enorme potencial que esta red ha alcanzado, con lo cual se puede decir que en un futuro no muy lejano, será el principal medio de comunicación utilizado para distintos fines.

Pero, no sólo es una vía para hacer negocios, sino también una gran fuente de información, siendo éste uno de los principales propósitos con que fue creada.

Una gran porción de dicha información requiere de un manejo especial, y puede ser provista por bases de datos.

En el pasado, las bases de datos sólo podían utilizarse al interior de las instituciones o en redes locales, pero actualmente el Web permite acceder a bases de datos desde cualquier parte del mundo. Estas ofrecen, a través de la red, un manejo dinámico y una gran flexibilidad de los datos, como ventajas que no podrían obtenerse a través de otro medio informativo.

Con estos propósitos, los usuarios de Internet o intranets pueden obtener un medio que puede adecuarse a sus necesidades de información, con un costo, inversión de tiempo, y recursos mínimos. Asimismo, las bases de datos serían usadas para permitir el acceso y manejo de la variada información que se encuentra a lo largo de la red.

II. OBJETIVOS

GENERAL:

- Realizar un estudio de dos diferentes tecnologías de Integración de Bases de Datos en el Web.

ESPECÍFICO:

- Conectar una base de datos diseñada en Microsoft Access a un sitio web utilizando **Active Server Pages** (ASP) y scripts **CGI**, desarrollados en lenguaje **Perl**.

- Valorar las diferencias entre las dos tecnologías implementadas en la conexión de la base de datos al sitio web.

III. JUSTIFICACIÓN

El tema de este trabajo lo he seleccionado debido a que las tendencias de las nuevas tecnologías de publicación de información sensible en el web, está actualmente empleando métodos que integran bases de datos, mediante el uso y explotación de los servicios de Internet que le dan más potencia a dichos sitios, dinamismo e interactividad con el usuario, y por lo que considero tema de interés y estudio tanto para los docentes del Departamento de Computación de la UNAN – León y estudiantes. Además deja la posibilidad de desarrollar un sistema de información con todos las especificaciones de requisitos de usuario que sea accesible desde un sitio web, y a nuevas implementaciones que surjan a su uso.

IV. MARCO TEORICO

Entre los objetivos de esto proyecto, como se destaca anteriormente, es realizar una integración de bases de datos mediante tecnologías correspondientes a través del web, para poder luego presentar las diferencias y similitudes que puedan existir entre ellas, para lo que se necesita tener conocimientos previos de ciertos términos y abreviaturas utilizados.

Definiciones:

- **Servidor**: Ordenador con ciertas características predominantes que mediante un sistema operativo de red permite administrar y controlar ciertos ordenadores pequeños llamados clientes, con el objetivo de que éstos compartan recursos del servidor y puedan al mismo tiempo comunicarse entre sí para algunos propósitos.

- **Cliente:** Ordenadores pequeños, que mediante un programa y algunos periféricos son capaces de acceder al servidor para algunos propósitos (en algunos casos se debe entender como cliente a un Navegador de Internet, ya que es un programa que se ejecuta en el cliente haciendo una conexión a un servidor web).

- **Cliente/Servidor**: son los sistemas que pueden compartir los recursos de sistemas muy distintos de una organización e incluso más allá de los límites de una organización.

- **DBMS(D**ata **B**ase **M**anagement **S**ystem**)**: Sistemas Generadores de Bancos o Bases de Datos. Son programas que se ocupan de accesar y actualizar las bases de datos, como Access, Visual Fox Pro, Sql Server, Oracle. Actúan como una interfaz entre el programa de aplicación y la base de datos. Por ejemplo, cuando un programa debe actualizar una información en la base de datos, no lo hace directamente, sino que le

pide al DBMS hacerlo, pues este es quien conoce cómo es la base de datos por dentro.

- **ODBC(O**pen **D**atabse **Co**nnectivity**)**: Conectividad Abierta de Bases de Datos, es un conjunto de funciones que permiten acceder prácticamente a cualquier servidor de bases de datos, mediante un manejador sin tener que cambiar el programa.

- **CGI(C**ommon **G**ateway **I**nterface**)**: son programas que se encuentran en un servidor en Internet y que al ser ejecutados procesan determinados datos. El procesamiento de datos ocurre en el servidor. Programas de CGI pueden memorizar en el servidor, por ejemplo, cuantas páginas han sido accesas, o que ha escrito un visitante en un libro de visitas. Un programa CGI puede también por ejemplo generar código HTML llamando y leyendo datos ya memorizados. Esas páginas generadas "dinámicamente" son enviadas al navegador del usuario y pueden ser representadas en la pantalla. La llamada interfaz o interfaz CGI debe ser apoyada por la software del servidor WWW (Web).

- Lenguaje **Perl**: **PERL** significa **P**ractical **E**xtraction and **R**eport **L**anguage, algo así como lenguaje práctico de extracción y de informes. Es un lenguaje creado por Larry Wall con el objetivo principal de simplificar las tareas de administración de un sistema Unix; en realidad hoy en día se ha convertido de propósito general, y una de las principales herramientas de un buen webmaster (Administrador Web), encontrando su aplicación en la escritura de CGI ejecutados desde páginas de la World Wide Web.

  PERL es un lenguaje que hereda estructuras principalmente de los intérpretes de comandos de Unix; es un lenguaje interpretado, aunque en realidad, el intérprete de **PERL** como todos los intérpretes modernos, compila los programas antes de ejecutarlos.

Aunque desarrollado originalmente en un entorno Unix, actualmente hay versiones para casi todos los sistema operativos: DOS, Windows NT, MacOS. Los scripts con compatibles entre diversas plataformas, de forma que es un verdadero lenguaje multiplataforma.

En general, los programas en **PERL** se ejecutan en el servidor a diferencia de otros programas ejecutados por el cliente (generalmente un navegador como el Internet Explorer o el Navigator de Netscape), como aquellos escritos en JavaScript o Java.

Mediante una serie de módulos adicionales, tales como el DBD o el ODBC, **PERL** puede servir para acceder a bases de datos, desde DBMS gratuitas como **MySQL** hasta **Microsoft SQL Server** usando ODBC.

- **ASP (A**ctive **S**erver **P**ages**)**: La tecnología ASP ha sido diseñada por Microsoft para facilitar la creación de sitios web con una sencillez mayor que la empleada en la programación CGI.

- **ISAPI(I**nternet **S**erver **A**plication **P**rograming Interface**)**: Interfaz de Programación de Aplicaciones de Servidor Internet. Es la interfaz propuesta por Micro$oft como una alternativa más rápida que el CGI, y ya está incluida en el servidor Microsoft Internet Information Server. Una aplicación ISAPI es una DLL (Dinamic Link Library) de Windows que se ejecuta en el mismo espacio de direcciones que el servidor Web. Estas aplicaciones cuyo código es reentrante, pueden soportar las peticiones simultáneas de diversos clientes con una sola imagen en memoria.

- **ADO (A**ctivex **D**ata **O**bjects**):** es una tecnología ampliable y de fácil uso para agregar acceso a bases de datos a páginas web. Se puede utilizar ADO para escribir secuencias de comandos compactas y escalables que conecten con bases de datos compatibles con ODBC y orígenes de datos compatibles con OLE DB.

- **Visual Basic Script:** Microsoft Visual Basic Script Edition, un conjunto del sistema de programación Visual Basic. Los programas de Visual Basic Script se pueden ejecutar en el servidor web o en el equipo donde está instalado el explorado del web.

- **Script:** en la programación de computadoras, un script es un programa o una secuencia de instrucciones que es interpretado o ejecutado por otro programa en vez del procesador del computador (es como un programa compilador).

  Algunos lenguajes se han concebido expresamente como lenguajes script. Entre los más populares están PERL, JavaScript y TCL/Tk.

  En general, los lenguajes script son más fácil y rápido de codificar que los lenguajes más estructurados y compilados como C y C++, y son ideales para programas de capacidad limitada o pueden ser rehusados y agregados a programas compilados existentes.

- **Driver**: Controlador, traducido del inglés.
- **Handle**: Manejador, traducido del inglés.

Abreviaturas:

- **IIS**: Internet Information Server.
- **VBScript**: Visual Basic Script.
- **JScript**: Java Script.

Funcionamiento de la Integración de Bases de Datos en el Web

En este apartado toca abordar la manera en que se da la integración de bases de datos en el web mediante las más comunes tecnologías.

Para realizar una requisición de acceso desde el Web hasta una base de datos no sólo se necesita de un navegador del Web y de un Servidor Web, sino también de un software de procesamiento (aplicación CGI o ASP para este caso), el cual es el programa que es llamado directamente desde un documento HTML en el cliente. Dicho programa lee la entrada de datos desde que provienen del cliente y toma cierta información de variables de ambiente. El método usado para el paso de datos está determinado por la llamada al programa.
Una vez se reciben los datos de entrada (sentencias SQL o piezas de ellas), el software de procesamiento los prepara para enviarlos a la interfaz en forma de SQL, y luego ésta procesa los resultados que se extraen de la base de datos.

La interfaz contiene las especificaciones de la base de datos necesarias para traducir las solicitudes enviadas desde el cliente, a un formato que sea reconocido por dicha base. Además, contiene toda la información, estructuras, variables y llamadas a funciones, necesarias para comunicarse con la base de datos.

El software de acceso usualmente es el software distribuido con la base de datos, el cual permite el acceso a la misma, a través de solicitudes con formato. Luego, el software de acceso recibe los resultados de la base de datos, aún los mensajes de error, y los pasa hacia la interfaz, y ésta a su vez, los pasa hasta el software de procesamiento.
Cualquier otro software (servidor HTTP, software de redes, etc.) agrega enlaces adicionales a este proceso de extracción de la información, ya que el software de procesamiento pasa los resultados hacia el servidor Web, y éste hasta el navegador del Web (ya sea directamente o a través de una red).

Categorización de Interfaces Web/DBMS

Tradicionalmente en el Web se han utilizado documentos HTML estáticos para los cuales se creaban las posibles respuestas ante requisiciones del cliente. Este método requiere de un gran desarrollo de aplicaciones y de mantenimiento de las mismas. Al interactuar con las bases de datos, este proceso se complica aún más.

Como la necesidad de acceder a bases de datos desde el Web se ha incrementado, han sido creadas también interfaces que manipulan sus escritos para procesar la información, teniendo como punto común la ejecución de sentencias SQL para requerir datos a la base.

Aplicaciones de interfaz para la interacción de bases de datos con el Web han surgido ya. Los productos iniciales son simplemente modelos del ambiente cliente/servidor, con una capa adicional para crear resultados HTML que pueden ser vistos a través del Web, por medio de un procesamiento de los datos de la forma introducidos por el cliente. Además, al usar estas interfaces se puede crear el programa principal de la aplicación. Estas herramientas permiten construir poderosas aplicaciones en el Web, pero se requiere del mantenimiento de las mismas, lo cual es significativamente complejo y extenso.
Una de las estrategias más famosas para la creación de aplicaciones de interacción con el Web, es la de descargar del Web, aplicaciones o componentes funcionales que se ejecutarán dentro del navegador. Con ellas se realizará un procesamiento complejo del lado del cliente, lo cual requiere un gran esfuerzo para crear las piezas de la aplicación. Estas estrategias poseen dos características principales:

1. Garantizan la seguridad tanto en los sistemas de distribución como en la comunicación que se establece con tales aplicaciones, a través de Internet.

2. También han aparecido bibliotecas que incluyen motores propios de servidor que corren de forma conjunta con el Servidor Web, lo cual facilita el desarrollo de nuevas aplicaciones.

Una aplicación que posibilita interconectar al Web con una base de datos tiene muchas ventajas, además de que las funciones que cumplen actualmente los Servidores Web y las herramientas de desarrollo de aplicaciones Web, hacen más fácil que nunca la construcción de aplicaciones más robustas. Tal vez el mayor beneficio del desarrollo de estas aplicaciones en el Web sea la habilidad de que sean para múltiples plataformas, sin el costo de distribuir múltiples versiones del software.

Cada una de las interfaces para comunicar al Web con bases de datos, ha sido creada basándose en una tecnología de integración especial, a través de procesos de interconexión especiales, que serán descritos posteriormente.

Seguridad:

La evaluación de este punto es uno de los más importantes en la interconexión del Web con bases de datos. Al nivel de una red local, se puede permitir o impedir, a diferentes usuarios el acceso a cierta información, pero en la red mundial de Internet se necesita de controles más efectivos en este sentido, ante posible espionaje, copia de datos, manipulación de éstos, etc.

La identificación del usuario es una de las formas de guardar la seguridad. Las identidades y permisos de usuarios están definidas en los Archivos de Control de Acceso.
Pero la seguridad e integridad total de los datos puede conservarse, permitiendo el acceso a distintos campos de una base de datos, solamente a usuarios autorizados para ello.
En este sentido, los datos pueden ser presentados a través del Web de una forma segura, y con mayor impacto en todos los usuarios de la red mundial.

Para la integración de bases de datos con el Web es necesario contar con una interfaz que realice las conexiones, extraiga la información de la base de datos, le dé un formato adecuado de tal manera que puede ser visualizada desde un navegador del Web, y permita lograr sesiones interactivas entre ambos, dejando que el usuario haga elecciones de la información que requiere.

V. MATERIALES

Este proyecto se llevó a cabo en la Universidad Nacional Autónoma de Nicaragua (UNAN - LEON), en el período comprendido entre los meses de Enero a Mayo del presente año.

Las implementaciones del presente proyecto se realizó en una computadora como servidor con las siguientes características:

- Procesador Intel Pentium II MMX.
- 64 MB de RAM.
- Capacidad de HD 3.5 GB.
- Sistema Operativo Windows NT 4.0, con Service Pack 6 instalado.
- Utilizando IIS versión 4.0: servidor Web y FTP.
- Intérprete de Perl instalado, desarrollado por ActiveState.
- Utilizando los manejadores ODBC versión 3.0 de Microsoft.

VI. METODOLOGÍA

1. INSTALACIÓN DE IIS versión 4.0 (SERVIDOR WEB Y FTP)

Para la instalación de Internet Information Server, específicamente el servidor Web y FTP, debe tenerse en cuenta las siguientes consideraciones generales:

- **Windows NT 4.0 y Service Pack 3** El servicio Servidor de Windows NT® debe estar instalado y ejecutarse.
- Al instalar IIS 4.0 en un equipo que ejecuta Windows NT Server, los servicios Inicio de sesión y Examinador de equipos deben estar en ejecución. Si no se ejecutan estos servicios, verá un cuadro de diálogo que dice "No se puede detectar el tipo de sistema operativo" y el programa de instalación fallará.
- **Quitar el software Alpha, Beta 1, o Beta 2** En caso de tener IIS versión 4.0 Alpha o IIS versión 4.0 (Beta 1 o Beta 2), hay que asegurarse de desinstalar esa versión (antes de instalar la versión actual). Para ello, en primer lugar hay que iniciar el programa de instalación del disco compacto que contiene la versión anterior o bien, en Windows NT, hacer clic en **Inicio**, seleccionar **Programas**, **Microsoft Internet Information Server (Común)** y, a continuación, hacer clic en **Instalación de Internet Information Server**. Hacer clic en **Siguiente** y, a continuación, en **Quitar todo**.

Se producirán errores de instalación de ODBC durante el programa de instalación si hay cualquier otra aplicación o servicio del sistema en ejecución que utilice el software de ODBC existente en el sistema. Si ocurre algún error de ODBC, hay que detener todas las aplicaciones de escritorio y de servicios del sistema, y ejecutar de nuevo el programa de instalación.

Otras consideraciones:

Reinstalar Windows NT Service Pack 3

En caso de que si se reinstala Windows NT Service Pack 3 o superior por cualquier razón después de instalar IIS 4.0, no se debe sobrescribir los nuevos archivos que instaló IIS 4.0.

Desinstalar el producto

Para desinstalar el producto, se debe ejecutar el programa de instalación de esta versión. Hay varias formas de tener acceso al programa de instalación Puede utilizar el disco compacto o hacer clic en **Inicio**, seleccionar **Programas**, **Windows NT Option Pack 4.0** y, a continuación, hacer clic en **Instalación de Windows NT Option Pack 4.0**. Seguir las instrucciones de los cuadros de diálogo y, cuando se le pida, haga clic en el botón **Agregar o Quitar** para desinstalar el software.

2. CONFIGURACIÓN Y ADMINISTRACIÓN DE IIS (SERVIDOR WEB Y FTP)

Internet Information Server incluye una herramienta de administración gráfica llamada Administrador de servicios de Internet que puede utilizarse para monitorizar, configurar y controlar los servicios de Internet.

El Administrador de servicios de Internet es la ubicación central desde la que se pueden controlar todos los equipos que ejecuten Internet Information Server en su organización. El Administrador de servicios de Internet puede ejecutarse desde cualquier equipo con Windows NT Workstation o Windows NT Server y que se encuentre conectado a través de la red con su servidor Web. Con la administración remota puede administrarse los servidores Web desde el propio equipo servidor, desde una estación de trabajo de administración en la red de área local (LAN) de la organización o, incluso, desde Internet.

El Administrador de servicios de Internet utiliza el modelo de seguridad de Windows NT, por lo que sólo los administradores validados podrán administrar servicios y las contraseñas de los administradores se transmiten de forma codificada a través de la red.

Además del Administrador de servicios de Internet, Internet Information Server incluye un Administrador de servicios de Internet para HTML que puede ejecutarse desde cualquier explorador de Web. Es posible realizar las mismas tareas administrativas desde cualquiera de las dos versiones del Administrador de servicios de Internet. Cualquier referencia al *Administrador de servicios de Internet* se refiere a ambas versiones de la herramienta, a menos que se indique lo contrario.

Administrador de servicios de Internet de Microsoft

El Administrador de servicios de Internet ayuda a configurar y monitorizar todos los servicios de Internet que se ejecuten en cualquier equipo con Windows NT en una red.

Conexión a servidores Web

Es posible administrar cualquier Internet Information Server de la red conectándose a él en el Administrador de servicios de Internet. Puede especificarse un servidor Web escribiendo el nombre de host con el Sistema de nombres de dominios (DNS), su dirección IP (Protocolo Internet) o su nombre NetBIOS (o nombre de equipo).

También puede encontrarse todos los equipos de su red que ejecuten Internet Information Server.

Para conectarse a un servidor Web

1. En el menú **Propiedades** del Administrador de servicios de Internet, seleccione **Conectar a servidor**.

2. En el cuadro **Nombre de servidor**, escriba el nombre de host del servidor Web, su dirección IP o el nombre NetBIOS.

Para conectarse seleccionando un servidor Web de una lista

1. En el menú **Propiedades** del Administrador de servicios de Internet, seleccione **Buscar todos los servidores**.

2. En la lista de servidores que aparece, haga clic en el servidor al que desee conectarse.

Selección de una vista

El Administrador de servicios de Internet muestra una vista gráfica de los servicios que se ejecutan en sus servidores. Puede verse un informe completo u ordenar la información por tipo de servicio o por nombre de equipo. Las vistas permiten ver rápidamente cuáles son los servicios que se están ejecutando. También puede mostrarse u ocultarse servicios, así como ordenarlos por su estado (en ejecución, en pausa o detenidos).

Para seleccionar una vista

- En el menú **Ver**, elija **Ver servidores**, **Ver servicios** o **Ver informe**. Estas vistas se describirán en las próximas secciones.

Para ordenar la información de una vista

- En el menú **Ver**, elija **Ordenar por servidor**, **Ordenar por servicio**, **Ordenar por comentario** u **Ordenar según estado**. Por ejemplo, para ver rápidamente cuáles son los servicios que se están ejecutando actualmente, deberá realizar una ordenación según estado.

Para mostrar u ocultar servicios

- En el menú **Ver**, elija el servicio que desee mostrar u ocultar (FTP, Gopher o WWW).

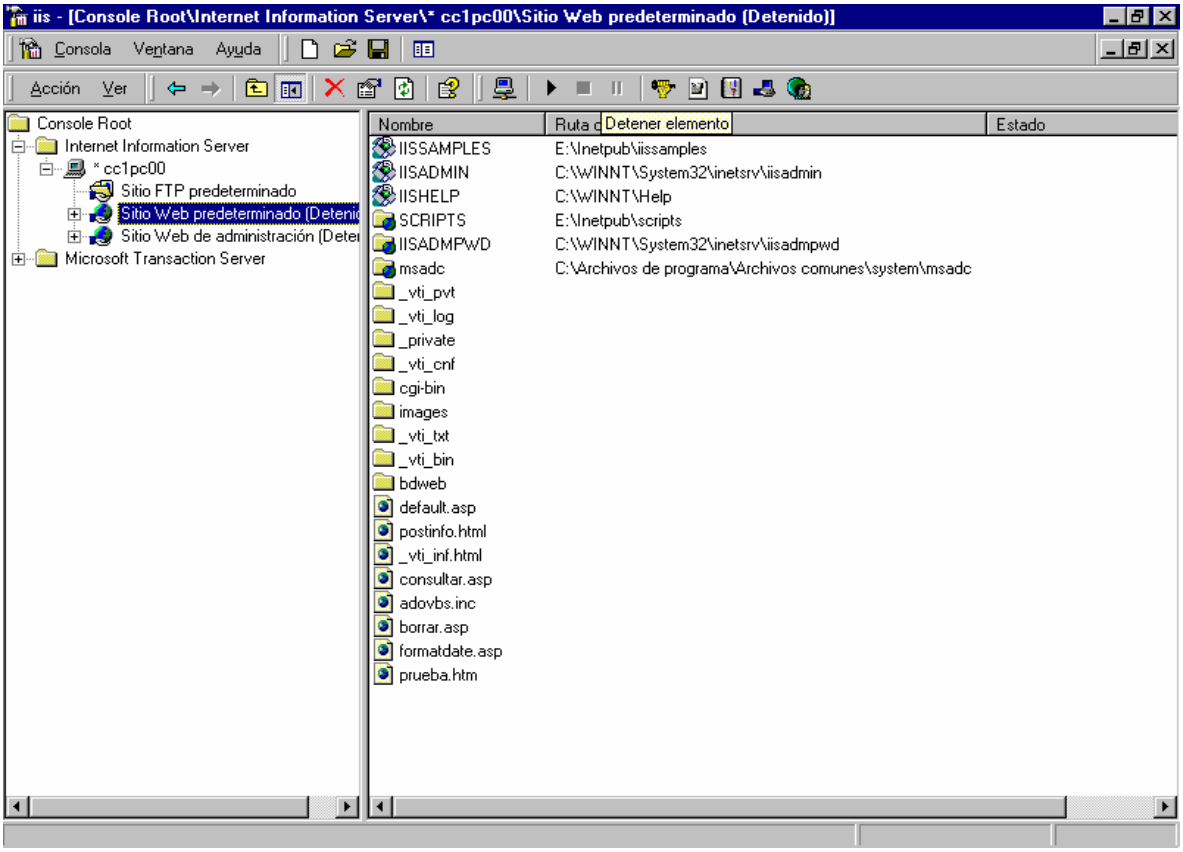
Ver Informe

El modo de ver Informe es el predeterminado. Este modo muestra alfabéticamente los equipos seleccionados, con sus servicios instalados en líneas separadas. Para esto hay que hacer clic en los encabezados de columna para ordenar alfabéticamente toda la lista. El modo de ver Informe es probablemente el más útil en sitios con uno o dos equipos que ejecuten Internet Information Server.[1]

[1] En caso de ejecutar otros servicios de Internet, como el Protocolo de transferencia de noticias de red (NNTP) y el Protocolo de transferencia de correo simple (SMTP), éstos aparecen en la lista que emite la vista de informe del Administrador de servicios de Internet, junto con los servicios WWW, FTP y Gopher.

La siguiente figura enumera las funciones de los botones e iconos del Administrador de servicios de Internet; para realizar las mismas funciones también puede usar los menús desplegables Propiedades y Ver.

Fig. Consola IIS



Conectar con servidores y ver hojas de propiedades

1. Conecta con un servidor Web específico.
2. Busca todos los servidores Web de la red.
3. Presenta hojas de propiedades para configurar el servicio seleccionado.

Iniciar, detener y hacer una pausa en un servicio

4. Inicia el servicio seleccionado.
5. Detiene el servicio seleccionado.
6. Hace una pausa en el servicio seleccionado.

Seleccionar los servicios que se deben presentar

7. Presenta el servicio FTP en la ventana principal del Administrador de servicios de Internet.
8. Presenta el servicio WWW en la ventana principal del Administrador de servicios de Internet.

Iniciar el Administrador de claves para crear una clave de Security Sockets Layer

9. Muestra la ventana del Administrador de claves.

Hacer los ajustes necesarios en los servicios

10. Ordena las listas cuando hace clic en los encabezados de columna.
11. Presenta las hojas de propiedades de un servicio cuando hace doble clic en él.
12. Presenta el estado del servidor y del servicio.

Inicio, detención y pausa en un servicio

Desde el Administrador de servicios de Internet se puede iniciar, detener o poner en pausa rápidamente un servicio.

Para iniciar, detener y hacer una pausa en un servicio

1. En el Administrador de servicios de Internet, seleccione el servicio que desee iniciar, detener o poner en pausa.
2. En el menú **Propiedades**, elija **Iniciar servicio**, **Detener servicio** o **Hacer una pausa en el servicio**.

Configuración y administración de servicios

Es posible configurar y administrar los servicios WWW, FTP y Gopher utilizando el Administrador de servicios de Internet. La siguiente información se centra en el servicio WWW.

En el Administrador de servicios de Internet, hacer doble clic en el nombre de un equipo o de un servicio para presentar sus hojas de propiedades. Luego Hacer clic en la ficha que aparece en la parte superior de cada hoja de propiedades para ver las propiedades de cada

categoría. Después de establecer las propiedades del servicio, hacer clic en **Aceptar** para volver a la ventana principal del Administrador de servicios de Internet.

La hoja de propiedades Servicio

La hoja de propiedades **Servicio** se utiliza para controlar el tipo de autentificación necesaria para obtener acceso a su sitio Web y especificar la cuenta utilizada para las peticiones de conexión al servidor de clientes anónimos. La mayoría de los sitios Internet permiten conexiones anónimas.

La hoja de propiedades Directorios

La hoja de propiedades **Directorios** permite especificar cuáles son los directorios (carpetas) a los que los usuarios pueden tener acceso y crear un sitio Web compuesto de carpetas que se encuentran en distintos equipos. También puede designar un documento predeterminado que aparece si un usuario remoto no especifica un archivo en particular o habilitar el examen de directorios. El examen de directorios significa que el usuario ve una lista en hipertexto de los directorios y archivos para que pueda desplazarse por su estructura de directorios.

La hoja de propiedades Registro

La hoja de propiedad **Registro** se utiliza para registrar la actividad del servicio. El registro proporciona información muy valiosa acerca de cómo se utiliza un servidor Web. Puede enviar los datos de registro a archivos de texto o a una base de datos compatible con Open Database Connectivity (ODBC). En caso de tener varios servidores o servicios Web en una red, puede registrarse toda su actividad en una única base de datos de cualquier equipo de la red.

Utilizando la hoja de propiedades **Registro**, también puede seleccionar el formato que desee para el registro, ya sea el formato Estándar o el formato de archivos de registro común del National Center for Supercomputing (NCSA).

La hoja de propiedades Avanzadas

La hoja de propiedades **Avanzadas** se utiliza para evitar que ciertas personas o grupos de personas obtengan acceso a su sitio Web. El acceso se controla especificando la dirección IP de los equipos a los que se otorga o deniega el acceso.

También se puede definir el ancho máximo de banda de la red para el tráfico de salida con el fin de controlar (regular) la cantidad máxima de tráfico en el sitio.

Limitación del uso de la red

Es posible restringir los servicios de Internet limitando el ancho de banda de la red permitido para todos los servicios de Internet de su servidor Web.
La limitación del ancho de banda dedicado a los usuarios de Microsoft Internet Information Server es especialmente útil si la línea Internet tiene varios propósitos. Limitar el ancho de banda permite que otras operaciones (como correo electrónico e inicios de sesión remotos) utilicen la misma línea sin que disminuya su velocidad por la gran cantidad de actividad del servidor Web.

Para cambiar el ancho de banda

1. En el Administrador de servicios de Internet, hacer doble clic en cualquiera de los servicios del equipo para el que desee cambiar el ancho de banda permitido.
2. Hacer clic en la ficha **Avanzadas**.
3. Seleccionar **Limitar el uso de la red por todos los servicios de Internet en este equipo**.
4. Seleccionar el número de kilobytes por segundo que se desea permitir para los servicios de Internet.
5. Hacer clic en **Aplicar** y, a continuación, en **Aceptar**.

Aunque el ancho de banda utilizado permanezca por debajo del nivel definido, las peticiones de información de los clientes se responden. Si el ancho de banda se encuentra cercano al valor definido, las peticiones de los clientes se demoran hasta que disminuya el tráfico de la red. El retraso de las respuestas permite que el servidor Web suavice el volumen del tráfico de la red sin denegar realmente las peticiones del explorador. Si el ancho de banda sobrepasa el nivel definido, las peticiones de lectura de archivos por parte

de los clientes se rechazarán, mientras que las peticiones de transferencia de archivo se retrasarán hasta que el ancho de banda iguale o caiga por debajo del valor definido.

Utilización de un explorador para administrar Internet Information Server

El programa HTML del Administrador de servicios de Internet de realiza las mismas funciones administrativas que el Administrador de servicios de Internet. Se puede utilizar el HTML del Administrador de servicios de Internet con un explorador de Web para administrar Internet Information Server en Internet. Para utilizar el HTML del Administrador de servicios de Internet, hay que ejecutar un explorador Web para abrir http://*nombre_archivo*/iisadmin. Para administrar cualquiera de los servicios, se debe conectar a una cuenta de usuario que tiene privilegios de Administrador en el equipo que se está administrando. Si se utiliza un explorador que puede realizar la autentificación Desafío/Respuesta de Windows NT (como la versión 2.0 o posterior de Microsoft Internet Explorer), se puede utilizar dicha autentificación. De lo contrario, se tendrá que utilizar la autentificación Básica (aunque no es recomendable).

Importante Cuando se administre de forma remota un servidor Web mediante un explorador, hay tres acciones que deben evitarse.

- Si el explorador sólo es compatible con la autentificación Básica, no hay que desactivarla mientras esté administrando Internet Information Server.
- Si se detiene un servicio, se perderá la conexión y no será capaz de restablecerla utilizando el HTML del Administrador de servicios de Internet.
- Si se elimina el directorio virtual Iisadmin en el servidor que se está administrando, no será capaz de utilizar el HTML del Administración de servicios de Internet en ese equipo.

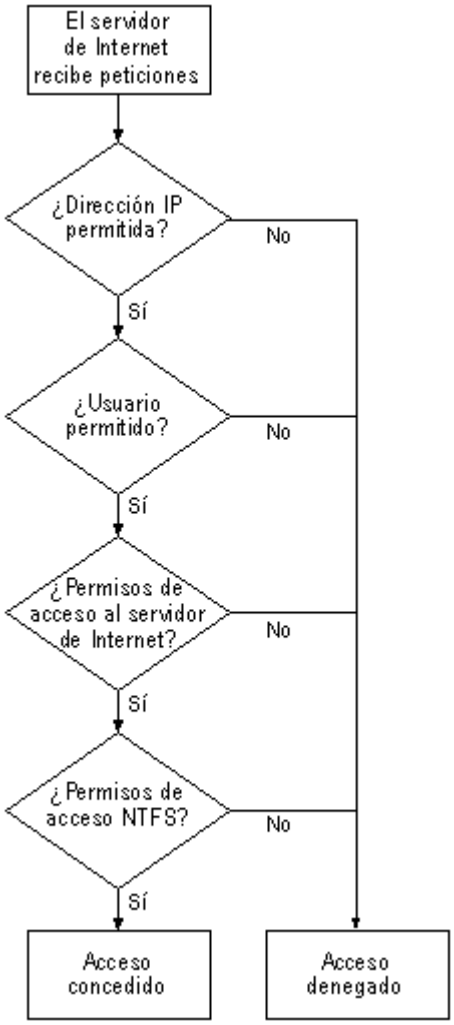
Funcionamiento de la seguridad de Internet Information Server

Internet Information Server se ha construido basándose en el modelo de seguridad de Windows NT. La seguridad de Windows NT ayuda a proteger el equipo y sus recursos mediante el requisito de cuentas de usuario y contraseñas asignadas. Se puede controlar el acceso a los recursos del equipo limitando los derechos de usuario de estas cuentas. Puede utilizarse el Sistema de archivos de Windows NT (NTFS) para asignar permisos a las carpetas y los archivos del equipo. Puede controlarse el acceso a carpetas y archivos impidiendo que los usuarios copien archivos hacia o desde una carpeta, o impidiendo que los usuarios ejecuten archivos en ciertas carpetas.

Además de las características de seguridad de Windows NT, puede establecerse directorios virtuales de sólo lectura o de sólo ejecución utilizando el Administrador de servicios de Internet. Internet Information Server también proporciona una manera de denegar el acceso de los usuarios a equipos con determinadas direcciones IP. es compatible con el protocolo Secure Sockets Layer (SSL), que codifica con seguridad las transmisiones de datos entre clientes y servidores.

Cuando un servidor Web de IIS recibe una petición de información de un explorador, determina si la petición es válida.

En la siguiente ilustración se muestra una simple perspectiva general del proceso de seguridad que se utiliza en cada petición.



Control de accesos anónimos

En muchos servidores Web, la mayoría del acceso a WWW, FTP y Gopher es anónimo; es decir, la petición del cliente no contiene un nombre y una contraseña de usuario. Esto ocurre en los siguientes casos:

- Un cliente FTP inicia una sesión con el nombre de usuario “anonymous”.
- Todas las peticiones de Gopher.
- La petición de un explorador de Web no contiene un nombre y una contraseña de usuario en el encabezado HTTP (esta es la forma predeterminada en las conexiones Web nuevas con la mayoría de los exploradores).

Aunque el usuario no inicie la sesión con un nombre y una contraseña de usuario individual, todavía se puede controlar y monitorizar el acceso anónimo. Cada servicio de Internet mantiene un nombre de usuario y una contraseña de Windows NT que se utiliza para procesar las peticiones anónimas. Cuando se recibe una petición anónima, el servicio "suplanta" al usuario configurado como usuario con "inicio de sesión anónimo". La petición tiene éxito si el usuario con inicio de sesión anónimo tiene permiso de acceso al recurso solicitado, según lo determina la ACL (Lista de control de acceso) del recurso. Si el usuario con inicio de sesión anónimo no tiene permiso, la petición falla. Puede configurarse el servicio WWW para que responda a un petición anónima fallida solicitando al usuario que proporcione un nombre y una contraseña válidos de Windows NT, proceso que se denomina autentificación.

Configuración de la cuenta de usuario anónimo

Es posible ver y monitorizar la cuenta de usuario con inicio de sesión anónimo en las hojas de propiedades **Servicio** del Administrador de servicios de Internet (para los servicios WWW, FTP y Gopher). Cada servicio que se ejecuta en el mismo equipo puede utilizar las mismas cuentas o bien cuentas de usuario con inicio de sesión anónimo diferentes. La inclusión de la cuenta de usuario con inicio de sesión anónimo en las ACL de un archivo o de una carpeta le permite controlar con precisión los recursos a disposición de clientes anónimos.

La cuenta de usuario con inicio de sesión anónimo debe ser una cuenta de usuario de Windows NT válida en el servidor que proporciona los servicios Web, y la contraseña debe coincidir con la contraseña para este usuario en la base de datos de usuarios de ese equipo. Las cuentas de usuario y las contraseñas se configuran en el Administrador de usuarios de Windows NT mediante el establecimiento de **Derechos de usuario** en el menú **Directivas**. La cuenta de usuario con inicio de sesión anónimo debe tener el derecho de usuario **Iniciar sesión en modo local**.

La cuenta IUSR_*nombreequipo* (con una contraseña generada aleatoriamente) se crea automáticamente durante la instalación de Internet Information Server. Por ejemplo, si el

nombre del equipo es CC1PC00, el nombre de la cuenta de acceso anónimo es IUSR_CC1PC00.

De forma predeterminada, todas las peticiones de clientes Web utilizan esta cuenta. Es decir, los clientes de Web se conectan al equipo utilizando la cuenta IUSR_*nombreequipo*. A la cuenta IUSR_*nombreequipo* sólo se le permite iniciar sesión en modo local en el servidor que proporciona los servicios Web.[2]

Para los servicios WWW y FTP, se puede permitir o evitar accesos anónimos (todas las peticiones Gopher son anónimas). Para cada uno de los servicios Web (WWW, FTP y Gopher), se puede cambiar la cuenta de usuario que se utiliza para peticiones anónimas y cambiar la contraseña para esa cuenta.

Para permitir accesos anónimos

1. En el Administrador de servicios de Internet, hacer doble clic en el servicio WWW o en el servicio FTP para mostrar sus hojas de propiedades; a continuación, hacer clic en la ficha **Servicio**.

2. En el caso del servicio WWW, activar la casilla de verificación **Permitir anónimos**. En el servicio FTP, active la casilla de verificación **Permitir conexiones anónimas**.
3. Hacer clic en **Aceptar**.

Para cambiar la cuenta o la contraseña utilizada para accesos anónimos

1. En el Administrador de servicios de Internet, haga doble clic en el servicio para mostrar sus hojas de propiedades y, a continuación, haga clic en la ficha **Servicios**.

2. En el cuadro de nombre de usuario **Inicio de sesión anónima**, escriba el nombre de usuario nuevo.

[2] La cuenta IUSR_*nombreequipo* también se agrega como miembro del grupo Invitados. Si se cambian los valores de dicho grupo, esos cambios también se aplicarán a la cuenta IUSR_*nombreequipo*. Se debe revisar los valores del grupo Invitados para asegurarse de que son adecuados para la cuenta IUSR_*nombreequipo*.

La cuenta de usuario predeterminada es IUSR_*nombreequipo*, donde *nombreequipo* es el nombre de su servidor. Esta cuenta se crea automáticamente al instalar Internet Information Server.

3. En el cuadro **Contraseña**, escriba la nueva contraseña.

   Se crea automáticamente una contraseña generada aleatoriamente para la cuenta IUSR_*nombreequipo*.[3]

4. Hacer clic en **Aceptar**.

Control del acceso por usuario o por grupo

Es posible controlar el acceso a su sitio Web utilizando el Administrador de usuarios de Windows NT con el fin de especificar lo que se permite hacer a ciertos usuarios o grupos en su servidor. Se puede controlar aún más el acceso solicitando a las peticiones de clientes Web que proporcionen un nombre de usuario y una contraseña que confirma Internet Information Server antes de completar la petición.

Configuración de las cuentas de usuario

La seguridad de Windows NT ayuda a proteger el equipo y sus recursos al requerir cuentas de usuario asignadas. Todas las operaciones en un equipo que ejecuta Windows NT identifican quién está haciendo la operación. Por ejemplo, el nombre de usuario y la contraseña que utiliza para iniciar la sesión en Windows NT identifica quién es y define lo que se le autoriza a hacer en ese equipo.

Las acciones que un usuario puede realizar en un equipo se configuran con el Administrador de usuarios definiendo **Derechos del usuario** en el menú **Directivas**. Los derechos del usuario autorizan a un usuario para la realización de ciertas acciones en el

[3] Si se cambia la contraseña para esta cuenta, debe especificar también la contraseña nueva para la cuenta en el Administrador de usuarios.

sistema, incluyendo el derecho de **Iniciar sesión en modo local**, que es necesario para que los usuarios puedan emplear los servicios de Internet si se está utilizando la autentificación Básica.

En caso de utilizar la autentificación Desafío/Respuesta de Windows NT, entonces se requiere el derecho **Acceso al equipo desde esta red** para que los usuarios utilicen los servicios de Internet. De forma predeterminada, todo el mundo tiene este derecho.
Para aumentar la seguridad:

- No conceder a la cuenta IUSR_*nombreequipo*, al grupo Invitados o al grupo Todos cualquier derecho que no sea el de **Iniciar sesión en modo local** o **Acceso al equipo desde esta red**.
- Asegurarse de que todas las cuentas de usuario del sistema, especialmente aquellas con derechos administrativos, tienen dificultad para adivinar contraseñas. En particular, elija una buena contraseña de administrador (la mejor es una larga, con mayúsculas y minúsculas mezcladas, y alfanumérica) y defina los planes de cuentas apropiados. Las contraseñas se pueden establecer utilizando el Administrador de usuarios o escribiéndolas en el símbolo de inicio de sesión en el sistema.
- Asegurarse de especificar con qué rapidez caducan las contraseñas de las cuentas (lo que obliga a los usuarios a cambiar las contraseñas periódicamente) y establecer otras medidas como cuántos inicios de sesión fallidos se permitirán antes de bloquear a un usuario. Utilice estas medidas para impedir ataques a la contraseña exhaustivos o al azar, especialmente en cuentas con acceso administrativo. Puede establecer esta medidas utilizando el Administrador de usuarios.
- Limitar el número de miembros del grupo Administradores a personas de confianza.
- En caso de utilizar las cuentas de usuario INTERACTIVA y RED predefinidas de Windows NT para el control de acceso, asegúrese de que los archivos de su sitio Web se encuentran accesibles para estas cuentas de usuario. Para que las peticiones de clientes anónimos o de clientes que utilizan la autentificación Básica tengan acceso a un archivo, el usuario INTERACTIVO debe tener acceso al archivo solicitado. Para que la petición de un cliente que utiliza el protocolo de

autentificación Desafío/Respuesta de Windows NT tenga acceso a un archivo, el usuario RED debe tener acceso al archivo.

Requerir un nombre de usuario y una contraseña

Se puede restringir el acceso a un sitio Web solamente a los clientes *autentificados*, es decir, a clientes de Web que proporcionan un nombre de usuario y una contraseña válidos de Windows NT. Cuando utiliza la autentificación, no se permite el acceso a menos que se proporcione un nombre de usuario y una contraseña válidos. La autentificación de la contraseña es útil si se desea autorizar el acceso solamente a personas individuales a su sitio Web o a partes específicas controladas por NTFS. Se puede tener activados al mismo tiempo tanto el acceso de inicio de sesión anónimo como el acceso autentificado.

El servicio WWW proporciona dos formas de autentificación: Básica y Desafío/Respuesta de Windows NT (al que a veces se denomina “NTLM”).

La autentificación Básica no codifica las transmisiones entre el cliente y el servidor. Puesto que la autentificación Básica envía la contraseña y el nombre de usuario de Windows NT del cliente en forma esencialmente decodificada a través de las redes, los intrusos podrían aprender fácilmente los nombres y las contraseñas de los usuarios.
La autentificación Desafío/Respuesta de Windows NT, compatible actualmente tan sólo con la versión 2.0 o posterior de Microsoft Internet Explorer, protege la contraseña, proporcionando un inicio de sesión seguro a través de la red. En la autentificación Desafío/Respuesta de Windows NT, la cuenta de usuario que se obtiene de un cliente es con la que el usuario inicia la sesión en el equipo cliente. Puesto que esta cuenta, incluyendo su dominio de Windows NT, debe ser una cuenta válida en el servidor basado en Windows NT que ejecuta Internet Information Server, la autentificación Desafío/Respuesta de Windows NT es muy útil en un entorno de intranet, donde el equipo del cliente y el del servidor están en los mismos dominios o en los de confianza. Debido al aumento en la seguridad, Microsoft recomienda utilizar el método de autentificación de contraseñas Desafío/Respuesta de Windows NT siempre que sea posible.

De forma predeterminada están activadas tanto la autentificación Básica como la autentificación Desafío/Respuesta de Windows NT. Si el explorador es compatible con Desafío/Respuesta de Windows NT, utiliza ese método de autentificación. De lo contrario, utiliza la autentificación Básica. La autentificación Desafío/Respuesta de Windows NT sólo es compatible actualmente con la versión 2.0 o posterior de Internet Explorer.

Puede requerir la autentificación del cliente para todas las peticiones del servicio FTP o sólo para las peticiones anónimas que fallan. El servicio FTP sólo es compatible con la autentificación Básica; por tanto, su sitio está más seguro si permite conexiones anónimas. La mayor seguridad para su sitio se consigue permitiendo solamente las conexiones FTP anónimas.

Para habilitar la autentificación para el servicio WWW

1. En el Administrador de servicios de Internet, hacer doble clic en el servicio WWW para mostrar sus hojas de propiedades y, a continuación, hacer clic en la ficha **Servicio**.
2. Seleccionar **Básica**[4] **(Texto simple)**, **Autentificación Windows NT** o ambas.
3. Hacer clic en **Aceptar**.

Para habilitar la autentificación para el servicio FTP

1. En el Administrador de servicios de Internet, hacer doble clic en el servicio FTP para mostrar sus hojas de propiedades y, a continuación, hacer clic en la ficha **Servicio**.
2. Para habilitar la autentificación para conexiones anónimas fallidas, desactivar la casilla de verificación **Permitir sólo conexiones anónimas**.
3. Para solicitar la autentificación de todas las peticiones de los clientes, desactivar la casilla de verificación **Permitir conexiones anónimas**.

Como interactúan los inicios de sesión anónimos y las autentificaciones de clientes

Puede habilitar tanto conexiones anónimas como autentificación de clientes para el servicio WWW y para el servicio FTP. Esta sección explica cómo un servidor Web de IIS responde a estos métodos de acceso cuando ambos están activados.

[4] **Advertencia** La autentificación Básica de FTP y WWW envía contraseñas a través de la red en forma de texto simple (es decir, sin codificar), al igual que la autentificación Básica de HTTP.

Si no está permitida la autentificación de cliente y sí las conexiones anónimas, una petición de cliente que contenga una contraseña y un nombre de usuario se procesará como una conexión anónima, y el servidor pasará por alto la contraseña y el nombre de usuario.

Servicio WWW

Cuando un servicio WWW recibe la petición de un cliente que contiene credenciales (un nombre de usuario y una contraseña), la cuenta de usuario del "inicio de sesión anónimo" no se utiliza en el procesamiento de la petición. En su lugar, el servicio utiliza el nombre de usuario y la contraseña recibidos por el cliente. Si al servicio no se le concede permiso de acceso al recurso solicitado mientras utiliza el nombre de usuario y la contraseña especificados, la petición falla y se devuelve una notificación de error al cliente.

Cuando una petición anónima falla porque la cuenta de usuario de "inicio de sesión anónimo" no dispone de permiso de acceso al recurso deseado, la respuesta al cliente indica qué esquemas de autentificación son compatibles con el servicio WWW. Si la respuesta indica al cliente que el servicio está configurado para la autentificación Básica de HTTP, la mayoría de los exploradores de Web mostrarán un cuadro de diálogo de nombre de usuario y contraseña, y volverán a realizar la petición anónima como una petición con credenciales, incluyendo el nombre de usuario y la contraseña introducidos por el usuario.

Si un explorador de Web es compatible con el protocolo de autentificación Desafío/Respuesta de Windows NT y el servicio WWW está configurado para este protocolo, una petición WWW anónima que falle debido a los permisos inadecuados traerá como consecuencia una utilización automática del protocolo de autentificación Desafío/Respuesta de Windows NT. Entonces, el explorador enviará un nombre de usuario y una contraseña codificada desde el cliente al servicio. La petición del cliente se vuelve a procesar, utilizando la información de usuario del cliente.

Si el servicio WWW está configurado para ser compatible tanto con la autentificación Básica como con Desafío/Respuesta de Windows NT, el explorador de Web devuelve ambos métodos de autentificación al explorador de Web en un encabezado. Entonces, el explorador de Web elige qué método de autentificación va a utilizar. Como el protocolo Desafío/Respuesta de Windows NT aparece el primero en el encabezado, un explorador que sea compatible con el protocolo Desafío/Respuesta de Windows NT lo utilizará. Un explorador que no sea compatible con ese protocolo de Windows NT utilizará la autentificación Básica. Actualmente, sólo la versión 2.0 o posteriores de Internet Explorer son compatibles con la autentificación Desafío/Respuesta de Windows NT.

Servicio FTP

Cuando el servicio FTP recibe la petición de un cliente que contiene credenciales (nombre de usuario y contraseña), la cuenta de usuario de “inicio de sesión anónimo” no se utiliza en el procesamiento de la petición. En su lugar, el servicio utiliza el nombre de usuario y la contraseña recibidos por el cliente. Si no se concede al servicio permiso de acceso al recurso solicitado durante la utilización del nombre de usuario y la contraseña especificados, la petición falla y se devuelve al cliente una notificación de error.

Cuando falla una petición anónima porque la cuenta de usuario de “inicio de sesión anónimo” no tiene permiso de acceso al recurso solicitado, el servidor responde con un mensaje de error. La mayoría de los exploradores de Web mostrarán un cuadro de diálogo de nombre de usuario y contraseña, y volverán a realizar la petición anónima como una petición con credenciales, incluyendo el nombre de usuario y la contraseña introducidos por el usuario.[5]

[5] **Advertencia** Puesto que el servicio FTP (y la autentificación Básica de WWW) envía nombres de usuario y contraseñas sin codificar a través de la red, los intrusos podrían utilizar analizadores de protocolo para leer los nombres de usuario y las contraseñas.

Configuración de los directorios de contenido

Se puede configurar el servicio WWW para que pueda publicar desde varios directorios utilizando el Administrador de servicios de Internet. La hoja de propiedades **Directorios** enumera los directorios de contenido utilizados por el servicio WWW.

Se han creado los siguientes directorios[6]:

- ***bdweb***: (directorio hijo de wwwroot) donde reside la aplicación ASP de este proyecto.
- ***cgi-bin***: (directorio virtual con permiso de ejecución) donde residen los scripts cgi de este proyecto.

Creación de directorios virtuales

En su servicio puede crear un número casi ilimitado de directorios virtuales, aunque es posible que si crea demasiados pueda verse afectado el rendimiento.

Para crear un directorio virtual

1. En el Administrador de servicios de Internet, hacer doble clic en el servicio al que desee agregar un directorio virtual para ver sus hojas de propiedades.
2. Hacer clic en la ficha **Directorios**.
3. Hacer clic en el botón **Agregar**.
4. Hacer clic en **Examinar** para seleccionar un directorio en el cuadro **Directorio**.
5. Hacer clic en **Directorio virtual** y, a continuación, escriba el nombre del directorio virtual en el cuadro **Alias**.
6. Establecer los permisos de acceso.
7. Hacer clic en **Aceptar**.
8. Hacer clic en **Aplicar** y, a continuación, haga clic en **Aceptar**.

[6] Ver Fig. “Consola IIS.”

3. FORMULARIOS

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

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

La sintaxis del marcador de formulario es la siguiente:

```
<FORM METHOD="método" ACTION="URL del SCRIPT ASP  O CGI">
```

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

Campos en formularios

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

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

Los tipos válidos de campos de formulario son los siguientes:

- **text**. Campo de tipo texto.

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

Resultado: valor

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

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

Resultado: *****

- **radio** . Botón de selección redondo. Selección múltiple.

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

Resultado:

- **checkbox**. Boton de selección.

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

Resultado:

- **submit**. Botón de aceptación de la información introducida en el formulario.

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

Resultado: Submit

- **reset**. Borra el contenido completo del formulario.

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

Resultado: Reset

- **image**. Imagen que actúa como botón.
- **hidden**. Campo oculto. No es visible al cliente WWW.

Estos controles admiten los modificadores siguientes, ya indicados:

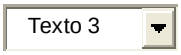
- **size**. Tamaño del campo.
- **width**.
- **maxlength**.

Otros controles tienen una sintaxis especial.

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

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

Resultado:



El control <OPTION> permite los modificadores siguientes:

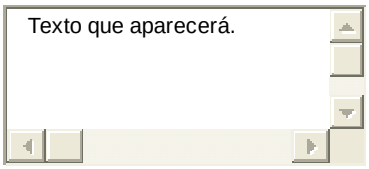
- **Size**.

El siguiente control permite introducir texto de cualquier longitud.

```
<TEXTAREA NAME="nombre">
```

Texto que aparecerá en el recuadro. Es opcional.

Resultado:



Este control acepta los atributos específicos

- **ROWS**. Número de filas.
- **COLS**. Número de columnas.

4. PROGRAMACIÓN ASP

Inicialmente, las páginas web eran estáticas, en el sentido, que a efectos de usuario, el único proceso realizado era el de visualización de sus contenidos (escritos en lenguaje HTML) por parte del navegador web del cliente.

Actualmente se siguen utilizando ampliamente las páginas web estáticas debido a que forman la base necesaria para la presentación de datos en muchos tipos de situaciones, y porque su creación es sencilla y fácil de instalar y mantener.

Surge entonces la necesidad de requerir una interacción mayor entre los usuarios y el sistema que soporta las páginas web, es decir la necesidad de reunir y procesar las peticiones del cliente con el fin de ofrecerle informaciones mejor dirigidas, escogidas y elaboradas. Por lo tanto es conveniente dotar de alguna fase de procesamiento al intercambio de información entre los usuarios y el servidor de páginas web, con lo que es posible introducir el concepto de páginas web activas o dinámicas que el modelo ASP soporta.

La capacidad de procesamiento que sustenta las páginas web dinámicas se puede llevar a cabo siguiendo alguno de estos modelos:

- Procesamiento en el equipo del usuario: páginas activas en el cliente.
- Procesamiento en el equipo donde reside el servidor Web: páginas activas en el servidor.
- Procesamiento mixto: páginas activas en el cliente más páginas activas en el servidor.

La tecnología ASP ha sido diseñada por Microsoft para facilitar la creación de sitios web con una sencillez mayor que la empleada en la programación CGI. Microsoft introdujo esta tecnología llamada **Active Server Pages** en diciembre de 1996, por lo que no es nada nueva. Es parte del **Internet Information Server** desde la versión 3.0 y es una tecnología

de páginas activas que permite el uso de diferentes scripts y componentes en conjunto con el tradicional HTML para mostrar páginas generadas dinámicamente.

Traduciendo la definición de Microsoft: "*Las Active Server Pages son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el web*".

En ASP, todas las páginas web pueden ser diseñadas con editores de HTML, puesto que las instrucciones ejecutables y el código HTML están suficientemente delimitados. El principio de la tecnología ASP es el VBScript, pero existe otra diversidad de lenguajes de programación que pueden ser utilizados como lo es Perl, JScript.

El ASP es una tecnología dinámica funcionando principalmente del lado del servidor, lo que significa que cuando el usuario solicita un documento ASP, las instrucciones de programación dentro del script son ejecutadas para enviar al navegador únicamente el código HTML resultante. Pero también permite compatibilizar la creación de páginas web activas en el cliente y en el servidor, pudiéndose así balancear la carga de proceso y de comunicaciones según los deseos que se requieran en una aplicación. ***La ventaja principal de las tecnologías dependientes del servidor radica en la seguridad que tiene el programador sobre su código, ya que éste se encuentra únicamente en los archivos del servidor que al ser solicitado a través del web, es ejecutado, por lo que los usuario no tienen acceso más que a la página resultante en su navegador***.

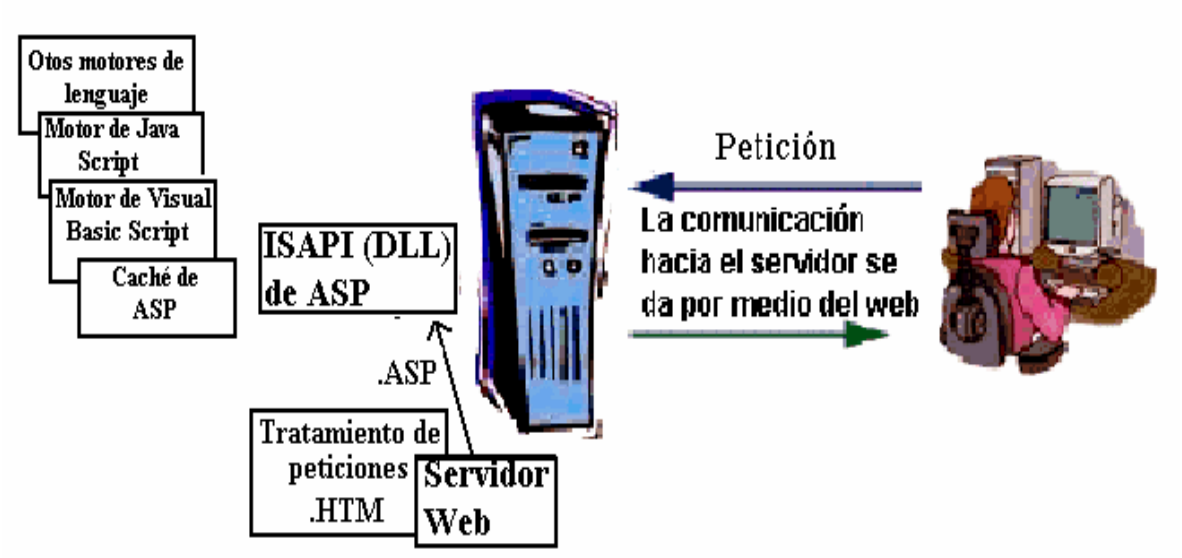
Funcionamiento de ASP

Para que un usuario realice una petición de página Web, deberá proporcionar en su explorador web una dirección que indique un archivo de extensión ".asp".

Cuando se trabaja con IIS y ASP, el servidor Web analiza las peticiones de página que recibe. Si se encuentra con una solicitud de página de extensión ".asp" en lugar de ".htm", entonces se apoya en la aplicación ISAPI que sirve de soporte de ejecución de las páginas ASP. Una muestra de ello en la siguiente gráfica:

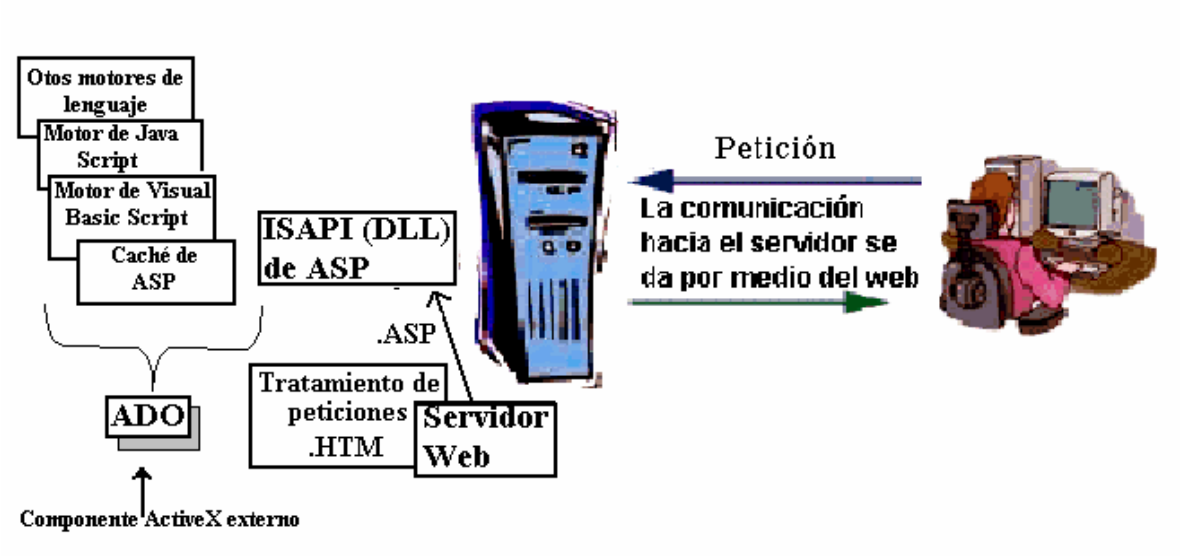
La aplicación ISAPI de ASP diferencia las líneas HTML de las instrucciones que dan la funcionalidad dinámica a las páginas activas. Cuando determina el lenguaje en el que se encuentran los programas (Scripts), da paso al motor de ejecución de scripts adecuado (Vscript, Jscripts, etc.). Los motores de ejecución de scripts se encargan de realizar el análisis sintáctico y la compilación de las instrucciones ejecutables. Existe una memoria caché de páginas recientemente procesadas que permite aumentar las prestaciones de ASP,

evitando repetir los procesos de separación de instrucciones, análisis sintáctico y compilación de las páginas más utilizadas.



Una vez resueltas las fases anteriores, se procede a ejecutar las instrucciones.

Los motores de ejecución de scripts a menudo se encuentran con objetos ActiveX exteriores con los que tienen que interactuar. Un ejemplo muy importante de esta situación se centra en el acceso a bases de datos a través de **ADO**, basado en tecnología **COM** (Component Object Model).



Resumiendo mejor el funcionamiento del ASP se presenta la siguiente gráfica:

Para insertar instrucciones ASP dentro del código HTML se incluye encerrado entre "<% %>". Estos comandos son los que procesa el servidor antes de enviar la página al navegador.

A continuación un ejemplo del código ASP en el servidor y los resultados HTML que serán vistos en el navegador:

Código ASP	Código HTML	Resultado en el Navegador
<P> <% For I = 1 To 5 Step 1 %> <FONT SIZE="<%= I %>">Active Server Pages</FONT> <% Next %> </P>	<P> <FONT SIZE="1"> Active Server Pages </FONT> <FONT SIZE="2"> Active Server Pages</FONT> <FONT SIZE="3"> Active Server Pages</FONT> <FONT SIZE="4"> Active Server Pages</FONT> <FONT SIZE="5"> Active Server Pages</FONT> </P>	Active Server Pages Active Server Pages Active Server Pages Active Server Pages Active Server Pages

El ejemplo anterior crea un ciclo que se repite 5 veces y aumenta el tamaño del tipo de letra en una frase establecida.

El desarrollo que se ha venido dando a lo que es ASP ha sido bastante amplio. Entre sus funciones principales están el acceso a base de datos, envió de correo electrónico, creación dinámica de gráficos y otros. Básicamente, muchas cosas que podemos realizar por medio de CGI pueden ser realizadas con esta tecnología. Esto debido a que el ASP es tan eficiente con escribir código directamente a la interfaz de aplicación del servidor, con la ventaja de que es más eficiente que el CGI que depende de un compilador ya que el ASP corre como un servicio en el servidor, tomando ventaja de la arquitectura de multitareas.

ACCESO A BASES DE DATOS CON ASP

Una de las mejores características que nos ofrece ASP es la facilidad en el uso de bases de datos. Esto se realiza utilizando Actives Data Objects (ADO), componentes ActiveX, los cuales ofrecen un acceso uniforme y sencillo a una base de datos de cualquier tipo, ya sea Access, SQL Server, Oracle, etc.

OBJETOS ADO

El modelo de bases de datos basado en ActiveX Data Objects (ADO) está formado, como su nombre indica, por objetos. Estos objetos nos ofrecen una serie de métodos y propiedades con los que podemos acceder fácilmente a las bases de datos que creamos conveniente.

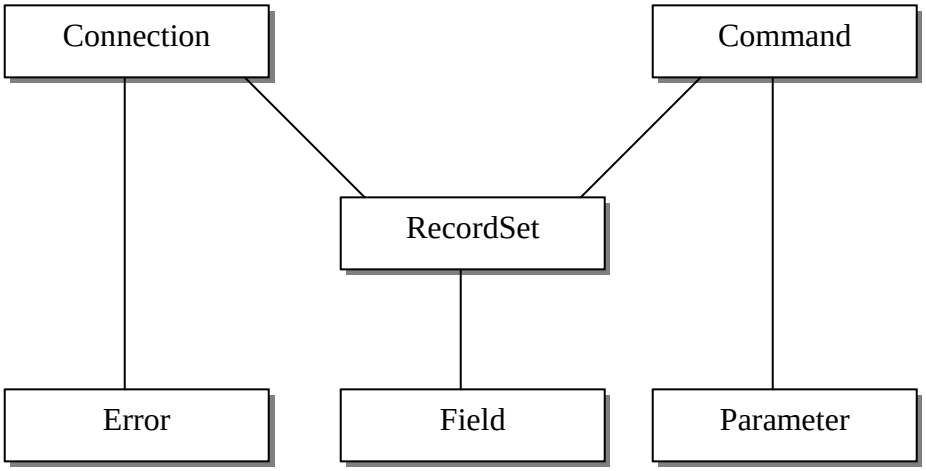
Para manejar este tipo de bases de datos tenemos siete objetos. De estos siete tres son muy importantes y principales: Connection, Recordset y Command[7]. Estos objetos se pueden instanciar directamente desde una página web. El resto de objetos: Field, Parameter, Property, Error, se utilizan como complementarios de los principales, y a partir de éstos, es decir después de haber creado alguno de los anteriores.

[7] Para saber más de las propiedades y métodos de estos objetos ver Apéndice.

Cada objeto tiene la siguiente función:

- ***Connection***: representa la conexión con una base de datos. Este objeto se utiliza para crear un enlace directo entre una página web y la bases de datos. Mientras dure la conexión es posible realizar todas las operaciones deseadas sobre la base de datos. La conexión terminará cuando así se indique con el método ***Close*** de este objeto.
- ***RecordSet***: representa una tabla de datos. En este objeto será donde se almacenan las consultas realizadas a la base de datos a la que se esté conectado. Estará formado por filas (registros) y columnas (campos) a los que se puede acceder para exponer la información adquirida.
- ***Command***: representa un comando SQL. Con este objeto se puede ejecutar sentencias SQL sobre la base de datos a la que se esté conectado.
- ***Field***: representa un campo de un objeto RecorSet. Este objeto solo existirá si existe el correspondiente objeto RecordSet. El objeto RecordSet lleva implícito la colección Fields que representa todos sus campos. Cada elemento de esa colección es un objeto Field.
- ***Parameter***: representa un parámetro de un procedimiento o cuestión. Es de gran ayuda al utilizar el objeto Command para lanzar procedimientos o cuestiones sobre una base de datos.
- ***Error***: representa un error ADO. Se puede producir un error al realizar la conexión sobre la base de datos. Esto quiere decir que este objeto sólo existirá si previamente se ha intentado conectar con una base de datos erróneamente.
- ***Property***: representa una propiedad específica de un proveedor de datos. Este objeto se encuentra un poco al margen de los demás, ya que no tiene relación alguna con ellos.

Las relaciones que existen entre cada objeto se puede ver en la gráfica siguiente:



Connection-RecordSet: una cualidad que tiene el objeto Connection es la de poder ejecutar comandos SQL. Al hacer esto puede darse el caso de que el comando devuelva cierto resultado, por ejemplo, si el comando realiza una consulta sobre la base de datos. En tal caso el resultado vendrá dado dentro de un objeto RecordSet.

Command-RecordSet: al igual que en el caso anterior, al ejecutar una consulta con el objeto Command sobre una base de datos el resultado viene dado en un objeto RecordSet.

Command-Connection: cuando se crea un objeto Command no será utilizable hasta que no se relacione con una base de datos. Para esto es necesario el objeto Connection. Con este objeto se abre una sesión con una base de datos y se luego se asigna al objeto Command como conexión activa. Sólo de esta manera se podrá ejecutar los comandos de este objeto sobre una base de datos.

Connection-Error: el objeto Error sólo aparece cuando se produce una conexión errónea a una base de datos. Con este objeto y sus métodos y propiedades se puede determinar la causa del error.

RecordSet-Field: todo objeto RecordSet tiene entre sus atributos la colección Fields. Esta colección está formada por objetos del tipo Field. Estos objetos representan cada campo de

cada registro del objeto RecordSet. El objeto Field permite un acceso sencillo a la información del objeto RecordSet.

Command-Parameter: un comando SQL puede tener parámetros. Esos parámetros pertenecerán al objeto Parameter. Además no sólo se puede ejecutar sentencias simples de SQL con el objeto Command, sino también procedimientos que probablemente llevarán parámetros, tanto de entrada como de salida.

Conexión con una Base de Datos

El primer paso para tener acceso a la información de una base de datos consiste en establecer una conexión con un origen de datos. A través del objeto **Connection** que ADO proporciona se puede utilizar para establecer y administrar las conexiones entre sus aplicaciones y las bases de datos de ODBC.

La siguiente secuencia de comandos crea una instancia del objeto **Connection** y procede a abrir una conexión:

```
<% 'Crear un objeto Connection
Set cn = Server.CreateObject("ADO.Connection")
'Abrir una conexión, la cadena hace referencia al DSN.
cn.open "Conexión"
%>
```

Ejecución de Consultas con el Objeto Connection

Con el método **Execute** del objeto **Connection** se puede enviar consultas en el lenguaje SQL al origen de datos y recuperar los resultados.

La siguiente secuencia de comandos utiliza el método **Execute** del objeto **Connection** para enviar una consulta con un comando **Insert** de SQL, que inserta datos en una base de datos concreta.

```
<%
'Definir el DSN
strDSN = "Conexion"
'Crear la instancia del objeto Connection y abrir una conexión con la base de 'datos
set cn = Server.CreateObject("ADO.Connection")
cn.Open strDSN
'Definir la instrucción Insert de SQL
strSQL = "Insert into Cliente (Nombre, Apellidos) values ('Santiago', _ 'Molina')"
'Utilizar el método Execute para enviar una consulta SQL a la base de datos
cn.Execute(strSQL)
%>
```

Se puede usar los comandos **Update** y **Delete** de SQL para modificar y quitar información de la base de datos.

La siguiente secuencia de comando usa el comando **Update** para cambiar todos los campos Nombre de la tabla Cliente a "Juan" en todas las filas cuyo campo Apellido contenga el apellido Reyes

```
<%
Set cn = Server.CreateObject("ADO.Connection")
cn.Open "Conexion"
cn.Execute "Update Cliente set Nombre='Juan' where Apellido='Reyes'"
%>
```

Para quitar determinados registros de una tabla de la base de datos, se utiliza el comando **Delete**

```
<%
Set cn = Server.CreateObject("ADO.Connection")
cn.Open "Conexion"
cn.Execute "Delete from Cliente where Apellido='Reyes'"%>
```

Uso del objeto RecordSet para manipulación de resultados

Para recuperar datos, examinar resultados y modificar la base de datos, ADO proporciona el objeto **RecordSet**, que tiene las funciones necesarias para, dependiendo de las restricciones de las consultas, recuperar y presentar un conjunto de filas, o registros, de una base de datos. Éste mantiene la posición de cada registro devuelto por la consulta, lo que permite "recorrer" los resultados de uno en uno.

Recuperar un conjunto de registros

La siguiente secuencia de comandos utiliza el objeto **RecordSet** para ejecutar un comando **Select** de SQL.

```
<%
'Establecer una conexión con un origen de datos
strDSN = "Conexion"
Set cn = Server.CreateObject("ADO.Connection")
cn.Open strDSN
'Crear la instancia de un objeto RecordSet
Set rsCliente = Server.CreateObject("ADO.Recordset")
'Abrir un objeto Recordset con el método Open y utilizar la
conexión 'establecida por el objeto Connection
strSQL = "Select Nombre, Apellido from Cliente where
Apellido= 'Soto' "
```

```
rsCliente.Open strSQL, cn
‘Recorrer el conjunto de los registros y presentar los resultados e incrementar ‘la posición del registro con el método MoveNext
Set Nombre = rsCliente(“Nombre”)
Set Apellido = rsCliente(“Apellido”)
Do until rsCliente.EOF
     Response.write Nombre & “ ”& Apellido & “<br>”
     RsCliente.MoveNext
Loop
%>
```

Manejo de consultas con el objeto Command

Con el objeto **Command** de ADO se puede ejecutar consultas de la misma forma que con los objetos **Connection** y **RecordSet**, excepto que con el objeto **Command** se puede preparar, o compilar la consulta con el origen de la base de datos y después enviar la misma consulta repetidamente con distintos valores. La ventaja de compilar consultas de esta manera consiste en la enorme reducción del tiempo necesario para volver a enviar las modificaciones de una consulta existente. Además, se puede dejar las consultas SQL parcialmente indefinidas, con la posibilidad de alterar partes de las consultas justo antes de su ejecución

La colección **Parameter** del objeto **Command** ahorra los problemas de tener que volver a construir la consulta cada vez que se tenga que ejecutar. Por ejemplo:

```
<%
‘Abrir una conexión mediante el objeto Command del Objeto Connection no ‘tiene un método Open para establecer la conexión
strDSN = “Conexión”
```

```
Set cn = Server.CreateObject("ADO.Connection")
cn.Open strDSN
'Crear el objeto Command; utilizar la propiedad ActiveConnection para 'adjuntar la conexión al objeto Command
Set cm = Server.CreateObject("ADO.Command")
Set cm.ActiveConnection = cn

'Definir la consulta SQL
cm.CommandText = "Delete from Cliente where Nombre = 'Juan' "
cm.Execute
%>
```

Abandonar los intentos de conexión

Un servidor de base de datos que experimente un repentino incremento en su actividad puede quedar saturado en sus conexiones, lo que incrementa considerablemente el tiempo necesario para establecer una conexión nueva. Como resultado, los retrasos excesivos en la conexión pueden reducir el rendimiento de una aplicación de base de datos.

Con la propiedad **ConnectionTimeout** del objeto **Connection** puede limitar la cantidad de tiempo que la aplicación espera antes de abandonar un intento de conexión y emitir un mensaje de error.

Por ejemplo, la siguiente secuencia de comandos establece la propiedad **ConnectionTimeout** para esperar veinte segundos antes de cancelar el intento de conexión:

```
<%
Set cn = Server.CreateObject("ADO.Connection")
cn.ConnectionTimeout = 20
cn.Open "Conexion"
%>
```

El valor predeterminado de la propiedad **ConnectionTimeout** es 30 segundos.

Aplicaciones ASP

Global.asa → ASA (Active Server Application)

Global.asa es un archivo opcional en el que se puede especificar secuencias de comandos de eventos y declarar objetos con alcance de sesión o de aplicación.
No es un archivo de contenido que se presente a los usuarios; almacena información de eventos y objetos utilizados globalmente por la aplicación.

Este archivo debe llamarse Global.asa y estar almacenado en el directorio raíz de la aplicación. Una aplicación sólo puede tener un archivo Global.asa.
Los archivos Global.asa sólo pueden contener lo siguiente:

- Eventos de Aplicación (Application).
- Eventos de Sesión (Session).
- Declaraciones <OBJECT>
- Declaraciones TypeLibrary.

Si se incluye una secuencia de comandos que no esté enmarcada dentro de etiquetas <*SCRIPTS*>, o que defina un objeto que no tenga alcance de sesión o de aplicación, el servidor devuelve un error. El servidor pasa por alto las secuencia de comandos que se encuentren dentro de etiquetas no utilizadas por los eventos de la aplicación o de la sesión, así como cualquier código HTML del archivo.

Las secuencias de comandos contenidas en el archivo Global.asa pueden escribirse en cualquier lenguaje de secuencia de comandos compatible. Si varios eventos u objetos utilizan el mismo lenguaje de secuencia de comandos, pueden agruparse en un mismo conjunto de etiquetas <*SCRIPTS*>.

Cuando se guardan las modificaciones en el archivo Global.asa, el servidor termina de procesar todas las peticiones de la aplicación actual antes de volver a compilar el archivo Global.asa. Durante este tiempo, el servidor rechaza las nuevas peticiones y devuelve un

mensaje de error para indicar que no se puede procesar la petición mientras se reinicia la aplicación.

Una vez procesadas todas las peticiones actuales de los usuarios, el servidor elimina todas las sesiones activas, invocando el evento ***Session_OnEnd*** por cada sesión que vaya a eliminar, cierra la aplicación y llama al evento ***Application_OnEnd***.

Después se vuelve a compilar el archivo Global.asa. Las siguientes peticiones de los usuarios iniciarán la aplicación y crearán nuevas sesiones, y desencadenarán los eventos ***Application_OnStart*** y ***Session_OnStart***.

Sin embargo, guardar las modificaciones en un archivo incluido en el archivo Global.asa no hace que el servidor vuelva a compilar Global.asa.
Para que el servidor reconozca las modificaciones en el archivo incluido, debe volverse a guardar el archivo Global.asa.

Sólo se puede llamar a los procedimientos declarados en el archivo Global.asa desde una o varias de las secuencias de comandos asociadas a los eventos ***Application_OnStar***, ***Application_OnEnd***, ***Session_OnStart*** y ***Session_OnEnd***. No están disponibles para las páginas ASP de la aplicación ASP.

Para compartir procedimientos en toda una aplicación, puede declararse los procedimientos en un archivo independiente y después utilizar instrucciones de inclusión en el servidor (SSI: Server Side Include) para incluir el archivo en las páginas ASP que llamen a los procedimientos. Los archivos de inclusión suelen tener la extensión "**.inc**".

Características del Global.asa

1. Debe ser almacenado en el directorio raíz (root) de la aplicación y ASP lo lee cuando:
    - El servidor Web recibe la primera solicitud de inicio o de envío de un archivo .asp en una aplicación.
    - Un usuario quien no tiene una sesión solicita un archivo .asp de la aplicación.
2. Se puede incluir lo siguiente en un archivo global.asa:
    - Eventos de Inicio de Aplicación y/o eventos de inicio de Sesión.
    - Eventos de final de Aplicación y/o eventos de final de Sesión.
    - Etiquetas Objeto.
3. Eventos de Inicio de Sesión e Inicio de Aplicación:
    - Los eventos son Session_OnStart y Aplication_OnStart respectivamente.

Sintaxis:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Application_OnStart
     ‘Instrucciones
End Sub
</SCRIPT>
```

4. Eventos de Final de Sesión y Final de Aplicación:
    - Los eventos son Session_OnEnd y Aplication_OnEnd respectivamente.

Sintaxis:

```
<SCRIPT LANGUAGE=VBScript RUNAT=Server>
Sub Session_OnEnd
     ‘Instrucciones
End Sub
</SCRIPT>
```

Manejo de Aplicaciones y Sesiones

Inicio y finalización de una Aplicación

La aplicación inicia cuando se recibe por primera vez una solicitud de una página ASP y finaliza cuando el servidor Web es bajado o si la aplicación está configurada para correr en un proceso diferente al del servidor Web (aislada) se puede finalizar con el botón UNLOAD.

- Para aislar una Aplicación:

1. En Administrador de Servicios Internet, seleccionar el sitio Web o el directorio raíz de una aplicación.
2. Abrir la hoja de propiedades del directorio, y seleccionar la pestaña Directorio.
3. Seleccionar la propiedad Correr en un espacio de Memoria Separado.
4. Clic OK.

Uso del estado de Aplicación

Se usa para hacer disponible información a todos los usuarios de la aplicación. Esta información debe ser almacenada por medio del objeto **APPLICATION**.

Manejo de Sesiones:

Se usa para almacenar información de un usuario en particular y para colocar esta información disponible entre las páginas que dicho usuario visita.
Para lograr esto se debe almacenar la información en objetos **SESSION**.

Iniciando y finalizando una Sesión

Hay tres formas de iniciar una sesión:

- Una solicitud de una página .ASP y el global.asa tiene un procedimiento *Session_OnStart*.
- Un usuario guarda algún valor en el objeto Session.
- Cuando se instancia un objeto con <OBJECT> y se fija la propiedad scope a SESSION.

Una sesión finaliza cuando el usuario deja de refrescar una página durante un tiempo determinado o cuando por programación se hace finalizar.

```
<%Session.Timeout=5%><%Session.Abandon%>
```

VENTAJAS DE ASP:

Podría resumirse ahora las principales ventajas que presenta ASP, como sigue a continuación.

1. La información existente es de naturaleza restringida. Aunque se puede realizar diseños de páginas web que minimicen el ancho de banda necesario para las consultas o el procesamiento del servidor, los datos y una buena parte de los cómputos no pueden ser eliminados del lugar “central”.
2. Las ASP proporcionan un adecuado soporte de procesamiento en el servidor, a la vez que permiten su coexistencia con páginas activas en el cliente.
3. El acceso a bases de datos a través de OBDC es sencillo y potente. Se sustenta en componentes ActiveX que en este caso se ejecutan únicamente en el servidor.
4. Con ASP, las páginas web devueltas al cliente se diseñan y mantienen de una forma muy sencilla, lo que supone una de las características más importantes que las diferencian de otros tipos de productos desarrollados hasta el momento. Por otra parte el código que se envía al usuario es HTML estándar, con lo que se puede asegurar que la información se visualizará correctamente con independencia de la plataforma y el navegador web del cliente.
5. Los desarrollos realizados con ASP no necesitan compilaciones que retarden el proceso de producción, y los errores de programación no provocan la caída del servidor Web.

DESVENTAJAS DE ASP

Aunque ASP es una solución de integración de bases de datos en el web bastante sencilla presenta las siguientes desventajas:

1. La principal desventaja se refiera a la ejecución de páginas web con ASP, ya que requiere que el servidor sea Windows NT/2000, porque es en dicho sistema operativo que coexiste IIS en el que ASP está ligado, por lo que resulta inconveniente para muchos desarrolladores que están acostumbrados a trabajar en sistemas Unix, Linux, u otros diferentes a la alternativa Microsoft, puesto que ASP no se puede introducir de manera semejante en dicha plataformas.

2. De la anterior se derivan muchas otras de carácter comercial como: la cuestión de licenciamiento del software (Sistema Operativo, Gestor de Bases de Datos) utilizado, lo que genera gastos monetarios para una institución que pretende implementar soluciones Microsoft.

3. Cuando se utiliza procesamiento mixto: páginas activas en el cliente más páginas activas en el servidor, esto puedo producir mal funcionamiento de las páginas activas, debido especialmente a problemas de los navegadores web que utiliza el cliente diferentes a Internet Explorer (por. Ej. Netscape), suele suceder que al acceder a un lugar web, las páginas no se pueden visualizar correctamente, debido a que no se dispone de un interprete de lenguaje adecuado (por ejemplo, VBScript con Netscape), o bien a que no sea posible cargar algún control ActiveX.

4. Referente al lenguaje que mayormente utiliza ASP, Visual Basi Script, aunque es excelente para manejo de Bases de datos, usando ADO, en muchas ocasiones presenta limitantes en cuanto a validación de la información suministrada por el cliente, lo cual obliga recurrir a lenguajes como JavaScript para estos casos.

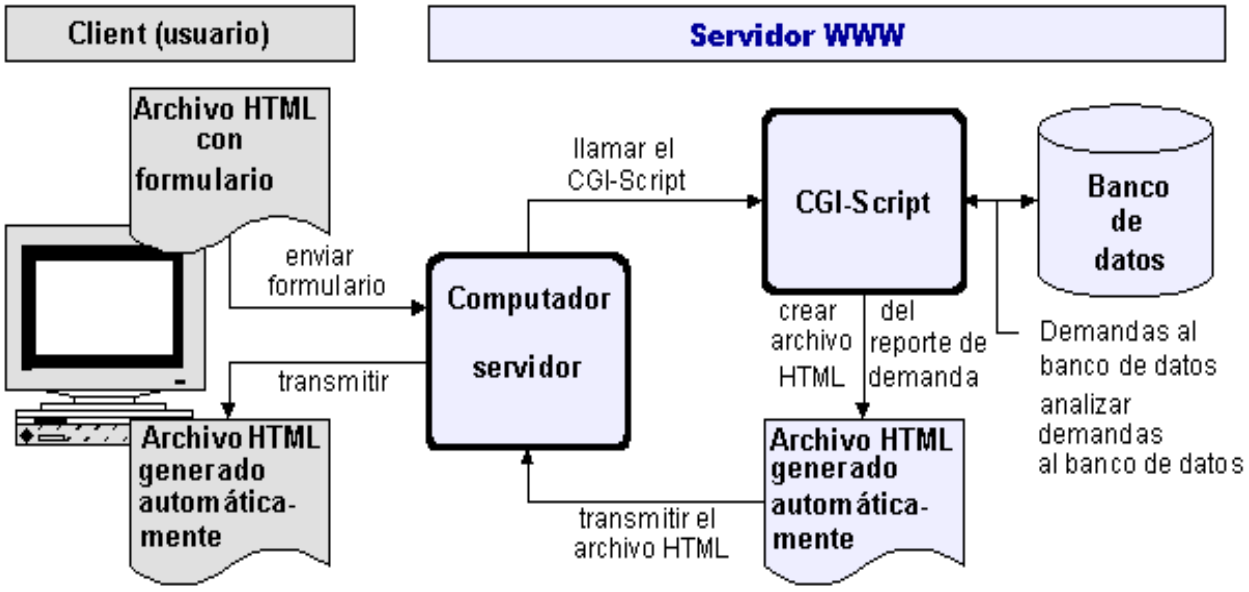
5. PROGRAMACIÓN CGI

Actualmente, ésta es la solución que más se está utilizando para la creación de interfaces Web/DBMS. Fue probada por primera vez en el servidor NCSA.

Se ha comprobado que si el Servidor Web recibe un URL con una llave, para devolver un documento HTML como respuesta, tendrá que cargar el servicio (programa) que le indique las variables de ambiente y de la forma HTML. La mayoría de las veces dicha llave es el "cgi-bin".

El Common Gateway Interface (Interfaz común de intercambio) es una posibilidad de poner a la disposición programas en Internet que pueden ser llamados con ayuda de páginas HTML y que pueden generar código HTML y enviarlos a un navegador.

El siguiente ejemplo muestra una típica situación CGI, por ejemplo como es necesaria en servicios de búsquedas en la WWW.



En el ejemplo un usuario puede hacer entradas en un archivo HTML que contiene un formulario, por ejemplo especificar la búsqueda en una base de datos (en la gráfica “banco de datos”). Un programa CGI es ejecutado cuando el usuario envía el formulario. El Script

transforma los datos del usuario en demandas a la base de datos. El funcionamiento de este procedimiento depende de la base de datos instalado. Existe un lenguaje internacional estandarizado de base de datos llamado SQL que muy a menudo entra en acción. La aplicación de la base de datos envía de vuelta los resultados de la búsqueda al programa CGI que ha sido ejecutado (o los escribe en un archivo, que puede ser leído por el programa CGI). El programa CGI genera código HTML tomando en cuenta los resultados de la búsqueda, o sea que los datos de la búsqueda son unidos o combinados en el código HTML. El código HTML es enviado por el programa CGI al navegador que ha hecho la demanda de búsqueda. En la pantalla del usuario desaparece la página WWW con el formulario y aparece una nueva página con los resultados de la búsqueda, la cual ha sido generada dinámicamente con ayuda del programa CGI.

Entrada y Salida CGI

El CGI define dos mecanismos para pasar datos desde el servicio WWW a una aplicación externa, el primero utiliza variables de entorno y el segundo la entrada estándar. La recuperación de resultados se produce siempre a través de la salida estándar.

El primer método recibe el nombre de método GET y se utiliza principalmente para búsquedas en documentos por palabra clave. El segundo método, llamado POST, se utiliza para procesar el contenido de formularios.

Método GET:

En el método GET la aplicación CGI recibe la información a través de variables de entorno. El proceso de lanzamiento de una aplicación CGI con el método GET es el siguiente.

1. El cliente WWW solicita un servicio de una aplicación CGI.
2. El servidor Web recibe la solicitud y los datos de entrada.
3. El servidor crea un entorno y crea variables en el con los datos de entrada.
4. El servidor ejecuta la aplicación CGI en este entorno.

5. La aplicación CGI procesa las variables de entorno y recupera los datos de entrada.
6. La aplicación CGI se ejecuta produciendo un resultado sobre su salida estándar.
7. EL servidor Web redirecciona la salida estándar de la aplicación CGI hacia el cliente WWW.
8. El cliente WWW recibe el resultado de su consulta.

Método POST:

El método POST es el método recomendado para el paso de información de formulario a una aplicación CGI. En este método la información se pasa a través de la entrada estándar de la aplicación CGI.

El proceso de lanzamiento de una aplicación CGI con el método POST es el siguiente.

1. El cliente WWW solicita un servicio de una aplicación CGI.
2. El servidor Web recibe la solicitud y los datos de entrada.
3. El servidor ejecuta la aplicación CGI pasándole la información a través de la entrada estándar.
4. La aplicación CGI procesa su entrada estándar y recupera los datos de entrada.
5. La aplicación CGI se ejecuta produciendo un resultado sobre su salida estándar.
6. EL servidor Web redirecciona la salida estándar de la aplicación CGI hacia el cliente WWW.
7. El cliente WWW recibe el resultado de su consulta.

Junto con las opciones del servidor el protocolo CGI se basa también en los mecanismos de los clientes para introducir información. A continuación revisamos estos mecanismos.

ISSEARCH

Introduciendo la directiva <ISSEARCH> en la página HTML, esta página se convierte en un documento de consulta. Estos documentos se generan por aplicaciones CGI y en ellos el cliente WWW añade un cuadro de introducción de texto con el cual se pueden realizar consultas al documento. Las consultas se resuelven por el CGI que generó la página.

Las consultas ISSEARCH se resuelven por el método GET.

IMAGEMAP

El método IMAGEMAP registra pulsaciones del ratón sobre una imagen. Por este método es posible conocer la posición de pulsación del ratón en el cliente WWW.

El uso principal de las imágenes pinchables es la creación de menús gráficos.
EL método IMAGEMAP se gestiona por una aplicación CGI específica que se incluye en la mayoría de servidores HTTP. En el NCSA HTTPD la aplicación se denomina imagemap, en CERN HTTPD la aplicación se denomina htimage.

Los CGI que gestionan la información del método IMAGEMAP utilizan mascaras donde se definen las acciones a ejecutar dependiendo de la posición de la imagen donde se haya pinchado.

COMPARACIÓN DE LOS MÉTODOS GET Y POST[8]:

La diferencia está en que los datos que se envían con el método POST no se guardan en la variable de entorno QUERY_STRING[9] de ningún programa CGI. En su lugar se guardan en la entrada estándar (STDIN) de dicho programa. La variable REQUEST_METHOD toma el valor de POST, mientras que los datos codificados se almacenan en STDIN del programa CGI y se invoca una nueva variable de entorno llamada CONTENT_LENGTH en la que se guarda el número de bytes almacenados en STDIN.

El programa CGI comprueba el valor de la variable REQUEST_METHOD. Si toma el valor de POST para las peticiones http POST, el programa tendrá que comenzar por determinar el tamaño de los datos introducidos, información que se encuentra en la variable CONTENT_LENGTH, y después leer datos que se encuentran en STDIN. El servidor Web

[8] Martínez Lorío, José Evenor. "Seguridad en el Servidor Apache con SSL". Trabajo de fin de carrera, julio 2000.
[9] Para más detalle ver la sección de Apéndice.

no es quien inserta la marca de final de archivo (EOF), por eso se utiliza la variable CONTENT_LENGTH, con cuyo valor se determina la longitud de los datos en bytes.

PROGRAMACIÓN DE BASES DE DATOS CON PERL

Existen diversos módulos desarrollados en PERL que permiten acceder a diversas bases de datos en plataformas Windows, entre los módulos más comunes se encuentran *WIN32::ODBC y DBI.* Este último es el que detallaremos a continuación, por el hecho de ser una solución bastante sencilla y fácil de utilizar para integrar bases de datos en el web.

Nombre:

- DBI – Interfaz independiente de bases de datos para PERL.

Plataformas soportadas:

- Linux.
- Solaris.
- Windows.

Descripción:

"DBI es un Interfaz de Programación de Aplicaciones de Acceso a Bases de Dados (API-Application Programing Interface) para el lenguaje Perl. La especificación del API DBI define un juego de funciones, variable y convenciones que proporcionan un interface de base de datos consistente e independiente de la base de datos que se esté utilizando."[10]

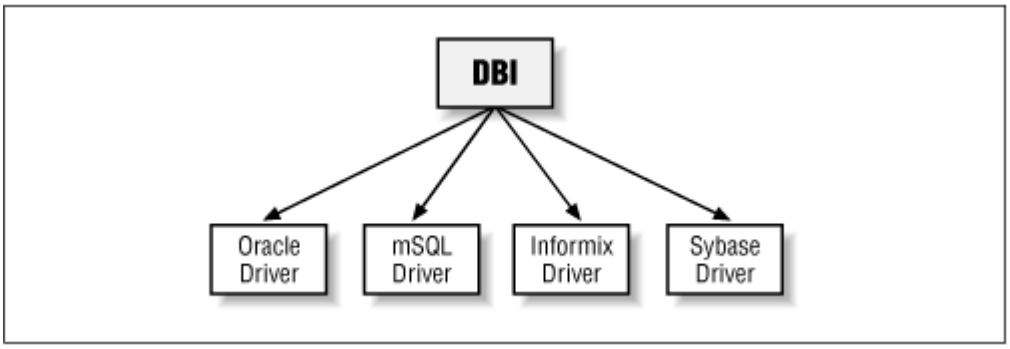
En un lenguaje sencillo, la interface DBI permite a los usuarios acceder a múltiples tipos de bases de datos de modo transparente. Por ello, si se está conectando a una Oracle, Informix,

[10] Tomado de Tim Bunce, el arquitecto y autor de DBI.

mSQL, Sybase, o cualquier otra base de datos, no se necesita conocer los mecanismos subyacentes de lenguaje 3GL. La API definida por DBI trabajara en todos estos tipos de datos. El secreto está en la librería DBD (Database Dependent) que viene con la distribución estándar de Perl. Esta librería permite al DBI comunicarse con cualquier base de datos.

DBI contiene una arquitectura que está dividida en dos grupos principales de software: el DBI en sí mismo, y los controladores[11]. El DBI define la interfaz de programación real de DBI, encamina llamadas a métodos de los controladores apropiados, y les proporciona varios servicios de ayuda. Los controladores específicos se ponen en ejecución para cada tipo de base de datos y realizan realmente las operaciones en la base de datos. La siguiente figura ilustra esta arquitectura.

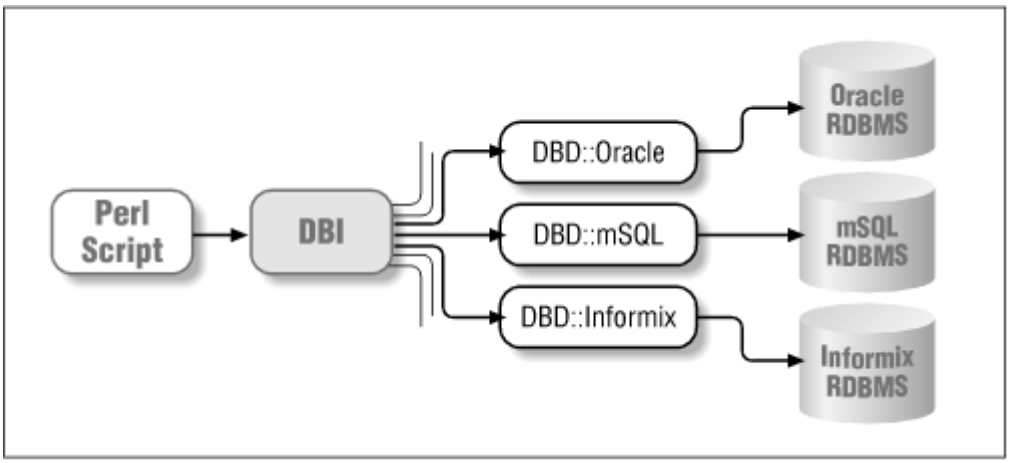
Fig. La arquitectura DBI



Por lo tanto cuando se está usando al interfaz de programación DBI, el método utilizado se define dentro del módulo DBI. De ahí, el módulo DBI resuelve que el controlador debe manejar la ejecución del método y pasa el método al controlador apropiado para la ejecución real. Se reconoce entonces que el módulo DBI en sí mismo no realiza ningún trabajo en la bases de datos.

[11] Del inglés “drivers”.

Fig. Flujo de Datos a través de DBI



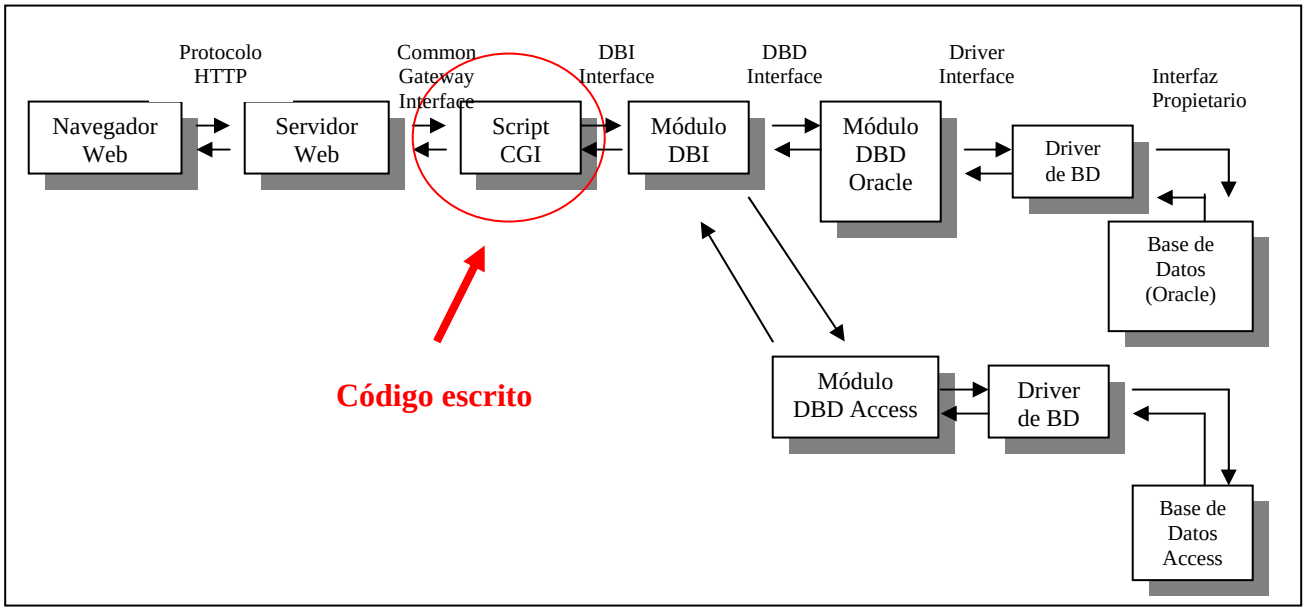
Bajo esta arquitectura, es relativamente mantener directo poner un controlador en ejecución para cualquier tipo de base de datos. Todo los que se requiere es implementar los métodos definidos en la especificación DBI, según lo utilizado por el módulo DBI, de una manera que sea significativa para esa base de datos. Los datos vueltos de este módulo se pasan nuevamente dentro del módulo DBI, y de él se vuelve al programa de Perl. Toda la información que pasa entre el DBI y sus controladores son tipos de datos estándares en Perl, con lo cual preservan el aislamiento del módulo DBI de cualquier conocimiento de la base de datos.

La separación de los controladores del DBI en sí mismo, hace del DBI una poderosa interfaz de programación de gran alcance que se puede extender para soportar casi cualquier base de datos disponible hoy. Los controladores existen para muchas base de datos populares incluyendo Oracle, Informix, mSQL, MySQL, Ingres, Sybase, DB2, Empress, SearchServer, and PostgreSQL. Hay controladores incluso para archivos Xbase y CSV.

Estos controladores pueden ser usados alternativamente con una pequeña modificación a los programas. El nivel de portabilidad de la base de datos con la portabilidad de los scripts de Perl a través de múltiples sistemas operativos, verdaderamente tiene una rápida aplicación en el desarrollo de herramientas dignas de mención.

Los controladores son también llamados controladores de base de datos, o DBDs, después del espacio de nombre en el cual son declarados. Por ejemplo, Oracle usa DBD::Oracle, Informix usa DBD::Informix, así sucesivamente. Un aviso para recordar del gran uso de la arquitectura DBI es que DBI puede situarse para independientes base de datos y para base de datos dependientes.

Véase la siguiente gráfica para entender cómo funciona todo el conjunto: Navegador, Servidor Web, DBI, DBD, Controladores y Base de datos.



El CGI habla con el módulo DBI y éste lo hace con el DBD que tiene un controlador para comunicarse con la base de datos, que para el caso de sistemas Windows mientras tenga el controlador (driver) para ODBC se encargará de lo demás, esto es manipular la respectiva base de datos.

Porque DBI usa las características orientadas a objetos de Perl, esto hace extremadamente simple para inicializar DBI para usar con los programas. Esto puede llevarse a cabo añadiendo la línea:

```
Use DBI;
```

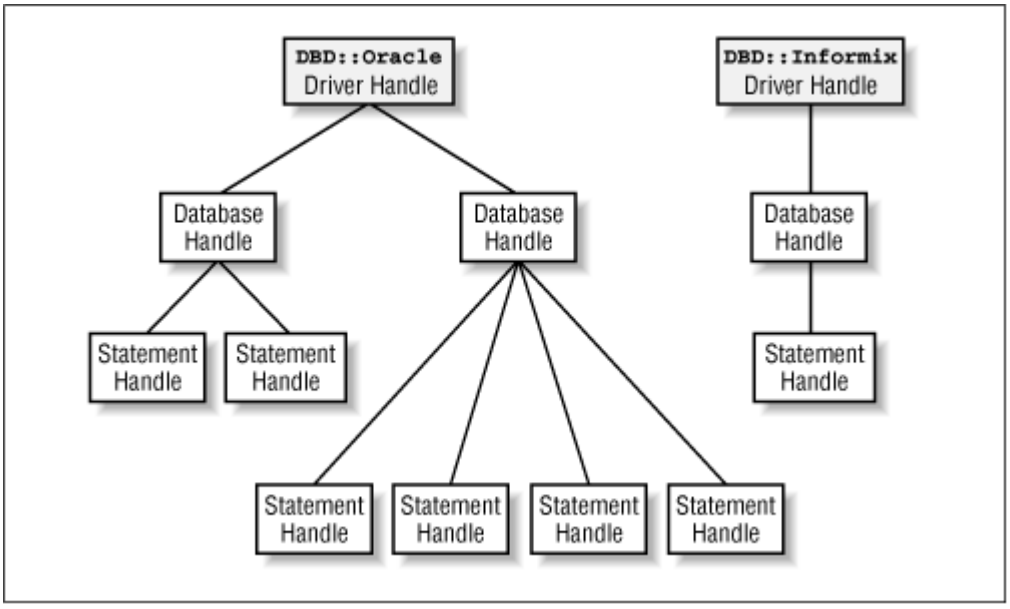
al inicio de los programas. Esta línea ubica y carga el módulo central de DBI. Los módulos individuales del controlador de base de datos se cargan según lo requerido, y no deben ser cargados generalmente explícitamente.

Manejadores:

El DBI define tres tipos principales de objetos que puede usarse para **interactuar con las base de datos**. Estos objetos son denominados ***Manejadores(handles).***

Hay manejadores para **controladores** (Drivers Handle), otros para **conexiones a base de datos** (Database Handle), y los que se crean para realizar **sentencias** (Statement Handle) individuales en la base de datos. La figura siguiente ilustra toda la estructura de la manera en que los manejadores están relacionados.

Fig. Manejadores DBI



Manejadores de Controladores:

Los manejadores de controladores representan los controladores que están cargados y se crean inicializan cuando los controladores son cargados en el DBI. Sólo debería haber exactamente uno por cada base de datos con la que se está trabajando. Inicialmente, el manejador de controlador es el único contacto que tiene DBI con el controlador, y hasta ese momento, no se ha hecho contacto con ninguna base de datos a través del controlador. Por convención se les suele denominar usando como nombre de variable **$drh.**

Los únicos dos métodos significativos disponibles a través de los manejadores de controladores son **data_sources()**, para enumerar con cuál controlador puede ser conectado, y **connect()**, que realiza una conexión. Estos métodos son más comúnmente invocados como métodos de clase DBI.

Manejadores de Base de Datos:

Los manejadores de base de datos son el primer paso para trabajar con la base de datos.
Por convención se les suele denominar usando como nombre de variable **$dbh.**

Manejadores de Sentencias:

Los manejadores de Sentencias son el tipo final de objetos que DBI define para interactuar y manipular bases de datos. Estos manejadores encapsulan una sentencia SQL individual para ser ejecutada en la base de datos.
Los manejadores de sentencias son hijos del correspondiente manejador de base de datos. Para cualquier base de datos no hay ningún límite de los manejadores de sentencias que se pueden crear y ejecutar. El número de manejadores que se pueden ejecutar a la vez (concurrentemente) depende del controlador de la base de datos (módulo DBD), por ejemplo ODBC sólo se puede ejecutar una sentencia a la vez, mientras que en MySQL se pueden ejecutar todas las que se deseen. Por convención se les suele denominar usando como nombre de variable **$sth**.

CONECTAR Y DESCONECTAR CON LA BASE DE DATOS USANDO DBI

Para trabajar con DBI lo primero (antes que nada) que hay que hacer es leer el módulo DBI:

```
use DBI;
```

Después y para cualquier tarea que se desee realizar hay que abrir una conexión con la base de datos, la información que se utiliza **para establecer la conexión** es:

- **Nombre de la fuente de datos**: Esto es una cadena que contiene los siguientes elementos, separados por dos puntos:
  - dbi
  - Nombre del driver: Nombre del driver, por ejemplo, odbc, mysql, oracle, etc.
  - Nombre de la base de datos: Nombre de la base de datos con la que estamos trabajando.
- **Usuario**: Usuario que se va a conectar a la base de datos. Dentro de la base de datos hay distintos usuarios, cada uno, con sus propias tablas y con una serie de privilegios asociados.
- **Password**: Clave de seguridad del usuario.
- **Host**(OPCIONAL): Aqui indicamos la máquina donde está la base de datos.
- **Puerto**(OPCIONAL): Aquí indicamos el puerto que se va a usar para conectarse a la base de datos.

Una vez que se ha terminado de trabajar con la base de datos se debe **cerrar la conexión** con la base de datos, no es estrictamente necesario, pero si aconsejable, ya que algunas bases de datos pueden no terminar sus transacciones correctamente.

Veamos un ejemplo de un script en Perl que conecta y desconecta a un sistema de bases de datos Access, cuya base de datos se llama *conexion*, el nombre del usuario es y el password no suministrados, en ocasiones se puede especificar.

```
#Usamos las librerias de acceso a BD
use DBI;

my $base_datos="conexion"; #Nombre de las base de datos
my $usuario=""; #Usuario de la BD
my $clave=""; #Password de la BD
my $driver="ODBC"; #Utilizamos el driver de ODBC

#Conectamos con la BD, si no podemos, ponemos un mensaje de
#error
```

```
my $dbh = DBI->connect("dbi:$driver:$base_datos",$usuario,$clave)
|| die "\nError al abrir la base datos: $DBI::errstr\n";

#Decimos que hemos creado la tabla
print "\nSe ha conectado con la BD $base_datos del driver
$driver\n";

#Nos desconectamos de la BD. Mostramos un mensaje si hay
#algun fallo
$dbh->disconnect || warn "\nFallo al desconectar.\nError:
$DBI::errstr\n";

#Terminamos
exit;
```

CONSULTAS EN LA BASE DE DATOS USANDO DBI

Las tareas más comunes entre un script en Perl y la base de datos es obtener datos de la misma. En SQL, este proceso se realiza con una sentencia Select.

El proceso de obtener datos usando DBI se divide en cuatro etapas:

1. **Preparación**: En esta etapa se analiza y valida la sentencia SQL y se devuelve un manejador de sentencia representando la sentencia en la base de datos. Se utiliza el método **prepare** (con la consulta SQL como parámetro) del manejador de la base de datos, devolviendo el manejador de la consulta.

2. **Ejecución**: En esta etapa se *ejecuta* el manejador de sentencia obtenido en la etapa anterior. En esta etapa se hace la consulta y se almacenan los datos en la estructuras de datos correspondientes de la base de datos, aunque en esta etapa, el script en Perl no puede acceder a los datos obtenidos. Se utiliza en método **execute** del manejador de sentencia.

3. **Extracción**: En esta etapa se *captan los datos* de la base de datos usando el manejador de sentencia. En dicha captación de datos se va almacenando los datos consultados, tupla a tupla en la estructuras de datos de Perl, para que sean posteriormente manipulados por el script.

4. **Liberación**: En esta fase se *liberan los recursos* ocupados por el manejador de sentencia y por la base de datos. Se utiliza el método **finish** del manejador de sentencia.

Un ejemplo en una base de datos donde se realiza una consulta para obtener información de la misma.

```
#Usamos la librerías de acceso a BD
use DBI;
my $base_datos=”conexion”; #Nombre de la base de datos
my $usuario=””; #Usuario de la BD
my $clave=””; #Password de la BD
my $driver=”ODBC”; #Utilizamos el controlador ODBC
my $tabla=”Cliente”; #Nombre de la tabla ejemplo

#Conectamos con la BD, si no podemos, ponemos un mensaje de
#error
my $dbh = DBI->(“dbi:$driver:$base_datos”,$usuario,$clave) ||
die “\nError al conectar a la base de datos”;

#Decimos que hemos conectado
print “\nSe ha conectado con la base de BD $base_datos del
driver $driver\n”;
#Realizamos la etapa de preparación de la sentencia
my $sth=$dbh->prepare(“Select Nombre, Apellido from
Cliente;”);
#Realizamos la etapa de ejecución de la sentencia
$sth->execute();

#Realizamos la etapa de extracción de datos. Imprimimos tupla a tupla.
While(@tupla=$sth->fetchrow_array()){
     Print “Nombre: $tupla[0]\t Apellido: $tupla[1]\n”;
}
#Realizamos la etapa de liberación de recursos ocupados por
la sentencia
$sth->finish();
```

EJECUCIÓN DE SENTENCIAS NO SELECT USANDO DBI

Hay operaciones que se realizan en la base de datos que al no ser consultas (sentencias SELECT) no necesitan atravesar las cuatro etapas que se han visto anteriormente: Preparación->Ejecución->Extracción->Liberación. Por la simple razón de que no existe la etapa de extracción de datos.

En este tipo de sentencias es un poco dilatado llamar a los métodos **prepare, execute y** finish, por lo que DBI define el método **do** para el manejador de la base de datos. Este método devuelve **false** sólo si hay un error.

Un ejemplo donde se crea una tabla y se inserta una tupla en la misma:

```
#Usamos la librerías de acceso a BD
use DBI;
my $base_datos=”conexion”; #Nombre de la base de datos
my $usuario=””; #Usuario de la BD
my $clave=””; #Password de la BD
my $driver=”ODBC”; #Utilizamos el controlador ODBC
my $tabla=”articulos”; #Nombre de la tabla ejemplo
my $campos_tabla=”id INT PRIMARY KEY, nombre VARCHAR(250) NOT
NULL, PRECIO float”;

#Conectamos con la BD, si no podemos, ponemos un mensaje de
#error
my $dbh = DBI->(“dbi:$driver:$base_datos”,$usuario,$clave) ||
die “\nError al conectar a la base de datos”;

#Creamos la tabla
my $resultado=$dbh->do(“create table $tabla ($campos_tabla;)”
|| die “\nError en crear la tabla\n”);
```

```
#Decimos que la hemos creado.
print “\nSe ha creado la tabla $tabla en la BD $base_datos
del driver $driver\n”;

#Insertamos datos en la BD
print “\nInsertamos datos en la BD\n”;
$resultado=$dbh->do(“insert into $tabla values (100,
‘Tomates’,6.5);”)|| warn “Error al insertar”;
$resultado=$dbh->do(“insert into $tabla values (120,
‘Melocontones’,4.5);”)|| warn “Error al insertar”;
$resultado=$dbh->do(“insert into $tabla values (145,
‘Pan’,1.5);”)|| warn “Error al insertar”;
$resultado=$dbh->do(“insert into $tabla values (980,
‘Naranjas’,7.0);”)|| warn “Error al insertar”;

#Nos desconectamos de la BD. Mostramos un mensaje si hay
#algun fallo.
$dbh->disconnect || Warn “\nFallo al desconectar.\nErrror:
$DBI::errstr\n”;

#Terminamos
exit;
```

VENTAJAS DE CGIs

1. Entre las ventajas de la programación CGI, se tiene su sencillez, ya que es muy fácil de entender, además de ser un lenguaje de programación independiente, ya que los escritos CGI pueden elaborarse en varios lenguajes. Para que el programa pueda ser ejecutado en el servidor, debe ser compilado para el sistema operativo del servidor, o en el servidor debe existir un intérprete que pueda ejecutar el programa. Si por ejemplo el servidor tiene un entorno UNIX, entonces ejecuta programas C que con ayuda de un compilador C para UNIX son compilados en un archivo ejecutable. Si el entorno es Windows NT, entonces los Scripts CGI pueden ser también archivos EXE que fueron creados con un compilador de 32 Bit para C, Pascal, Visual Basic, etc. Hoy en día la mayoría de los programas CGI son escritos en el lenguaje de la Shell de Unix o en Perl, para el cual debe estar instalado un determinado intérprete.

2. También es un estándar para usarse en todos los servidores Web, y funcionar bajo una arquitectura independiente, ya que ha sido creado para trabajar con cualquier arquitectura de servidor Web.

3. Como la aplicación CGI se encuentra funcionando de forma independiente, no pone en peligro al servidor, en cuanto al cumplimiento de todas las tareas que éste se encuentre realizando, o al acceso del estado interno del mismo.

DESVENTAJAS DE CGIs

Aunque este método es conceptualmente sencillo, presenta graves inconvenientes. Sus principales desventajas son:

1. Resulta difícil mantener las páginas web que se le devuelven al cliente, cuyas instrucciones HTML se encuentran insertadas en el propio código del programa CGI, mezclándose de esta manera sus funcionalidades.

2. El CGI presenta cierta desventaja en su eficiencia, debido a que el Servidor Web tiene que cargar el programa CGI (proceso de carga) en memoria cada vez que un usuario requiere su ejecución por medio de la página web que lo invoca, y conectar y desconectar con la base de datos cada vez que se recibe una requisición. La existencia de múltiples clientes simultáneos supone múltiples copias del programa en memoria del servidor (con el coste añadido de todas estas cargas), este funcionamiento resulta obsoleto y además compromete el funcionamiento de las máquinas que soportan servidores web muy utilizados. Además, no existe un registro del estado del servidor, sino que todo hay que hacerlo manualmente.

3. Ligado a situaciones de riesgos. Cuando los usuarios envían un formulario o invocan un CGI de alguna otra forma, en definitiva se les está permitiendo ejecutar remotamente un programa en el servidor. Es más, puesto que la mayoría de CGI's aceptan datos de la entrada de usuario (bien después de rellenar un formulario o directamente desde la línea de URL), en esencia se les brinda a los usuarios la oportunidad de controlar cómo se ejecutará el CGI, de manera que podrían intentar la introducción de una serie de parámetros inesperados hábilmente manipulados para que el CGI funcionase maliciosamente.

4. Referente a vulnerabilidad. El punto vulnerable de la programación en CGI, es decir, la amenaza que representan para la seguridad del servidor, es doble:

- Por un lado, la posibilidad de que el CGI sea engañado por la entrada del usuario para que ejecute comandos imprevistos, pudiendo llegar a causar graves daños en el servidor.

- De otro, la posibilidad de revelar innecesariamente información acerca del servidor, que permitirá al atacante conocer mejor la configuración del sistema y estar así mejor equipado para buscar posibles agujeros por los que colarse.

6. INSTALACIÓN DE PERL (INTERPRETE) Y CONFIGURACIÓN DE IIS PARA EJECUCIÓN DE SCRIPTS CGI

6.1. Instalación y configuración de Perl.

6.1.1. Instalación de Perl

Para la instalación del intérprete gratuito de Perl utilizado para la ejecución de scrips cgi se procedió a descargar el mismo desde el sitio web de ActiveState, empresa que desarrolla un intérprete de Perl win32, la versión descargada dicho intérprete es la 5.6.0.

Luego de descargarse se procedió a su instalación, y el procedimiento ese el siguientes:

Procedimiento:

1. Ejecutar instalador del intérprete de Perl de ActiveState descargado.
2. Aceptar los términos de Licencia.
3. Seleccionar la unidad de disco y/o carpeta de destino de Intérprete (X:\Perl), donde X es la unidad de disco duro.
4. Activar casillas de verificación:
    a. Add Perl to the Path enviroment variable(Añadir Perl a la ruta de la variable de ambiente).
    b. Create Perl file extension association (Crear la asociación de las extensión de archivo Perl).
5. Presionar el botón Finish (Finalizar).

6.1.1.1.- Instalación de los módulos DBI, CGI

Ahora corresponde instalar los módulos correspondientes para DBI y CGI, esto se realiza usando el PPM (Perl Packet Manager – Manejador de Módulos Perl).

¿Qué es el PPM?[12]

El Manejador de módulos Perl (PPM) provee una interfaz de línea de comandos para manejar los módulos y extensiones Perl. PPM permite acceder a los depósitos de los módulos, instalar y quitar módulos del sistema, y actualizar los módulos, que previamente fueron instalados usando PPM, con las últimas versiones.

¿Cómo Instalar PPM?

PPM se instala cuando se realiza la instalación de la distribución de Perl de ActiveState.

¿Cómo ejecutar PPM?

Antes de ejecutar PPM, es necesario estar conectado a Internet. Si la conexión a Internet es vía Firewall o Proxy, debe configurarse la variable de ambiente “HTTP_proxy” con el nombre del servidor proxy que da acceso. Si el servidor proxy requiere Nombre de usuario y contraseña, las variables de ambiente “HTTP_proxy_user” y “HTTP_user_pass” deben ser establecidas con los respectivos valores.

[12] Ver la documentación de ActivePerl de ActiveState o visitar el sitio de ActiveState.com para saber más sobre PPM y sus comandos.

Instalando los módulos

Procedimiento:

1. Desde la interfaz de comando teclear PPM
2. Se visualiza el prompt de PPM: **ppm>**
3. Teclear “search” para obtener una lista de todos los módulos disponibles, o teclear “search [patrón]”, donde patrón, es un parámetro de búsqueda de un módulo. Ejemplo:
   ppm>search ODBC|DBI.
   Para el caso actual:
   ppm>search CGI
   ppm> search DBI
4. Encontrado el modulo correspondiente se procede a instalarlo tecleando “install <nombre_módulo>, donde “nombre_módulo” es el nombre del módulo a instalar. Para el caso actual:
   ppm>install CGI
   ppm>install DBI

   Nota: Se puede omitir el paso 3 en caso de conocer el módulo a instalar y sabiendo que existe en el proveedor, en este caso ActiveState.com

6.1.2 **Configuración del intérprete de Perl**

Para que sea posible la ejecución de scripts cgi, creados en lenguaje perl, por el servidor web IIS, se requiere asociar dicho intérprete con archivos de extensión *.pl*(extensión usada comúnmente por los scripts cgi en un sistema Windows) en el registro de Windows. Esto se hace en la clave ScriptMap del registro de Windows, y se detalla a continuación.

Procedimiento:

1. Detener el servidor web y ejecutar el Regedit32.exe o Regedit.exe y abrir **HJEY_LOCAL_MACHINE** y seguir la secuencia:

   SYSTEM\CurrentControlSet\Services\W3SVC\Parameter\ScriptMap

2. En el menú Edición, elegir Nuevo y seleccionar como tipo Valor Alfanumérico.
3. Especificar la extensión que poseerán los escritos cgi, para el caso *.pl*.
4. Dar doble clic sobre el valor creado y en Editar cadena, digitar la ruta completa del intérprete a usar con los escritos *.pl* (Nuestro caso, algo semejante a: `X:\Perl\bin\perl.exe %s %s`. El primer %s es la ruta de acceso traducida del archivo de comandos de PERL - la dirección URL traducida a una ruta de acceso local-; el segundo %s es la cadena de consultas (información de la dirección URL) y sólo se pasa como un parámetro de la línea de comandos si la cadena de consultas no contiene el signo igual (=)).
5. Volver a iniciar el Servidor Web.

7. DESARROLLO DE APLICACIONES.

7.1. CREACIÓN DE LA BASE DE DATOS ACCESS (BASEDATOSWEB.MDB)

A continuación se muestra el diseño de la base de datos relacional creada en Access 97. Esta base de datos contiene 5 tablas relacionadas que pretende servir de ejemplo para esta caso, asemejando la base de datos de una empresa (ficticia) de servicios de asesoría y consultoría que le permitirá **registrar** sus clientes, **modificar**, **borrar** y **buscar** datos correspondientes a los mismos. Las tablas y sus relaciones se muestra seguidamente:

Tabla Clientes

Nombre del campo	Tipo de datos (Longitud)	Descripción
IDCliente	Texto (10)	identificador del cliente (Clave primaria), requerido.
Password	Texto (10)	Clave de acceso
Nombre	Texto (20)	Nombre del cliente, requerido.
Apellidos	Texto (20)	Apellidos del cliente, requerido.
Fecha_nac	Fecha/Hora (Fecha corta)	Fecha de nacimiento del cliente.
Domicilio	Texto (100)	Dirección de contacto del cliente, no requerido.
Ciudad	Texto (50)	
Pais	Texto (20)	
Email	Texto (50)	
Tipo_Negocio	Texto (10)	Negocio que posee el cliente, requerido.
Volumen_Negocio	Texto (10)	Volumen del negocio, no requerido.
Telef	Texto (15)	Teléfono del cliente, no requerido.

Tabla Servicios

Nombre del campo	Tipo de datos (Longitud)	Descripción
IDServicio	Autonumérico	Identificador del servicio (Clave primaria), requerido.
CategoriaID	Texto(10)	Categoría a la que pertenece el servicio.
Nombre	Texto (50)	Nombre del proyecto, no requerido.
Descripcion	Texto (200)	Descripción del proyecto.
Precio	Moneda	Costo del servicio.

Tabla OrdenServicio

Nombre del campo	Tipo de datos (Longitud)	Descripción
IDOrden	Autonumérico	No. de orden (clave primaria), requerido.
IDCliente	Texto (10)	
Factura_orden	Fecha/hora (Fecha corta)	
Total	Moneda	Total de la orden de los servicios.

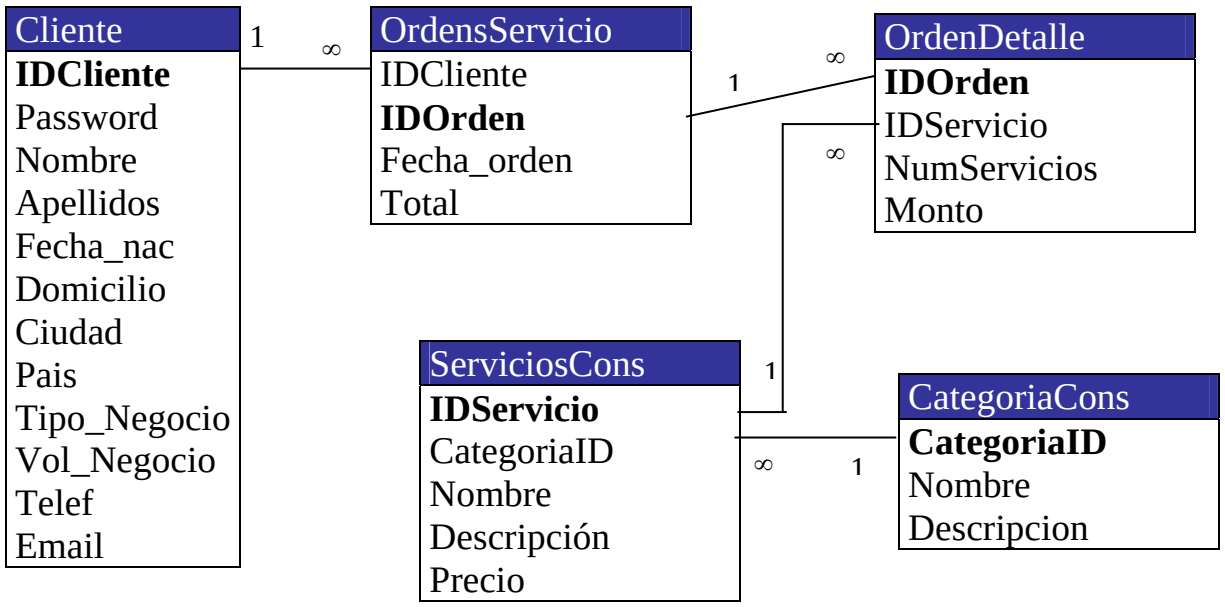
Tabla OrdenDetalle

Nombre del campo	Tipo de datos (Longitud)	Descripción
Id_factura	Texto (10)	
Id_proyecto	Texto (10)	
Monto	Numérico (Doble)	

Tabla CategoriaCons

Nombre del campo	Tipo de datos (Longitud)	Descripción
CategoriaID	Texto (10)	Llave primaria.
Nombre	Texto (50)	Nombre de la categoría.
Descripcion	Texto (50)	Descripción de la categoría.

Relaciones



7.2. CREACIÓN DE DSN

Para acceder a nuestra base de datos a través de una interfaz Web/DBMS utilizando ASP o DBI en scripts CGI Perl es necesario tener un vínculo con nuestra base de datos, esto se realiza a través de los controladores. Los controladores de base de datos (programas que pasan información desde la aplicación Web a una base de datos) utilizan un Nombre de origen de datos (DSN) para encontrar e identificar una base de datos ODBC en particular.
Normalmente, el DSN contiene información de configuración de la base de datos, seguridad de usuarios y ubicación, y puede tener la forma de una entrada en el registro del sistema de Windows NT o de un archivo de texto.

Con ODBC se puede elegir el tipo de DSN a crear: **Usuario**, **Sistema** o **Archivo**.

Los DSN de **Usuario** y de **Sistema** residen en el registro del sistema de Windows NT. Los DSN de **Sistema** permiten que todos los usuarios que han iniciado una sesión en un servidor concreto tengan acceso a una base de datos, mientras que los DSN de **Usuario** limitan la conectividad con la base de datos a los usuarios que dispongan de las credenciales de seguridad apropiadas. Los DSN de **Archivo**, que tiene la forma de archivos de texto, proporcionan acceso a varios usuarios y son fácilmente transferibles entre un servidor y otro mediante la copia de los archivos DSN.

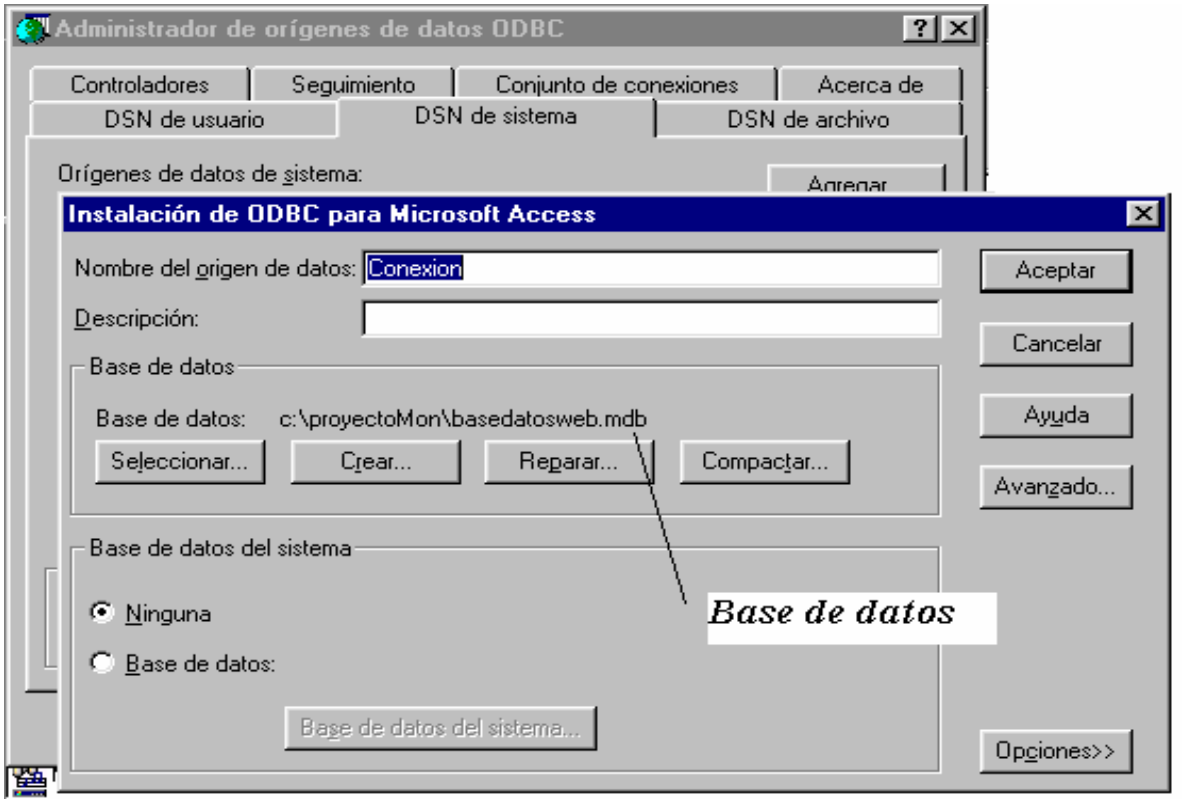
Como el DSN de Sistema permite que el usuario anónimo que crea IIS al instalarse en nuestro servidor puedan acceder a nuestra base de datos, dispongo entonces a crear un DSN de Sistema.

Pasos para la creación de DSN.

Para crear un DSN de sistema, debe abrirse el **Panel de Control** desde el menú **Inicio** de Windows. Luego hacer clic en el icono **ODBC** y seleccionar la hoja de propiedades **DSN de Sistema**. Hacer clic en **Agregar**, elegir el controlador de la base de datos (para este caso **Microsoft Access Driver**) y hacer clic en **Finalizar**.

Posteriormente hay que escribir el nombre del DSN de Sistema, en cual he llamado **Conexión**, y finalmente se selecciona la base de datos a utilizar, para este caso llamada **basedatosweb.mdb**, y dar clic en el botón **Aceptar**.

Gráficamente, es de la siguiente manera:



Dando clic en el botón **Avanzado...** se le puede agregar un nombre y contraseña de inicio de sesión para dicha origen de base de datos.

7.3. DESARROLLO DE UNA APLICACIÓN ASP

A continuación detallo la programación que corresponde para ASP que permite inicialmente registrar a un usuario, modificar sus datos o actualizarlos, borrarlos y hacer búsqueda del mismo. Para ello se debe realizar una conexión a la base de datos "basedatosweb" y esta podría realizarse de las siguientes maneras:

a. Conexión a una base de datos Access sin declarar DSN

```
'Cadena de conexión
strconn = "DRIVER={Microsoft Access Driver (*.mdb)};DBQ="
& Server.MapPath("\dirBD\aspbd.mdb") 'En una sola línea
'Objeto Connection
set conn = Server.Createobject("adodb.connection")
'Realizar conexión
conn.open strconn
```

b. Conexión a una base de datos ACCESS sin declarar DSN usado Jet

```
'Cadena de conexión.
strconn = ""Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
& Server.MapPath("/dirDb/asptutor.mdb")
'Objeto Connection
set conn = Server.Createobject("adodb.connection")
'Realizar conexión
conn.open strconn
```

c. Conexión a una base de datos con DSN

```
'Objeto Connection
set conn = server.createobject("adodb.connection")
'Realizar conexión
conn.open "asptutor"
```

d. Declaración de un recordset con DSN y con conexión implícita (es decir, sin declarar el objeto connection)

```
set rs = server.createobject("adodb.recordset") 'Objeto Connection.
rs.open "nombre_de_tabla", "Nombre_DSN" 'Realizar la conexión
```

El que utilizo comúnmente es el que corresponde a conectarse mediante DSN, por tal razón anteriormente se explica la manera de cómo crear un DSN de sistema. A continuación el código que corresponde a registrar, modificar, borrrar y buscar a un usuario.

Script: registrar.asp

Procedimiento "formulario"

```
<%private sub formulario %>
<form name="FormCl" method="post" action="registrar.asp"
onSubmit="return ValidarEntrada(this)">
  <table width="84%" border="0" align="center">
    <tr>
      <td colspan="3">
        <h2 align="center"><font color="#FFFFFF" face="Arial,
Helvetica, sans-serif"><b><font color="#003366">Registro
          del Cliente</font></b></font></h2>
        <hr>
      </td>
    </tr>
    <tr>
      <td colspan="3" background="images/bkgdhome.gif">
        <h2 align="left"><font face="Times New Roman, Times,
serif" color="#ffcc00"><b> Datos
          personales</b></font></h2>
      </td>
    </tr>
    <tr>
      <td colspan="3">
        <p align="left"><font face="Times New Roman, Times,
serif" size="3" color="#996633"><b>Los
          campos que aparecen con ast&eacute;riscos son datos
requeridos.</b></font></p>
      </td>
    </tr>
    <tr>
      <td width="27%" bgcolor="#DBEAF5" style="padding:2px;">
        <p align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Nombres</font><sup>*</sup>:</b></p>
      </td>
      <td width="26%">
        <div align="left">
          <input type="text" name="Nombre" size="20"
MAXLENGTH="30" value="<%=Request.form("Nombre")%>">
        </div></td>
```

./.

```
<td width="43%">
        <div align="left"></div>
      </td>
    </tr>
    <tr>
      <td width="27%" bgcolor="#DBEAF5" >
 <div align="left"><b><font face="Arial, Helvetica, sans-serif"
size="2">Apellidos</font><sup>*</sup>:</b></div>
      </td>
      <td width="26%">
        <div align="left">
         <input type="text" name="Apellido" size="20"
MAXLENGTH="30" value="<%=Request.form("Apellido")%>">
        </div>
      </td>
      <td width="43%">
        <div align="left"></div>
      </td>
    </tr>
    <tr>
      <td width="27%" bgcolor="#DBEAF5" >
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Domicilio</font><sup>*</sup><font face="Arial,
Helvetica, sans-serif" size="3">:</font></b></div>
      </td>
      <td width="26%">
        <div align="left">
          <input type="text" name="Domic" size="20"
maxlength="30" value="<%=Request.form("Domic")%>">
        </div></td>
      <td width="43%">
        <div align="left"></div></td></tr>
<tr>
      <td width="27%" bgcolor="#DBEAF5" >
        <div align="left"><font face="Arial, Helvetica, sans-
serif" size="2"><b>Ciudad<sup>*</sup>:</b></font>
</div>
      </td>
      <td width="26%">
        <div align="left">
  <input type="text" name="Ciudad" size="20" maxlength="25"
value="<%=Request.form("Ciudad")%>">
     </div>
     </td>
```

./.

```
   <td width="43%">
        <div align="left"></div>
      </td>
    </tr>
    <tr>
      <td width="27%" bgcolor="#DBEAF5" >
        <div align="left"><font size="2" face="Arial, Helvetica,
sans-serif"><b>Pa&iacute;s:</b></font></div>
      </td>
      <td width="26%">
        <div align="left">
          <select name="Pais" size="1">
            <option value="Nicaragua"
selected>Nicaragua</option>
            <option value="Belice">Belice</option>
            <option value="Guatemala">Guatemala</option>
            <option value="El Salvador">El Salvador</option>
            <option value="Honduras">Honduras</option>
            <option value="Costa Rica">Costa Rica</option>
            <option value="Panama">Panamá</option>
          </select>
        </div>
      </td>
      <td width="43%">
        <div align="left"></div>
      </td> </tr> <tr>
      <td width="27%" bgcolor="#DBEAF5" >
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Fecha
          de Nacimiento</font><sup>*</sup>:</b></div>
      </td><td width="26%"> <div align="left">
          <input type="text" name="F_nac" size="20"
maxlength="25" value="<%=Request.form("F_nac")%>">
        </div></td><td width="43%">
        <div align="left">(dd/mm/aaaa) </div>
      </td>
     </tr>
     <tr>
     <td width="27%" bgcolor="#DBEAF5">
        <div align="left">
     <b>
     <font face="Arial, Helvetica, sans-serif"
size="2">Tel&eacute;fono</font><font face="Arial, Helvetica,
sans-serif" size="3">:</font></b></div></td>
```

./.

```
<td width="26%">
        <div align="left">
  <input type="text" name="Telf" size="12" MAXLENGTH="14"
value=<%=Request.form("Telf")%>>
        </div>
      </td>
      <td width="43%">
        <div align="left"></div>
      </td>
    </tr>
    <tr>
      <td width="27%" bgcolor="#DBEAF5">
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">E-Mail<sup>*</sup>:</font></b></div>
      </td>
      <td width="26%">
        <div align="left">
          <input type="text" name="email" size="20"
maxlength="25" value="<%=Request.form("email")%>">
        </div></td>
      <td width="43%">
        <div align="left"></div></td></tr><tr>
      <td width="27%" bgcolor="#DBEAF5">
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Elija
          unLogin</font><sup>*</sup>:</b></div>
      </td>
      <td width="26%">
        <div align="left">
          <input type="text" name="Login" size="20"
maxlength="25">
        </div></td>
      <td width="43%" bgcolor="#0099CC"
background="images/bkgdhome.gif">
        <div align="left"><font color="#FFFFFF">El login debe
comenzar por una
          letra y sólo puede contener letras, números y el
carácter de subrayado
          (_) pero sin espacios.</font></div>
      </td></tr> <tr>
      <td width="27%" bgcolor="#DBEAF5">
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Password</font><sup>*</sup>:</b></div></td>
```

./.

```
<td width="26%"> <div align="left">
<input type="password" name="password" size="20" maxlength="25">
</div></td>
<td width="43%" bgcolor="#0099CC"
background="images/bkgdhome.gif">
<div align="left"><font color="#FFFFFF">El password debe tener
al menos seis carácteres de longitud y puede contener números y
letras, pero no puede incluir ningún espacio. Las contraseñas
distinguen entre mayúsculas y minúsculas.</font></div></td></tr>
<tr><td width="27%" bgcolor="#DBEAF5">
<div align="left"><b><font face="Arial, Helvetica, sans-serif"
size="2">Repetir Password</font><sup>*</sup>:</b></div></td><td
width="26%"> <div align="left">
<input type="password" name="Rpassword" size="20"
maxlength="25"></div></td><td width="43%">
        <div align="left"></div></td></tr>
    <tr><td width="27%" bgcolor="#DBEAF5">
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Tipo de Negocio<sup>*</sup>:</font></b>
</div></td><td width="26%"> <div align="left">
<input type="text" name="T_Nego" size="25" maxlength="30"
value="<%=Request.form("T_Nego")%>"></div></td>
      <td width="43%"><div align="left"></div></td></tr>
    <tr><td width="27%" height="5" bgcolor="#DBEAF5">
        <div align="left"><b><font face="Arial, Helvetica, sans-
serif" size="2">Volumen del Negocio:</font></b></div></td>
      <td width="26%" height="5"> <div align="left">
          <select name="V_Nego" size="1">
            <option selected>Ninguno</option>
            <option>Peque&ntilde;o</option>
            <option>Mediano</option>
            <option>Grande</option>
          </select></div></td>
      <td width="43%" height="5"><div
align="left"></div></td></tr>
    <tr><td width="27%"><p align="left">  </p></td>
      <td width="26%"><p align="left">  </p></td>
      <td width="43%"><p align="left"><font face="Arial,
Helvetica, sans-serif"></font></p></td></tr></table><blockquote>
<blockquote><p>
        <input type="submit" name="Submit"
value="Registrar"> 
        <input type="reset" name="Limpiar" value="Limpiar"></p>
    </blockquote></blockquote></form>
<%End Sub%>
```

./.

Rutinas JavaScript para validación de entradas del formulario

```
<html><head><script language="javascript">
    var whitespace = " \t\n\r";
   function isEmpty(s){  return ((s == null) || (s.length ==
0))}
//Función que chequea si se introduce una dirección de correo.
    function isEmail (s){
           if (isEmpty(s))
           // es s espacio en blanco?
           if (isWhitespace(s)) return false;
             if (CarsEnCol (s, "#!&$()?¿")) return false;
           // debe haber >= 1 carácteres antes @, así que
           // iniciamos buscando después de la posición 1
           // (segundo carácter)
           var i = 1;
           var sLength = s.length;

           // buscar @
           while ((i < sLength) && (s.charAt(i) != "@")) { i++ }
  if ((i >= sLength) || (s.charAt(i) != "@")) return false;
           else i += 2;
            // buscar "."
           while ((i < sLength) && (s.charAt(i) != ".")) { i++
}

           // Debe existir al menos un carácter después del "."
 if ((i >= sLength - 1) || (s.charAt(i) != ".")) return false;
           else return true;
        }
 //Función que busca si existe espacio en blanco.
    function isWhitespace (s){
       var i;
       // Es s vacía?
       if (isEmpty(s)) return true;
 // Buscar a través de la cadena de carácteres uno por uno hasta
encontrar un carácter //no espacio en blanco. Cuando lo
encontamos, retornamos false; sino, retornamos true.
        for (i = 0; i < s.length; i++){
 // Chequear que el carácter actual no es espacio en blanco.
            var c = s.charAt(i);
            if (whitespace.indexOf(c) == -1) return false;
        }
// Todos los carácteres son espacio en blanco.
         return true;
}
```

./.

```
//Función que busca caracteres en una colección de ellos.
  function CarsEnCol (s, col)
  {
    var i;
    // Buscar a través de la cadena de carácteres uno por uno.
    // Si carácter está en colección, añadir a returnString.
    for (i = 0; i < s.length; i++)
    {
// Chequear que el carácter actual no es espacio en blanco.
        var c = s.charAt(i);
        if (col.indexOf(c) == -1) return false;
    }
    return true;
  }

//Función para unir cadenas.
function ConstStr(s, s1){
     if (s.length > 0) s = s + ", ";
     s = s + s1;
     return s;
  }
function ValidarEntrada(form){
     var Q = "", S = "", O = "";
     var login = form.Login.value =
form.Login.value.toLowerCase();
 // Chequear todos los campos de información requerida.
  if (isWhitespace(form.Nombre.value)) Q = "Nombres";
  if (isWhitespace(form.Apellido.value)) Q = ConstStr( Q,
"Apellidos");
if (isWhitespace(form.Domic.value)) Q = ConstStr( Q,
"Domicilio");
if (isWhitespace(form.Ciudad.value)) Q = ConstStr( Q, "Ciudad");
if ( Q.length > 0 )
     { alert("Por favor provea de valores válidos para " + Q );
           return false;
     }//fi
//Validar la entrada de los campos Nombre y Apellidos.
if ( !CarsEnCol(form.Nombre.value,
"aábcdeéfghiíjklmnoópqrstuúvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ " ))
    {
      alert( "El Nombre puede contener sólo los siguientes
carácteres (mayúsculas o
minúsculas):\nabcdefghijklmnopqrstuvwxyz\ny espacio en blanco."
);
      return false;
 }//fi
```

./.

```
if ( !CarsEnCol(form.Apellido.value,
"aábcdeéfghiíjklmnoópqrstuúvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ " ))
    {      alert( "El Apellido puede contener sólo los siguientes
carácteres (mayúsculas o minúsculas):\nabcdefghijklmnopqrstuvwxyz\ny
espacio en blanco." );
      return false;
    }//fi
if(ValidarFecha(form.F_nac.value)==false) {return false;}
//Validar entrada del teléfono en caso de introducir.
 if ( !isWhitespace(form.Telf.value) && !CarsEnCol(form.Telf.value,
"0123456789-()" ))
    {
 alert( "El Teléfono puede contener sólo los siguientes carácteres
continuos:\n0123456789-()\nPor ejemplo:(505)-3114867" );      return
false;
    }//fi
else if(form.Telf.value.length < 9 && form.Telf.value.charAt(0)== "("
&& form.Telf.value.charAt(4)==")" ) { alert("El Teléfono debe ser un
valor válido.\nPor Ejemplo, para Nicaragua: (505)-2783654");
return false;}
else if(isWhitespace(form.Telf.value)) form.Telf.value = "NULL";

//Validar Email.
if (isEmail(form.email.value)==false){alert("Correo Electrónico
incorrecto!\nDebe parecer a:username@dominio.com"); return false;}
//Validar login.
    if ( !CarsEnCol( login, "abcdefghijklmnopqrstuvwxyz0123456789._"
))
    {
      alert( "El Login puede contener sólo los siguientes
carácteres:\nabcdefghijklmnopqrstuvwxyz0123456789.-_\n\nPor
ejemplo:smolina_98" );
      return false;
    }//fi
    else if ( !CarsEnCol( login.charAt(login.length - 1),
"abcdefghijklmnopqrstuvwxyz0123456789") ) {
  alert( "El Login debe finalizar en un carácter alfanumérico.");
      return false;
    }//else fi
    else if ( !CarsEnCol( login.charAt(0),
"abcdefghijklmnopqrstuvwxyz0123456789") ){
      alert( "El Login debe iniciar con un carácter alfanumérico.");
      return false;
    }//else fi
    else if ( login.length < 5 && login.length > 10 ) {
      alert( "El Login debe ser entre 5 o 10 carácteres." );
      return false;
    }//else fi
```

./.

```
//Verificar password
 if (form.password.value.length < 6 ){
alert( "Password debe contener mínimo 6 carácteres.");  return false;
    }//fi
    else if (form.password.value != form.Rpassword.value ){
      alert( "Sus passwords no coinciden. Por favor reescribalo y
pruebe de nuevo." );
      return false;
    }//else fi
Q = "";
if(isWhitespace(form.T_Nego.value)) Q = "Tipo de Negocio";
if ( Q.length > 0 ){   alert("Por favor provea de valores válidos
para " + Q );
     return false;
}//fi
return true;
}
// Valida Fecha. El Formato es dd/mm/aaaa
function ValidarFecha(Cadena){
      var Fecha= new String(Cadena);  // Crea un string
      var RealFecha= new Date(); // Para sacar la fecha de hoy
      // Cadena Año
         var Ano= new
String(Fecha.substring(Fecha.lastIndexOf("/")+1,Fecha.length));
      // Cadena Mes
        var Mes= new
String(Fecha.substring(Fecha.indexOf("/")+1,Fecha.lastIndexOf("/")));
      // Cadena Día
         var Dia= new String(Fecha.substring(0,Fecha.indexOf("/")));
      // Valido el año
if(isNaN(Ano)||Ano.length<4||parseFloat(Ano)<1900){alert('Año
inválido');return false;
      }
      // Valido el Mes
if(isNaN(Mes)||Mes.length<1||parseFloat(Mes)<1||parseFloat(Mes)>12){
           alert('Mes inválido');
           return false;
      }
      // Valido el Dia
         if (isNaN(Dia) || Dia.length<1 || parseInt(Dia)<1 ||
parseInt(Dia)>31){
           alert('Día inválido');
           return false;
      }
      if (Mes==4 || Mes==6 || Mes==9 || Mes==11 || Mes==2) {
           if (Mes==2 && Dia > 28 || Dia>30) {alert('Día inválido');
      return false:
           }
      }
} </Script></head>
```

./.

```
<title>Registro - Cliente</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1"><body bgcolor="#ffffff" text="#000000">
<table width="68%" border="0" align="center">
  <tr>
    <td colspan="2">
      <div align="left"><img src="images/ibdweb.gif" width="516"
height="50" alt="Integraci&oacute;n de Bases de Datos en el
Web"></div>
    </td></tr></table>
<hr>
<!--#INCLUDE File="ADOVBS.inc"-->
<%IF(Request.form<>"") THEN
                Set Objeto_Conn =
Server.CreateObject("ADODB.Connection")
           Objeto_Conn.Open "Conexion"
                Set Session("bdweb_conn") = Objeto_Conn

        sqlText ="Select * from Cliente where Ucase(IDCliente)='" &
Ucase(Request.form("Login")) & "'"

        'Crear objeto recordset
        Set Ob_RS = Server.CreateObject("ADODB.Recordset")
        'Abrir el Recordset
        Ob_RS.Open sqlText, Objeto_Conn, 2,3,1

        if Ob_RS.bof and Ob_RS.eof then 'Puede agregarse nuevo
registro
                'Agregar nuevo registro al recordset
                Ob_RS.AddNew
                Ob_RS("IDCliente") = Cstr(Request.form("Login"))
                Ob_RS("Password") = Cstr(Request.form("password"))
                Ob_RS("Nombre") = Request.Form("Nombre")
         Ob_RS("Apellidos") = Request.Form("Apellido")

         Ob_RS("Fecha_Nac") = Request.Form("F_Nac")
         Ob_RS("Domicilio") = Request.Form("Domic")
         Ob_RS("Ciudad") = Request.form("Ciudad")
         Ob_RS("Pais") = Request.form("Pais")
         Ob_RS("Email") = Request.form("email")
         Ob_RS("Tipo_Negocio") = Request.Form("T_Nego")
         Ob_RS("Volumen_Negocio") = Request.form("V_Nego")
         Ob_RS("Telef") = Request.form("Telf")
                Ob_RS.Update
Response.write "<br><br><H3>" & "Sus datos han sido insertados!." &
"</H3><p>"
Response.write "<A HREF=../index.asp>Inicio</A> "
Response.write "<A HREF=bdweb.asp>P gina de Ejemplos</A> "
Response.write "<A HREF=registrar.asp>Agregar otro Cliente</A>"
```

./.

```
else 'Error: Registro existente, el usuario intenta ingresar
dato existente.
Response.write "<br><br><H3><font face=Arial, Helvetica, sans-
serif color=red>" & "El login con que intenta registrarse ya
existe. Por favor seleccione otro." & "</font></H3>"
%>
<% call formulario%>
<%end if
   Ob_RS.Close
   Objeto_Conn.Close
   %>
<%ELSE call formulario
END IF %>
<p> </p>
</body>
</html>
```

Fin del Script Asp: **registrar.asp**

Script Asp: **modificar.asp**

```
<html><head>
<script language="javascript">
    var whitespace = " \t\n\r";
//Función que comprueba si es vacío.
function isEmpty(s){  return ((s == null) || (s.length == 0))}
    function isEmail (s){
           if (isEmpty(s))
           // es s espacio en blanco?
           if (isWhitespace(s)) return false;
             if (CarsEnCol (s, "#!&$()?¿")) return false;
           // debe haber >= 1 carácteres antes @, así que
           // iniciamos buscando después de la posición 1
           // (segundo carácter)
           var i = 1;
           var sLength = s.length;
           // buscar @
           while ((i < sLength) && (s.charAt(i) != "@")) { i++ }
      if ((i >= sLength) || (s.charAt(i) != "@")) return false;
           else i += 2;
           // buscar "."
           while ((i < sLength) && (s.charAt(i) != ".")) { i++
}
           // Debe existir al menos un carácter después del "."
           if ((i >= sLength - 1) || (s.charAt(i) != "."))
return false;
           else return true;
        }
//Función para unir cadenas.
     function ConstStr(s, s1){
          if (s.length > 0) s = s + ", ";
          s = s + s1;
          return s;
     }
     function CarsEnCol (s, col){
          var i;
 // Buscar a través de la cadena de carácteres uno por uno.
// Si carácter está en colección, añadir a returnString.
         for (i = 0; i < s.length; i++){
 // Chequear que el carácter actual no es espacio en blanco.
         var c = s.charAt(i);
          if (col.indexOf(c) == -1) return false;
         }
     return true;
```

```
//Función que busca si existe espacio en blanco.
    function isWhitespace (s){
       var i;
       // Es s vacía?
       if (isEmpty(s)) return true;
// Buscar a través de la cadena de carácteres uno por uno hasta
encontrar un carácter no espacio //en blanco. Cuando lo encontamos,
retornamos false; sino, retornamos true.
        for (i = 0; i < s.length; i++){
      // Chequear que el carácter actual no es espacio en blanco.
            var c = s.charAt(i);
            if (whitespace.indexOf(c) == -1) return false;
        }
        // Todos los carácteres son espacio en blanco.
         return true;
     }
     function ValidarDatos(form){
          var Q = "";
          var login = form.Login.value =
form.Login.value.toLowerCase();
// Chequear todos los campos de información requerida.
if (isWhitespace(login)) Q = "Login";
if (isWhitespace(form.Password.value)) Q = ConstStr( Q, "Password");
          if ( Q.length > 0 )  {
           alert("Por favor provea de valores válidos para " + Q );
           return false;
          }//fi
          //Validar login.
          if ( !CarsEnCol( login,
"abcdefghijklmnopqrstuvwxyz0123456789._" )){
alert( "El Login puede contener sólo los siguientes
carácteres:\nabcdefghijklmnopqrstuvwxyz0123456789.-_\n\nPor
ejemplo:smolina_98" );
                return false;
         }//fi
          else if ( !CarsEnCol( login.charAt(login.length - 1),
"abcdefghijklmnopqrstuvwxyz0123456789") ) {
alert( "El Login debe finalizar en un carácter alfanumérico.");
                return false;
          }//else fi
  else if ( !CarsEnCol( login.charAt(0),
"abcdefghijklmnopqrstuvwxyz0123456789") ){
alert( "El Login debe iniciar con un carácter alfanumérico.");
               return false;
}//else fi
else if ( login.length < 5 && login.length >10) {
  alert( "El Login debe ser entre 5 o 10 carácteres." );
               return false;
}//else fi
```

./.

```
//Verificar password
 if (form.Password.value.length < 6 ){alert( "Password debe contener
mínimo 6 carácteres.");
    return false;
}//fi
return true;
}

function ValidarActual(form){
     var Q = "", S = "", O = "";
   // Chequear todos los campos de información requerida.
  if (isWhitespace(form.Nombre.value)) Q = "Nombres";
  if (isWhitespace(form.Apellidos.value)) Q = ConstStr( Q,
"Apellidos");
  if (isWhitespace(form.Domicilio.value)) Q = ConstStr( Q,
"Domicilio");
  if (isWhitespace(form.Ciudad.value)) Q = ConstStr( Q, "Ciudad");
  if ( Q.length > 0 )
     {
           alert("Por favor provea de valores válidos para " + Q );
           return false;
     }//fi

//Validar la entrada de los campos Nombre y Apellidos.
if ( !CarsEnCol(form.Nombre.value,
"aábcdeéfghiíjklmnoópqrstuúvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ " ))
    {
      alert( "El Nombre puede contener sólo los siguientes carácteres
(mayúsculas o minúsculas):\nabcdefghijklmnopqrstuvwxyz\ny espacio en
blanco." );
      return false;
    }//fi
if ( !CarsEnCol(form.Apellidos.value,
"aábcdeéfghiíjklmnoópqrstuúvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ " ))
    { alert( "El Apellido puede contener sólo los siguientes
carácteres (mayúsculas o minúsculas):\nabcdefghijklmnopqrstuvwxyz\ny
espacio en blanco." );
      return false;
    }//fi
if(ValidarFecha(form.Fecha_nac.value)==false) {return false;}
//Validar entrada del teléfono en caso de introducir.
 if ( !isWhitespace(form.Telef.value) && !CarsEnCol(form.Telef.value,
"0123456789-()" ))
    { alert( "El Teléfono puede contener sólo los siguientes
carácteres continuos:\n0123456789-()\nPor ejemplo:(505)-3114867" );
      return false;
}//fi
```

./.

```
else if(form.Telef.value.length < 9 && form.Telef.value.charAt(0)==
"(" && form.Telef.value.charAt(4)==")" ) {alert("El Teléfono debe ser
un valor válido.\nPor Ejemplo, para Nicaragua: (505)-2783654");
     return false;}
//Validar Email.
if (isEmail(form.Email.value)==false){alert("Correo Electrónico
incorrecto!\nDebe parecer a:username@dominio.com"); return false;}
     Q = "";
if(isWhitespace(form.Tipo_Negocio.value)) Q = "Tipo de Negocio";
    if ( Q.length > 0 )
     {
 alert("Por favor provea de valores válidos para " + Q );
     return false;
}//fi
if(isWhitespace(form.Telef.value)) form.Telf.value = "NULL";
return true;
}
function ValidarFecha(Cadena){
     var Fecha= new String(Cadena)   // Crea un string
     var RealFecha= new Date()  // Para sacar la fecha de hoy
     // Cadena Año
        var Ano= new
String(Fecha.substring(Fecha.lastIndexOf("/")+1,Fecha.length))
     // Cadena Mes
        var Mes= new
String(Fecha.substring(Fecha.indexOf("/")+1,Fecha.lastIndexOf("/")))
     // Cadena Día
        var Dia= new String(Fecha.substring(0,Fecha.indexOf("/")))
     // Valido el año
     if (isNaN(Ano) || Ano.length<4 || parseFloat(Ano)<1900){
          alert('Año inválido')
          return false
     }
     // Valido el Mes
        if (isNaN(Mes) || Mes.length<1 || parseFloat(Mes)<1 ||
parseFloat(Mes)>12){
          alert('Mes inválido')
          return false
     }
     // Valido el Dia
        if (isNaN(Dia) || Dia.length<1 || parseInt(Dia)<1 ||
parseInt(Dia)>31){
          alert('Día inválido');     return false;
     }
     if (Mes==4 || Mes==6 || Mes==9 || Mes==11 || Mes==2) {
          if (Mes==2 && Dia > 28 || Dia>30) {
               alert('Día inválido');     return false; } } }
</Script>
```

./.

```
</head>
  <title>Modificar Registro del Cliente</title>
    <body bgcolor="#FFFFFF">
     <table width="68%" border="0" align="center">
       <tr>
         <td colspan="2">
           <div align="left"><img src="images/ibdweb.gif"
width="522" height="50" alt="Integraci&oacute;n de Bases de
Datos en el Web"></div>
         </td>
       </tr>
     </table>
     <hr>
     <p>
     <!--#INCLUDE File="ADOVBS.inc"-->
     <%IF(Request.form<>"") THEN
       'Crear un Objeto Connection
        Set conBD=Server.CreateObject("ADODB.Connection")
        Set MiReg=Server.CreateObject("ADODB.Recordset")
       'Abrir una conexión
        conBD.Open "Conexion"
        if(Request.form("modificar"))<>"" then
          sqlupdate="Update Cliente set Nombre='" &
          Request.form("Nombre") & "', Apellidos='" _
          & Request.form("Apellidos") & "', Fecha_nac='" &
          Request.form("Fecha_nac") & "'," & _
          "Domicilio='" & Request.form("Domicilio") & "',
          Ciudad='" & Request.form("Ciudad") & "'," & _
          "Pais='" & Request.form("Pais") & "', Tipo_Negocio='"
          & Request.form("Tipo_Negocio") & "'," & _
          "Volumen_Negocio='" & Request.form("Volumen_Negocio")
          & "', Telef='" & Request.form("Telef") & "'," & _
          "Email='" & Request.form("Email") & "' where
          IDCliente='" & Request.form("IDCliente") & "';"
          conBD.Execute(sqlupdate)
          Response.write "<h3>Registro modificado</h3>"
                        Set MiReg=nothing
                        conBD.close
                        Set conBD=nothing %>
          <a href="../index.asp">Página de inicio</a> 
          <a href="bdweb.asp">Página de Ejemplos</a> 
          <a href="modificar.asp"> Otra modificación</a>
```

./.

```
<%else
     'Definir consulta de selección.
     sqlselect="Select * from Cliente where UCase(IDCliente)='"
     & UCase(Request.form("Login")) & "' and Ucase(Password)='"
     & Ucase(Request.form("Password")) & "'"
     'Ejecutar consulta
     MiReg.open sqlselect, conBD, 2,3,1
     if MiReg.eof then
          Response.write "<h3>No hay ningún usuario registrado
          con el Login " & request.form("Login") & "</h3>"
     else
          Response.write "<table border=1>"
          Response.write "<form action=modificar.asp method=post
          onsubmit=return(ValidarActual(this))>"
          do while not MiReg.eof
             for each parametro in MiReg.fields
             Response.write "<tr>"
             if parametro.name="IDCliente" then
               Response.write "<th>Login</th><td><font
               color=blue>"& MiReg("IDCliente") & "</font>"
          elseif parametro.name="Password" then
          Response.write "<th>Password</th><td>No disponible"
             elseif parametro.name="Fecha_nac" then
               Response.write "<th>Fecha de Nacimiento"
             elseif parametro.name="Tipo_Negocio" then
               Response.write "<th>Tipo de Negocio"
             elseif parametro.name="Volumen_Negocio" then
               Response.write "<th>Volumen del Negocio"
                       else
               Response.write "<th>" & parametro.name & "</th>"
             end if
        if parametro.name="IDCliente" or
parametro.name="Password" then
          if parametro.name="Password" then%>
            <td><input type=hidden name=<%=parametro.name%>></td>
         <%else%>
        <td><input type=hidden name=<%=parametro.name%>
        value="<%=MiReg(parametro.name)%>"></td>
     <%end if%>
     <% elseif parametro.name="Domicilio" or
parametro.name="Email" then %>
       <td><input size=30 name=<%=parametro.name%>
       value="<%=MiReg(parametro.name)%>"></td>
     <%else %> <td><input size=20 name=<%=parametro.name%>
     value="<%=MiReg(parametro.name)%>"></td>
     <%end if
```

./.

```
next
   Response.write "<tr><td><input type=submit name=modificar
value=Modificar></td></tr>"
   Response.write "</form>"
   MiReg.movenext
loop
   Response.write "</table><br>"
   end if
  end if %>
<%else%>
<table width="46%" border="0">
  <tr>
    <td width="10%" valign="bottom"> </td>
    <td background="images/bkgdhome.gif" width="90%"
valign="bottom"> <b><font color="#FFFFFF" face="Times New Roman,
Times, serif" size="5">Escriba
      el login y contraseña del usuario a
modificar</font></b></td>
  </tr>
</table>
<br>
<form method= "post" action="modificar.asp"
onsubmit="return(ValidarDatos)">
  <table width="33%" border="0">
    <tr>
      <td width="14%"> </td>
   <td width="35%" bgcolor="#DBEAF5"><font face="Arial,
Helvetica, sans-serif"><b>Login:
        </b></font></td>
      <td width="51%">
        <input name="Login" size=15 maxlength="15">
      </td></tr><tr>
      <td width="14%"> </td>
      <td width="35%" bgcolor="#DBEAF5"><font face="Arial,
Helvetica, sans-serif"><b>Password:</b></font></td>
      <td width="51%">
<input type="password" name="Password" size=15 maxlength="20">
</td></tr>
  </table> <blockquote>
      <INPUT TYPE="Submit" VALUE="Enviar Datos">  
      <INPUT TYPE="Reset" VALUE="Limpiar">
  </blockquote></form>
<%end if%>
    </body></html>
```

Fin del script asp: **modificar.asp**

Script Asp: **borrar.asp**

```
<html>
  <head>
  <script language="javascript">
    var whitespace = " \t\n\r";
    function isEmpty(s){  return ((s == null) || (s.length ==
0))}
     function ConstStr(s, s1){
          if (s.length > 0) s = s + ", ";
          s = s + s1;
          return s;
     }
     function CarsEnCol (s, col){
       var i;
       for (i = 0; i < s.length; i++){
       var c = s.charAt(i);
            if (col.indexOf(c) == -1) return false;
         }
     return true;
     }
    function isWhitespace (s){
       var i;
       if (isEmpty(s)) return true;
        for (i = 0; i < s.length; i++){
            var c = s.charAt(i);
            if (whitespace.indexOf(c) == -1) return false;
        }
         return true;
     }
     function ValidarDatos(form){
          var Q = "";
          var login = form.Login.value =
form.Login.value.toLowerCase();
if (isWhitespace(login)) Q = "Login";
if (isWhitespace(form.Password.value)) Q = ConstStr( Q,
"Password");
if ( Q.length > 0 ) {
     alert("Por favor provea de valores válidos para " + Q );
 return false;}//fi
```

./.

```
//Validar login.
if ( !CarsEnCol( login, "abcdefghijklmnopqrstuvwxyz0123456789._"
)){
alert( "El Login puede contener sólo los siguientes
carácteres:\nabcdefghijklmnopqrstuvwxyz0123456789.-_\n\nPor
ejemplo:smolina_98" );
     return false;
}//fi
else if ( !CarsEnCol( login.charAt(login.length - 1),
"abcdefghijklmnopqrstuvwxyz0123456789") ) {
alert( "El Login debe finalizar en un carácter alfanumérico.");
     return false;
}//else fi
else if ( !CarsEnCol( login.charAt(0),
"abcdefghijklmnopqrstuvwxyz0123456789") ){
  alert( "El Login debe iniciar con un carácter alfanumérico.");
              return false;
          }//else fi
     else if ( login.length < 5 && login.length >10) {
              alert( "El Login debe ser entre 5 o 10
carácteres." );
              return false;
          }//else fi
     //Verificar password
          if (form.Password.value.length < 6 ){
           alert( "Password debe contener mínimo 6 carácteres.");
              return false;
          }//fi
     return true;
}
</Script>
</head>
  <title>Borrar Registro del Cliente</title>
<body bgcolor="#FFFFFF">
<table width="68%" border="0" align="center"> <tr> <td
colspan="2">
      <div align="left"><img src="images/ibdweb.gif" width="522"
height="50" alt="Integraci&oacute;n de Bases de Datos en el
Web"></div>
    </td>
  </tr>
</table><hr align="left">
  <!--#INCLUDE File="ADOVBS.inc"-->
```

./.

```
  <%IF(Request.form<>"") THEN
       'Crear un Objeto Connection
        Set conBD=Server.CreateObject("ADODB.Connection")
        Set MiReg=Server.CreateObject("ADODB.Recordset")
       'Abrir una conexión
        conBD.Open "Conexion"
        'Definir consulta SQL.
        sqltext = "Select * from Cliente where
Ucase(IDCliente)='" & Ucase(request.form("Login")) & "' and
Ucase(Password)='" & Ucase(request.form("Password")) & "'"
        'Mireg.open "Cliente",con,adopendynamic,adcmdtable
        'Abrir el registro
        Mireg.open sqltext, conBD, 2,3,1

        'Comprobar si registro existe
        if Mireg.bof and Mireg.eof then
           response.write "<h3>Registro no encontrado</h3>"
        else
          Mireg.delete
          Mireg.close
          conBD.close
          response.write "<h3>Registro borrado</h3>"
        end if %>
  <a href="index.asp"> Página de inicio</a> <a
href="bdweb.asp"> P gina de Ejemplos</a>    <a
href="borrar.asp"> Otro borrado</a>
  <%else%>
<table width="32%" border="0">
  <tr valign="bottom">
    <td width="13%" align="center" valign="bottom"> </td>
    <td width="87%" background="images/bkgdhome.gif"
align="center" valign="bottom">
      <div align="left"><font size="5" face="Times New Roman,
Times, serif"><b><font color="#FFFFFF"> Darse de
baja</font></b></font></div></td></tr></table>
<form name="FormBorrar" method= "post" action="borrar.asp"
onsubmit="return(ValidarDatos(this))">
  <table width="33%" border="0"> <tr valign="bottom"><td
height="40" bgcolor="#ffffff" width="14%"> <div align="left">
<p></p></div>
      </td> <td height="40" bgcolor="#ffffff" colspan="2"><font
face="Arial, Helvetica, sans-serif" color="#996633"><b>ESCRIBA
EL LOGIN Y LA CLAVE DEL CLIENTE A BORRAR</b></font></td>
</tr>
```

./.

```
<tr>
      <td width="14%"> </td>
      <td width="41%" bgcolor="#DBEAF5"><font face="Arial,
Helvetica, sans-serif"><b>Login:
        </b></font></td>
      <td width="45%">
        <input name="Login" size=15 maxlength="15">
      </td>
    </tr>
    <tr>
      <td width="14%"> </td>
      <td width="41%" bgcolor="#DBEAF5"><font face="Arial,
Helvetica, sans-serif"><b>Password:</b></font></td>
      <td width="45%">
        <input type="password" name="Password" size=15
maxlength="20">
      </td>
    </tr>
  </table>
  <blockquote>
      <INPUT TYPE="Submit" VALUE="Enviar Datos">
       
      <INPUT TYPE="Reset" VALUE="Limpiar">
  </blockquote>
</form>
<%end if%>
</body></html>
```

Fin del script: **borrar.asp**

Script Asp: **buscar.asp**

```
<html>
<head>
<title>B&uacute;queda de Clientes</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1"></head>
<body bgcolor="#FFFFFF" text="#000000">
<div align="center">
  <table width="75%" border="0">
    <tr><td><img src="images/ibdweb.gif" width="578" height="50"
align="middle" alt="Integraci&oacute;n de Bases de Datos en el
Web"></td></tr> </table></div>
<!--#include file="adovbs.inc"-->
<%if(Request.form = "") then %>
<table width="42%" border="0" height="63">
  <tr>
    <td background="images/bkgdhome.gif" height="58"><font
face="Times New Roman, Times, serif"><b><font color="#FFFFFF"
size="3">Escriba el dato o datos por los cuales va a realizar la
b&uacute;squeda. Si quiere ver toda la tabla deje los campos
vac&iacute;os.</font></b></font></td> </tr></table>
<form method="Post" action="buscar.asp">
  <table width="43%" border="0">
    <tr>
     <td width="26%" bgcolor="#DBEAF5"><b>Login:</b></td><td
width="74%"><input type="text" name="IDCliente" value=""></td>
</tr><tr><td width="26%" bgcolor="#DBEAF5"><b>Nombre:</b></td>
<td width="74%"><input type="text" name="Nombre" value=""></td>
</tr><tr><td width="26%" bgcolor="#DBEAF5"><b>Apellidos:</b>
</td><td width="74%"><input type="text" name="Apellidos"
value=""></td></tr> <tr>
 <td width="26%" bgcolor="#DBEAF5"><b>Domicilio:</b></td>
 <td width="74%">
   <input type="text" name="Domicilio" value=""></td></tr>  <tr>
      <td width="26%" bgcolor="#DBEAF5"><b>Ciudad</b></td>
      <td width="74%">
        <input type="text" name="Ciudad" value=""></td></tr>
    <tr>
      <td width="26%" bgcolor="#DBEAF5"><b>Pa&iacute;s:</b></td>
      <td width="74%">
        <input type="text" name="Pais" value="">
      </td>
    </tr>
  </table>
```

```
<p><input type="Submit" value="Enviar Datos"> 
    <input type="reset" value="Borrar"></p></form>
<%else
     if (request.form("pagina")="Página siguiente") then
               Session("pagina")=Session("pagina")+1
     else
          if(request.form("pagina")="Página anterior") then
               Session("pagina")=Session("pagina")-1
          else Session("pagina")=1
          end if
     end if
     rem Crear objeto de Connección
     Set ConBD=Server.CreateObject("ADODB.Connection")
     Set Ob_Rs=Server.CreateObject("ADODB.Recordset")
     ConBD.open "Conexion"
     Ob_Rs.open "Cliente", ConBD, adOpenStatic, adCmdTable
     poner_and=False
     filtro=""
     for each parametro in request.form
     if request.form(parametro)<>"" and parametro<>"pagina" then
       if poner_and then
          filtro=filtro & " and " & parametro & "='"
          filtro=filtro & request.form(parametro) & "'"
       else
          filtro=filtro & parametro & "='"
          filtro=filtro & request.form(parametro) & "'"
          poner_and=true
       end if
     end if
     next
     filtro = LCase(filtro)
     Ob_Rs.Filter=filtro
     if not Ob_Rs.eof then
          Ob_Rs.Pagesize=9
          Ob_Rs.AbsolutePage = Session("pagina")
     %>
<h3>Página <%=Session("pagina")%> de <%=Ob_Rs.PageCount%> </h3>
     <table border=1><tr>
          <%num_campos=Ob_Rs.Fields.Count
            for campo=0 to num_campos-1
               if campo=0 then%><td><b>Login</b></td>
```

./.

```
          <%elseif campo<>1 then%>
               <th><%=Ob_Rs(campo).name %> </th>
          <%  end if
          next
          registro=0
          do while registro<9 and not Ob_Rs.eof %>
             <tr>
             <%for campo=0 to num_campos-1
               if campo<>1 then
                  if Ob_Rs(campo)<>"" then%>
                    <td> <%=Ob_Rs(campo)%> </td>
                 <%else %> <td> No hay dato</td>
                <%  end if
                   end if
          next
          registro=registro+1
          Ob_Rs.movenext %>
          </tr>
          <%loop %>
          </table><br><form method="Post" action="buscar.asp">
          <%for each parametro in request.form
if request.form(parametro)<>"" and parametro<>"pagina" then %>
          <input type="hidden" name= <%=parametro%> value=
          <%=request.form("parametro")%> >
           <%end if
      next
if Session("pagina")>1 then %>
     <input type="Submit" name="pagina" value="Página anterior">
<%end if
if Session("pagina")< Ob_Rs.PageCount then %>
<input type="Submit" name="pagina" value="Página siguiente">
<%end if %>
     </form><p>
  <%else Response.write "<h3> No se ha encontrado ningún
registro</h3><br>"
end if %> <a href="index.asp">Página de inicio</a>  <a
href="bdweb.asp">P gina de Ejemplos</a>  <a
href="buscar.asp">Hacer nueva búsqueda</a></p>
  <%Ob_Rs.close
     ConBD.close
     Set Ob_Rs=nothing
     Set ConBD=nothing
end if %></body></html>
```

Fin del script: **buscar.asp**

7.4. DESARROLLO DE UNA APLICACIÓN CGI

Ahora corresponde a realizar todo lo anterior en CGI. El código a continuación.

Script cgi: **registrar.pl**

```
use DBI; #Cargar DBI
use CGI; #Cargar CGI
use CGI::Carp qw(fatalsToBrowser);

$dbName     ="Conexion"; #DSN
$dbDriver   ="ODBC";#Controlador
$dbUserName ="";
$dbPassword ="";

$dataIn =new CGI; #Crear CGI
$dataIn->header();#Crear cabecera HTML
$requestType = $dataIn->param('requestType');

$Login    = $dataIn->param('Login');    #campo requerido
$Password = $dataIn->param('Password'); #campo requerido
$TMPpasswd= $dataIn->param('tmpPasswd');#para comparar si
#password y password repetido son =.
$Nombre   = $dataIn->param('Nombre');   #campo requerido
$Apellido = $dataIn->param('Apellidos');#campo requerido
$Fecha_N  = $dataIn->param('F_Nac');    #campo requerido
$Domicilio = $dataIn->param('Domic');    #campo requerdio
$Ciudad = $dataIn->param('Ciudad');   #campo requerido
$Pais   = $dataIn->param('Pais');     #campo requerido
$Email  = $dataIn->param('email');    #campo requerido
$Telf   = $dataIn->param('Telf');
$Tipo_Neg = $dataIn->param('T_Nego');   #campo requerido
$Vol_Neg  = $dataIn->param('V_Nego');

#Evaluamos los datos del formulario.
if((defined $Login) && (defined $Password) && (defined
$TMPpasswd) && (defined $Nombre) && (defined $Apellido) &&
(defined $Fecha_N) &&(defined $Domicilio) && (defined $Telf)
&& (defined $Tipo_Neg) && (defined $Vol_Neg)){
($dia,$mes,$anyo) =split("/",$Fecha_N);
my($evaluar)=&Form_Eval($Login,$Password,$TMPpasswd,$Nombre,$A
pellido,$dia,$mes,$anyo,$Domicilio,$Ciudad,$Pais,$Email,
$Telf,$Tipo_Neg,$Vol_Neg);
if($evaluar!=0){&imprimir_form($evaluar);
                exit;}
```

```
 else {
  #Conectar a la base de datos.
  $dbh = &ConectarBD();
  #Verificar que los datos a insertar no ser n duplicados, se
comprueba por el Login.
  my $sql = "Select IDCliente from Cliente where IDCliente = '";
  $sql = $sql.$Login."';";
  my $resultado = Ejecutar_Sql($sql);
  @dbfila = &ObtenerDBFilas($resultado);
  $Columna = $dbfila[0];
 if ( $Columna eq $Login) { #Si Login existe entonces no se
#insertan los datos.
 print "Content-Type: text/html\n\n";
 print qq!
 <HEAD>
   <TITLE>Registro del cliente</TITLE>
 </HEAD>
 <H1>Login existente. No es posible agregar dichos datos.</H1>
 <A HREF="/bdweb/index.asp">Página de inicio</A>
  
 <A HREF="/bdweb/bdweb.asp">Página de Ejemplos</A>
  
 <A HREF="/cgi-bin/Clientes.pl">Probar de nuevo</A>!;
exit;}#fin del if.

#Prepara ejecución de inserción en la tabla Cliente
my $sth = $dbh->prepare(q { insert into Cliente (IDCliente,
Password, Nombre, Apellidos, Fecha_nac, Domicilio, Ciudad, Pais,
Telef, Email, Tipo_Negocio, Volumen_Negocio) values
(?,?,?,?,?,?,?,?,?,?,?,?)});
if($Domicilio eq "" || $Telf eq "") {
      if($Domicilio eq "") { $Domicilio = "Null";}
      else {$Telf = "Null";}
 }

#Ejecutar instrucción insert para inserción de los datos.
$sth->execute($Login, $Password, $Nombre, $Apellido, $Fecha_N,
$Domicilio, $Ciudad, $Pais, $Telf, $Email,$Tipo_Neg, $Vol_Neg);

#Realizar actualización en la tabla
$dbh->commit;
#Desconectar de la base de datos.
$dbh->disconnect;
```

./.

```
print "Content-Type: text/html\n\n";
print qq!
<HTML>
     <HEAD>
     <TITLE>Registro del cliente</TITLE>
     </HEAD>
     <BODY>
     <H1>Sus datos han sido insertados.</H1>
   <A HREF="/index.asp">Inicio</A> 
   <A HREF="/bdweb/bdweb.asp">P gina de Ejemplos</A> 
   <A HREF="/cgi-bin/registrar.pl">Agregar otro Cliente</A>
   </BODY></HTML>!;
 exit;
  }
 }
 &imprimir_form(0);
 #Fin del Script Principal
```

./.

```
sub Form_Eval{
        #Sacar los parametros del array por defecto @_.
        my($login) = Shift;
        my($password) = Shift;
        my($TmpPass) = Shift;
        my($Nombre) = Shift;
        my($Apellido) = Shift;
        my($dia) = Shift;
        my($mes) =Shift;
        my($anyo) = Shift;
        my($Domicilio) = Shift;
        my($Ciudad) = Shift;
        my($Pais)   = Shift;
        my($Email)  = Shift;
        my($Telf) = Shift;
        my($Tipo_N) = Shift;
        my($Vol_N) = Shift;
        my($bolean) = 0;

#Verificar que hay datos para Login, password, Nombre,
#Apellido, Domicilio, Ciudad, Pais, Fecha_N,Email,Tipo_N
if($login eq ""||$password eq ""||$Nombre eq ""||$Apellido eq
""||$Domicilio eq ""||$Ciudad eq ""||$Pais eq ""||$Email eq
""||$Fecha_N eq ""|| $Tipo_N eq ""){ $bolean=1;
     }
elsif($login !~/(\w+)[a-zA-Z0-9]/ || (length($login))< 5){
                  $bolean=2;}
elsif(($password !~/(.+)/)|| ($password ne $TmpPass)){
                  $bolean=3;}
elsif($Email !~/(.*@.*\.[a-zA-Z]{2,3}$)/ &&
$Email=~/(^\.)|(\.$)|( )|(\.\.)|(@\.)|(\.@)|(@.*@)/){
                 $bolean=4;}
elsif((length($Telf))>9 && $Telf =~/^\(?\d+\)?/){
                  $bolean=5;}
return $bolean;
}
```

./.

Procedimiento que imprime el formulario de entrada

```
#Subrrutina que imprime el formulario de entrada
sub imprimir_form{
        my $casoerr = shift;
        print "Content-Type: text/html\n\n";
        print qq!
        <HEAD>
          <TITLE>Registro del cliente</TITLE>
        </HEAD>
       <table width="68%" border="0" align="center">
         <tr>
           <td colspan="2">
           <div align="left"><img src="/bdweb/images/ibdweb.gif"
           width="516" height="50" alt="Integraci&oacute;n de
           Bases de Datos en el Web"></div>
           </td>
         </tr></table><hr>
<FORM name="form1" method="post" action="/cgi-bin/registrar.pl">
    <table with="84%" border="0" align="center">
    <tr><td colspan="3">
     <h2 align="center"><font color="#FFFFFF" face="Arial,
     Helvetica, sans-serif"><b><font color="#003366">Registro
     del Cliente</font></b></font></h2>
     <hr>
      <p><font face="Arial, Helvetica, sans-serif" size="3"
color="#FF0000"><b>!;
     #Comprobar los errores de existentes.
     if($casoerr==1){
     print qq!Error al llenar datos del formulario. Por favor
     ingreso sus datos correctamente.!;}
     elsif($casoerr==2){print qq!Error: Login incorrecto.!;}
     elsif($casoerr==3){print qq!Error: Password incorrecto.!;}
     elsif($casoerr==4){print qq!Error: E-Mail incorrecto.!;}
     elsif($casoerr==5){print qq!Error: Tele&acute;fono
     incorrecto. Por ejemplo para Nicaragua: (505)-2786525!;}
     print qq!</b></font></p></td></tr></table>!;
     print qq!<table width="84%" border="0" align="center">
<tr>
    <td colspan="3" background="/bdweb/images/bkgdhome.gif">
    <h2 align="left"><font face="Times New Roman, Times, serif"
color="#ffcc00"><b> Datos personales</b></font></h2>
    </td>
</tr><tr>
      <td colspan="3">
```

```
<p align="left"><font face="Times New Roman, Times, serif"
size="3" color="#996633"><b>Los campos que aparecen con
ast&eacute;riscos son datos requeridos.</b></font></p>
</td>
</tr>
  <tr>
<td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Nombres<sup>*</sup>:</font></b>
</td>
<td width="26%">
  <input type="text" name="Nombre" size="20" maxlength="30"
value="$Nombre">
 </td>
 <td width="43%"> </td>
</tr>
<tr>
 <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Apellidos<sup>*</sup>:</font>
</b></td>
  <td width="26%">
    <input type="text" name="Apellidos" size="20" maxlength="30"
value="$Apellido">
  </td>
  <td width="43%"> </td>
</tr>
<tr>
  <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Domicilio<sup>*</sup>:</font>
</b></td>
<td width="26%">
 <input type="text" name="Domic" size="20" maxlength="30"
value="$Domicilio">
 </td>
 <td width="43%"> </td>
</tr>
  <tr>
  <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Ciudad<sup>*</sup>:</font>
</b></td><td width="26%">
 <input type="text" name="Ciudad" size="20" maxlength="30"
value="$Ciudad"></td>
 <td width="43%"> </td>
</tr>
```

```
 <tr>
      <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif"
size="2">Pa&iacute;s<sup>*</sup>:</font></b></td>
      <td width="26%">
        <select name="Pais" size="1">
           <option value="Nicaragua" selected>Nicaragua</option>
           <option value="Belice">Belice</option>
           <option value="Guatemala">Guatemala</option>
           <option value="El Salvador">El Salvador</option>
           <option value="Honduras">Honduras</option>
           <option value="Costa Rica">Costa Rica</option>
           <option value="Panama">Panam&aacute;</option>
         </select>
      </td>
      <td width="43%"> </td>
    </tr>
    <tr>
 <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Fecha de Nacimiento<sup>*</sup>:
</font></b></td>
<td width="26%">
<input type="text" name="F_Nac" size="20" maxlength="25"
value="$Fecha_N">
 </td>
    <td width="43%"> (dd/mm/aaaa)</td>
    </tr>
    <tr>
      <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Tel&eacute;fono:</font></b></td>
<td width="26%">
<input type="text" name="Telf" size="12" maxlength="14"
value="$Telf"></td>
<td width="43%"> </td>
  </tr>
<tr>
 <td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">E-mail<sup>*</sup>:</font></b>
</td>
<td width="26%">
<input type="text" name="email" size="20" maxlength="25"
value="$Email">
</td>
      <td width="43%"> </td>
</tr><tr>
```

```
<td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Elija un Login <sup>*</sup>:
</font></b></td><td width="26%">
<input type="text" name="Login" size="20" maxlength="25"
value="$Login"></td>
<td width="43%" bgcolor="#0099CC"
background="/bdweb/images/bkgdhome.gif">
<div align="left"><font color="#FFFFFF">El login debe comenzar
por una letra y sólo puede contener letras, números y el
carácter de subrayado (_) pero sin espacios. </font></div>
</td></tr><tr>
<td width="27%" bgcolor="#DBEAF5"><b><font face="Arial,
Helvetica, sans-serif" size="2">Password <sup>*</sup>:</font>
</b></td><td width="26%">
<input type="password" name="Password" size="20" maxlength="25">
</td>
<td width="43%" bgcolor="#0099CC"
background="/bdweb/images/bkgdhome.gif">
<div align="left"><font color="#FFFFFF">El password debe tener
al menos seis car cteres de longitud y puede contener números y
letras, pero no puede incluir ningún espacio. Las contraseñas
distinguen entre mayúsculas y minúsculas. </font></div></td>
</tr><tr>
<td width="27%" bgcolor="#DBEAF5"><font face="Arial, Helvetica,
sans-serif" size="2"><b>Repetir Password:</b></font></td>
<td width="26%">
<input type="password" name="tmpPasswd" size="20"
maxlength="25">
</td><td width="43%"></td></tr><tr><td colspan="3"></td></tr>
<tr><td width="27%" bgcolor="#DBEAF5"><font face="Times New
Roman, Times, serif" color="#000000"><b>Tipo del Negocio<sup>*
</sup>:</b></font></td><td width="26%">
<input type="text" name="T_Nego" size="25" value="$Tipo_Neg">
</td><td width="43%"> </td></tr>
<tr><td width="27%" bgcolor="#DBEAF5" height="5"><b>Volumen del
Negocio<sup>*</sup>:</b></td><td width="26%" height="5">
<select name="V_Nego" size="1">
          <option selected>Ninguno</option>
          <option>Peque&ntilde;o</option>
          <option>Mediano</option>
          <option>Grande</option>
        </select></td><td width="43%" height="5"> </td>
</tr>
```

./.

```
<tr>
     <td width="27%">   </td>
   </tr>
  </table>
   <blockquote>
    <blockquote>
    <p>
          <input type="submit" name="Submit" value="Enviar">
            
          <input type="reset" name="Limpiar" value="Limpiar">
    </p>
     </blockquote>
  </blockquote>
</FORM>!;
}
```

Procedimientos Varios

```
########################################
#Procedimiento que conecta con la BD.#
########################################
sub ConectarBD {
return(DBI->connect($dbName,$dbUserName,$dbPassword,
$dbDriver,{RaiseError => 1, AutoCommit => 0}))|| die "No se
puede conectar a la base de datos";
}

##############################################
#Procedimiento que ejecuta sentencia Select.#
##############################################
sub Ejecutar_Sql{
     my($sql) = shift;
     $dataObject = $dbh->prepare($sql);
     $dataObject->execute();
     return $dataObject;
}

################################################
#Procedimiento que extrae la fila con el Login.#
################################################
sub ObtenerDBFilas{
        my ($dataObject) = shift;
        return $dataObject->fetchrow_array();
}
```

Script cgi: **modificar.pl**

```
use DBI; #Cargar DBI
use CGI; #Cargar CGI
use CGI::Carp qw(fatalsToBrowser);

$dbName     ="Conexion"; #DSN
$dbDriver   ="ODBC";#Controlador
$dbUserName ="";
$dbPassword ="";

$dataIn =new CGI; #Crear CGI
$dataIn->header();#Crear cabecera HTML
$requestType = $dataIn->param('requestType');

$Login          = $dataIn->param('Login');    #campo requerido
$Password       = $dataIn->param('Password'); #campo requerido
$Nombre         = $dataIn->param('Nombres');   #campo requerido
$Apellido       = $dataIn->param('Apellidos');#campo requerido
$Fecha_N        = $dataIn->param("Fecha de Nac.");#campo
#requerido
$Domicilio      = $dataIn->param('Domicilio'); #campo requerdio
$Ciudad         = $dataIn->param('Ciudad');   #campo requerido
$Pais           = $dataIn->param('Pais');     #campo requerido
$Email          = $dataIn->param('Email');    #campo requerido
$Telf           = $dataIn->param('Telefono');
$Tipo_Neg       = $dataIn->param("Tipo Negocio");#campo requerido
$Vol_Neg        = $dataIn->param("Volumen Negocio");
$Modificar      = $dataIn->param('modificar');

#Evaluamos los datos del formulario.
if(defined $Modificar && $Modificar eq "Modificar"){
my($evaluar)=&Form_Eval($Nombre,$Apellido,$Fecha_N,$Domicilio,$Ci
udad,$Pais,$Email,$Telf,$Tipo_Neg, $Vol_Neg);
if($evaluar!=0){
@campos_t=($Login,$Password,$Nombre,$Apellido,$Fecha_N,$Domicilio
,$Ciudad,$Pais,$Tipo_Neg,$Vol_Neg,$Telef,$Email);
&imprimir_form($evaluar,@campos_t);
exit;
}
else { #Conectar a la base de datos.
 $dbh = &ConectarBD();
 #Prepara ejecución de actualizaci¢n en la tabla Cliente
my $sth = $dbh->prepare(q {UPDATE Cliente set Nombre=?,
Apellidos=?, Fecha_nac=?, Domicilio=?, Ciudad=?, Pais=?,
Tipo_Negocio=?, Volumen_Negocio=?, Telef=?, Email=? where
IDCliente=?});
```

```
#Ejecutar instrucci¢n update para actulizaci¢n de los datos.
$sth->execute($Nombre,$Apellido,$Fecha_N,$Domicilio,$Ciudad,
$Pais,$Tipo_Neg,$Volumen_Neg,$Telef,$Email,$Login);
#Realizar actualización en la tabla
$sth->finish();
$dbh->commit;
#Desconectar de la base de datos.
$dbh->disconnect;

header();
print qq!<h3>Registro actualizado</h3>
<A HREF="/index.asp">Inicio</A>
 
<A HREF="/bdweb/bdweb.asp">P gina de Ejemplos</A>
 
<A HREF="/cgi-bin/modificar.pl">Actulizar otro</A>!;
exit;
}
}
elsif((defined $Login) && (defined $Password)){
   my($evaluar) = &Evaluar_Usr($Login,$Password);
   if($evaluar!=0){
    &form_entrada($evaluar);
    exit;
   }
   else{
     #Conectar a la base de datos.
     $dbh = &ConectarBD();
     #Definir Consulta SQL
     my $sql = "Select * from Cliente where IDCliente = '$Login'
     and Password='$Password'";
     my $resultado = Ejecutar_Select($sql);
     @dbfila = &ObtenerDBFilas($resultado);
     header();
     print qq!<CENTER><form name="frm_modificar" action=/cgi-
     bin/modificar.pl method=post>
     <TABLE BORDER = "1">!;
     @campos=("Login","Password","Nombres","Apellidos","Fecha de
     Nac.","Domicilio","Ciudad","Pais","Tipo Negocio","Volumen
     Negocio","Telefono","Email");
     $i=0;
     foreach $rowReference (@dbfila){
        foreach $columnReference (@$rowReference){
             foreach $column (@$columnReference){
                print qq!<tr><td>$campos[$i]</td>!;
```

```
if($i==4){
 ($year,$month,$day)=split(/-/,$column);
 ($day,$basura,$basura)=split(/ /,$day);
 @fecha=("$day","$month","$year");
 $column=join("/",@fecha);
}
if($i==0){
  print qq!<td><font color="blue">$column</font></td></tr>
  <input type="hidden" name="$campos[$i++]" value="$column">\n!;
}
elsif($i==1){
  print qq!<td><input type="hidden" name="$campos[$i++]">
  <font color="red">No disponible </font> </td>!;}
else{ print qq!<td><input type="text" name="$campos[$i++]"
value="$column"\n></td></tr>!;}
}#Fin del 3er foreach
}#Fin del 2do foreach
}#Fin del 1er foreach
print qq!
</TABLE>
<blockquote>
<input type="submit" name="modificar" value="Modificar">
 
<input type="reset" name="cancelar" value="Cancelar">
</blockquote>
</form>
</CENTER>
!;
exit;
}#Fin del else.
}#Fin del if.
&form_entrada(0);
exit;
#Fin del Script principal.
```

Prodimientos varios

```
########################################
#Procedimiento que pide login y password#
########################################
sub form_entrada{
        my $casoerror = Shift;
        header();
        print qq!
        <table width="46%" border="0">
          <tr><td width="10%" valign="bottom"> </td>
            <td background="/bdweb/images/bkgdhome.gif"
width="90%" valign="bottom"> <b><font color="#FFFFFF"
face="Times New Roman, Times, serif" size="5">Escriba
el login y contraseña del usuario a modificar</font></b></td>
</tr></table>
<br>
  <form method= "post" action="/cgi-bin/modificar.pl" >
   <table width="33%" border="0"><tr><td>
   <p><font face="Arial, Helvetica, sans-serif" size="3"
color="#FF0000"><b>!;
 #Comprobar los errores de existentes.
 if($casoerr==1){
     print qq!El Login y el Password no pueden estar vac¡os.!;}
 elsif($casoerr==2) { print qq!Login incorrecto.!;}
 elsif($casoerr==3) { print qq!Password incorrecto.!;}
 print qq!</b></font></p></td></tr>
 <tr>
   <td width="14%"> </td>
   <td width="35%" bgcolor="#DBEAF5"><font face="Arial,
Helvetica, sans-serif"><b>Login:
</b></font></td>
<td width="51%">
  <input name="Login" size=15 maxlength="15">
</td>
</tr>
<tr>
  <td width="14%"> </td>
  <td width="35%" bgcolor="#DBEAF5"><font face="Arial,
Helvetica, sans-serif"><b>Password:</b></font></td>
  <td width="51%">
<input type="password" name="Password" size=15 maxlength="20">
</td></tr></table><blockquote>
<INPUT TYPE="Submit" VALUE="Enviar Datos"> 
<INPUT TYPE="Reset" VALUE="Limpiar"></blockquote></form>!;
}
```

```
sub Evaluar_Usr{
    #Sacar los parametros del array por defecto @_.
    my($login) = Shift;
    my($password) = Shift;
    my($bolean) = 0;

    #Verificar que hay datos para Login, password.
    if($login eq ""||$password eq ""){
        $bolean=1;
    }
    elsif($login !~/(\w+)[a-zA-Z0-9]/ || (length($login))< 5){
         $bolean=2;}
    elsif(($password !~/(.+)/)){$bolean=3;}
    return $bolean;
}

########################################
#Procedimiento que conecta con la BD.#
########################################
sub ConectarBD {
return(DBI->connect($dbName,$dbUserName,$dbPassword,$dbDriver,
{RaiseError => 1, AutoCommit => 0}))|| die "No se puede conectar
a la base de datos";
}

##############################################
#Procedimiento que ejecuta sentencia Sql.#
##############################################
sub Ejecutar_Sql{
my($sql) = shift;
$resultado = $dbh->do($sql)|| die "\nError al ejecutar sentencia
sql\n";
return($resultado);
}

##############################################
#Procedimiento que ejecuta sentencia Select.#
##############################################
sub Ejecutar_Select{
     my($sql) = shift;
     $dataObject = $dbh->prepare($sql);
     $dataObject->execute();
     return $dataObject;
}
```

```
##############################################
#Procedimiento que imprimi la cabecera html.#
##############################################
sub header{ print "Content-Type: text/html\n\n";
  print qq!<HTML><HEAD><TITLE>Actualizar registro</TITLE>
  </HEAD><p> </p><table width="68%" border="0"
align="center"><tr><td colspan="2"><div align="left"><img
src="/bdweb/images/ibdweb.gif" width="522" height="50"
alt="Integraci&oacute;n de Bases de Datos en el
Web"></div></td></tr></table><hr align="left">!;
}

sub imprimir_form{
  my $casoerr = shift;
  my $i = 0;
  my @campos=("Login","Password","Nombres","Apellidos","Fecha de
Nac.","Domicilio","Ciudad","Pais","Tipo Negocio","Volumen
Negocio","Telefono","Email");
  header();
  print qq!<CENTER>
  <table with="84%" border="0"><tr><td>
 <p><font face="Arial, Helvetica, sans-serif" size="3"
color="#FF0000"><b>!;
#Comprobar los errores de existentes.
if($casoerr==1){print qq!Error al modificar datos del
formulario. Por favor ingreso sus datos correctamente.!;}
elsif($casoerr==2) {print qq!Error: E-Mail incorrecto.!;}
elsif($casoerr==3) {print qq!Error: Tele&acute;fono incorrecto.
Por ejemplo para Nicaragua: (505)-2786525!;}
elsif($casoerr==4) {print qq!Error: Fecha incorrecta.!;}
print qq!</b></font></p></td></tr></table>
<form name="frm_modificar" action="/cgi-bin/modificar.pl"
method="post"><TABLE BORDER = "1">!;
foreach $column (@_){print qq!<tr><td>$campos[$i]</td>!;
if($i==0){print qq!<td><font color="blue">$column</font></td>
</tr><input type="hidden" name="$campos[$i++]"
value="$column">\n!;
}
elsif($i==1){print qq!<td><input type="hidden"
name="$campos[$i++]"><font color="red">No disponible </font>
</td>!;}
else{ print qq!<td><input type="text" name="$campos[$i++]"
value="$column"\n></td></tr>!;}}#Fin del foreach
print qq!</TABLE><blockquote><input type="submit"
name="modificar" value="Modificar"> 
<input type="reset" name="cancelar" value="Cancelar">
</blockquote></form></CENTER>!;}
```

```
sub Form_Eval{
        #Sacar los parametros del array por defecto @_.
        my($Nombre) = Shift;
        my($Apellido) = Shift;
        my($Fecha_N) = Shift;
        my($Domicilio) = Shift;
        my($Ciudad) = Shift;
        my($Pais)   = Shift;
        my($Email)  = Shift;
        my($Telf) = Shift;
        my($Tipo_N) = Shift;
        my($Vol_N) = Shift;
        my($bolean) = 0;

#Verificar que hay datos para Nombre, Apellido, Domicilio,
Ciudad, Pais, Fecha_N,Email,Tipo_N
if($Nombre eq ""||$Apellido eq ""||$Fecha_N eq ""||$Domicilio eq
""||$Ciudad eq ""||$Pais eq ""||$Email eq ""|| $Tipo_N eq ""){
          $bolean=1;
}
elsif($Email !~/(.*@.*\.[a-zA-Z]{2,3}$)/ && $Email = ~ / (^\.) |
(\.$)|( )|(\.\.)|(@\.)|(\.@)|(@.*@)/){
   $bolean=2;}
elsif((length($Telf))>9 && $Telf =~/^\(?\d+\)?/){
   $bolean=3}
return $bolean;
}

sub ObtenerDBFilas
        {
        my ($dataObject) = shift;
        return $dataObject->fetchall_arrayref();
        }
```

Fin del script cgi: **modificar.pl**

Script cgi: **borrar.pl**

```
use DBI;
use CGI;
use CGI::Carp qw(fatalsToBrowser);

$dbName = "Conexion";
$dbDriver = "ODBC";
$dbUserName = "";
$dbPassword = "";

$dataIn = new CGI;
$dataIn->header();
$requestType = $dataIn->param('requestType');

$Login = $dataIn->param('Login');
$Password = $dataIn->param('Password');

#Evaluamos los datos del formulario.
       if((defined $Login) && (defined $Password)){
           my($evaluar) = &Form_Eval($Login,$Password);
           if($evaluar!=0){
              &imprimir_form($evaluar);
             exit;
           }
           else {
              #Conectar a la base de datos.
              $dbh = &ConectarBD();
              #Definir Consulta SQL
               $sqltext = "Delete * from Cliente where
               IDCliente='$Login' and Password='$Password';";
              my $resultado = Ejecutar_Sql($sqltext);
              print "Content-Type: text/html\n\n";
              print qq!<HTML><HEAD><TITLE>Darse de Baja</TITLE>
              </HEAD><p> </p>
            <table width="68%" border="0" align="center">
             <tr><td colspan="2"><div align="left">
     <img src="/bdweb/images/ibdweb.gif" width="522" height="50"
     alt="Integraci&oacute;n de Bases de Datos en el Web"></div>
     </td></tr></table><hr align="left">!;
#Evaluar si se borr¢ el registro solicitado.
if($resultado==0){print qq!<h3>Registro no encontrado</h3>!;}
```

```
else {
    print qq!<h3>Registro borrado.</h3>!;}
    print qq!<A HREF="/index.asp">Página de Inicio</A>
     
    <A HREF="/bdweb/bdweb.asp">Página de Ejemplos</A>
     
    <A HREF="/cgi-bin/borrar.pl">Borrar otro</A>!;
    #Desconectar de la base de datos.
    $dbh->disconnect;
    exit;
  }
}
  &imprimir_form(0);
  #Fin del Script Principal

##########################################
#Procedimiento que evalua el formulario. #
##########################################

sub Form_Eval{
   #Sacar los parametros del array por defecto @_.
   my($login,$password) = Shift;
   my($bolean) = 0;

   #Verificar que hay datos para Login, password.
   if($login eq ""||$password eq ""){$bolean=1; }
   elsif($login !~/(\w+)[a-zA-Z0-9]/ || (length($login))< 5){
        $bolean=2;}
   elsif(($password !~/(.+)/)|| ($password ne $TmpPass)){
               $bolean=3;}
   return $bolean;
}
##########################################
#Procedimiento que imprimi el formulario.#
##########################################
sub imprimir_form{
my $casoerror = shift;
print "Content-Type: text/html\n\n";
print qq!<html><head></head><title>Borrar Registro del
Cliente</title><p>
<table width="68%" border="0" align="center"><tr>
 <td colspan="2"> <div align="left"><img
src="/bdweb/images/ibdweb.gif" width="522" height="50"
alt="Integraci&oacute;n de Bases de Datos en el Web"></div></td>
```

```
</tr></table>
 <hr align="left">
 <table width="32%" border="0">
 <tr valign="bottom">
<td width="13%" align="center" align="bottom"> </td>
  <td width="87%" background="/bdweb/images/bkgdhome.gif"
align="center" valign="bottom">
 <div align="left"><font size="5" face="Times New Roman, Times,
serif"><b><font color="#FFFFFF"> Darse de baja</font></b>
</font></div></td>
     </tr>
          </table>!;
    if($casoerror==1){
    print qq!Error al llenar datos del formulario. Por favor
ingreso sus datos correctamente.!;}
    elsif($casoerror==2){
       print qq!Error: Login incorrecto.!;}
    elsif($casoerror==3){
       print qq!Error: Password incorrecto.!;}

  print qq!<form name="FormBorrar" method= "post"
action="borrar.pl" >
<table width="33%" border="0">
      <tr valign="bottom">
<td height="40" bgcolor="#ffffff" width="14%">
<div align="left"><p></p></div></td>
<td height="40" bgcolor="#ffffff" colspan="2"><font face="Arial,
Helvetica, sans-serif" color="#996633" size="2"><b>ESCRIBA
EL LOGIN Y LA CLAVE DEL CLIENTE A BORRAR</b></font></td></tr>
<tr><td width="14%"> </td>
<td width="41%" bgcolor="#DBEAF5"><font face="Arial, Helvetica,
sans-serif"><b>Login: </b></font></td>
<td width="45%"><input name="Login" size=15 maxlength="15"></td>
</tr><tr><td width="14%"> </td>
<td width="41%" bgcolor="#DBEAF5"><font face="Arial, Helvetica,
sans-serif"><b>Password:</b></font></td><td width="45%">
<input type="password" name="Password" size=15 maxlength="20">
</td></tr></table>
  <blockquote>
     <INPUT TYPE="Submit" VALUE="Enviar Datos"> 
     <INPUT TYPE="Reset" VALUE="Limpiar">
 </blockquote>
 </form>!;
}
```

```
########################################
#Procedimiento que conecta con la BD.#
########################################
sub ConectarBD {
        return(DBI-
>connect($dbName,$dbUserName,$dbPassword,$dbDriver,{RaiseError
=> 1, AutoCommit => 0}))|| die "No se puede conectar a la base
de datos";
}

##############################################
#Procedimiento que ejecuta sentencia Sql.#
##############################################
sub Ejecutar_Sql{
     my($sql) = shift;
        $resultado = $dbh->do($sql)|| die "\nError al ejecutar
sentencia sql\n";
     return($resultado);
}
```

Script cgi: **buscar.pl**

```
use DBI; #Cargar DBI
use CGI; #Cargar CGI
use CGI::Carp qw(fatalsToBrowser);

$dbName     ="Conexion"; #DSN
$dbDriver   ="ODBC";#Controlador
$dbUserName ="";
$dbPassword ="";

$dataIn =new CGI; #Crear CGI
$dataIn->header();#Crear cabecera HTML
$requestType = $dataIn->param('requestType');

$Login  = $dataIn->param('Login');
$Nombre = $dataIn->param('Nombre');
$Apellido = $dataIn->param('Apellido');
$Domicilio = $dataIn->param('Domicilio');
$Ciudad = $dataIn->param('Ciudad');
$Pais = $dataIn->param('Pais');
$Consulta = $dataIn->param('Consultar');

if(defined $Consulta && $Consulta eq "Consultar" && defined
$Login && defined $Nombre && defined $Apellido && defined
$Domicilio && defined $Ciudad && defined $Pais){
 #Conectar a la base de datos.
 $dbh = &ConectarBD();

#Definir la consulta
if($Login eq "" && $Nombre eq "" && $Apellido eq "" &&
$Domicilio eq "" && $Ciudad eq "" && $Pais eq ""){
$sql = "Select * from Cliente";
}#fin del if
elsif($Login ne ""){$sql = "Select * from Cliente where
IDCliente='$Login'";
  if($Nombre ne ""){$sql = $sql." and Nombre='$Nombre'";
    if($Apellido ne ""){$sql= $sql." and Apellidos='$Apellido'";
      if($Domicilio ne ""){$sql = $sql." And
Domicilio='$Domicilio'";
        if($Ciudad ne ""){$sql = $sql." and Ciudad='$Ciudad'";
          if($Pais ne "" ){$sql = $sql." and Pais='$Pais'";}
        }#fin del if
      }#fin del if
     }#fin del if
   }#fin del if
}#fin del if
```

```
elsif($Nombre ne ""){
    $sql = "Select * from Cliente where Nombre='$Nombre'";
    if($Apellido ne ""){
      $sql= $sql." and Apellidos='$Apellido'";
      if($Domicilio ne ""){
        $sql = $sql." and Domicilio='$Domicilio'";
        if($Ciudad ne ""){
          $sql = $sql." and Ciudad='$Ciudad'";
          if($Pais ne "" ){
            $sql = $sql." and Pais='$Pais'";
          }
        }
      }
    }
}
elsif($Apellido ne ""){
 $sql= "Select * from Cliente where Apellidos='$Apellido'";
if($Domicilio ne ""){
 $sql = $sql." and Domicilio='$Domicilio'";
 if($Ciudad ne ""){
   $sql = $sql." and Ciudad='$Ciudad'";
   if($Pais ne "" ){
     $sql = $sql." and Pais='$Pais'";
    }
  }
 }
}
elsif($Domicilio ne ""){
   $sql = "Select * from Cliente where Domicilio='$Domicilio";
   if($Ciudad ne ""){
     $sql = $sql." and Ciudad='$Ciudad'";
     if($Pais ne "" ){
       $sql = $sql." and Pais='$Pais'";
      }
   }
 }
elsif($Ciudad ne ""){
   $sql = $sql." and Ciudad='$Ciudad'";
   if($Pais ne "" ){$sql = $sql." and Pais='$Pais'";
   }
}
elsif($Pais ne ""){
  $sql = "Select * from Cliente where Pais='$Pais'";
}
```

```
#Prepara ejecuci¢n de selecci¢n en la tabla Cliente
      $resultado = Ejecutar_Select($sql);
      @dbfila = &ObtenerDBFilas($resultado);
      header();
      print qq!<table border=1><tr>!;
      @campos=("Login","Password","Nombres","Apellidos","Fecha
de Nac.","Domicilio","Ciudad","Pais","Tipo Negocio","Volumen
Negocio","Telefono","Email");
      $i=0;
      foreach $reg (@campos){
        if($i!=1){
         print qq!<th>$reg</th>!;}
         $i++;
      }
      print qq!</tr>!;
      foreach $rowReference (@dbfila){
        foreach $columnReference (@$rowReference){
           print qq!<tr>!;
           $i=0;
           foreach $column (@$columnReference){
             if($i!=1){
                if($i==4){
                   ($year,$month,$day)=split(/-/,$column);
                   ($day,$basura,$basura)=split(/ /,$day);
                   @fecha=("$day","$month","$year");
                   $column=join("/",@fecha);
                }
             if($column eq ""){
             print qq!<td>No hay dato</td>\n!;
            }#fin del if
            else {print qq!<td>$column</td>\n!;}
           }#fin del if
           $i++;
         }#fin del 3er foreach
         print qq!</tr>!;
       }#fin del 2do foreach
     }#fin del 1er foreach
print qq!</table><blockquote><a href="/index.asp">P gina de
inicio</a> <a href="/bdweb/bdweb.asp">P gina de Ejemplos
</a> <a href="/cgi-bin/buscar.pl">Hacer nueva  b£squeda</a>
</blockquote>!;
exit;
 }
 #Formulario de B£squeda
 form_busqueda();
 exit;
```

```
##########################################
#Procedimiento que imprime formulario    #
##########################################
sub form_busqueda{
    header();
    print qq!<table width="42%" border="0" height="63">
          <tr>
            <td background="/bdweb/images/bkgdhome.gif"
height="58"><font face="Times New Roman, Times, serif"><b><font
color="#FFFFFF" size="3">Escriba el dato o datos por los cuales
va a realizar la b&uacute;squeda. Si quiere ver toda la tabla
deje los campos vac&iacute;os.</font></b></font></td>
</tr></table><form method="Post" action="/cgi-bin/buscar.pl">
<table width="43%" border="0"><tr><td width="26%"
bgcolor="#DBEAF5"><b>Login:</b></td>
     <td width="74%">
       <input type="text" name="Login" value="">
     </td>
  </tr>
  <tr>
     <td width="26%" bgcolor="#DBEAF5"><b>Nombre:</b></td>
     <td width="74%">
       <input type="text" name="Nombre" value="">
     </td>
  </tr>
  <tr>
     <td width="26%" bgcolor="#DBEAF5"><b>Apellidos:</b></td>
     <td width="74%">
       <input type="text" name="Apellido" value="">
     </td>
  </tr>
  <tr>
     <td width="26%" bgcolor="#DBEAF5"><b>Domicilio:</b></td>
     <td width="74%">
       <input type="text" name="Domicilio" value="">
     </td>
  </tr>
  <tr>
     <td width="26%" bgcolor="#DBEAF5"><b>Ciudad</b></td>
     <td width="74%">
       <input type="text" name="Ciudad" value="">
     </td>
  </tr>
  <tr><td width="26%" bgcolor="#DBEAF5"><b>Pa&iacute;s:</b></td>
  <td width="74%"><input type="text" name="Pais" value=""></td>
  </tr></table>
```

```
<p><input type="Submit" name="Consultar" value="Consultar">
  
 <input type="reset" value="Borrar">
 </p>
 </form>!;
}

##############################################
#Procedimiento que ejecuta sentencia Select.#
##############################################
sub Ejecutar_Select{
     my($sql) = shift;
     $dataObject = $dbh->prepare($sql);
     $dataObject->execute();
     return $dataObject;
}

#######################################
#Procedimiento que conecta con la BD.#
#######################################
sub ConectarBD {
  return(DBI->connect($dbName,$dbUserName,$dbPassword,$dbDriver,
{RaiseError => 1, AutoCommit => 0}))|| die "No se puede conectar
a la base de datos";
}

sub ObtenerDBFilas
        {
        my ($dataObject) = shift;
        return $dataObject->fetchall_arrayref();
        }
##############################################
#Procedimiento que imprimi la cabecera html.#
##############################################
sub header{
    print "Content-Type: text/html\n\n";
    print qq!<HTML><HEAD>
                <TITLE>Buscar registro</TITLE>
    </HEAD><p>  </p>
    <table width="68%" border="0" align="center">
    <tr><td colspan="2"><div align="left">
<img src="/bdweb/images/ibdweb.gif" width="522" height="50"
alt="Integraci&oacute;n de Bases de Datos en el Web"></div>
</td></tr></table><hr align="left">!;
}
```

VII. CONCLUSIÓN

Al realizar el estudio de las tecnologías de integración de bases de datos en el web más comunes actualmente como son ASP y CGI, se ha llegado a la conclusión de que se han cumplido con los objetivos propuestos, se han abordado cada una de las tecnologías haciendo énfasis en sus ventajas y desventajas, y notándose la diferencias que existe entre ellas, que podría resumirse en que:

1. ASP está ligado a tecnología Microsoft, el caso de que funciona principalmente en Sistemas Windows, mientras que CGI es una tecnología de amplio uso en distintos sistemas y no está ligado a un producto software web específico.
2. CGI usando los módulos y controladores correspondientes para acceso a base de datos aparenta ser de fácil aprendizaje, pero de hecho esto requiere un previo estudio o conocimiento amplio del lenguaje (por ejemplo Perl) en que se desarrollen, no tanto así de las librerías usadas para esto, como es el caso de DBI; en cambio ASP si presenta no sólo un fácil aprendizaje para aquella persona acostumbrada a trabajar en productos Microsoft, y de hecho es más comprensible su entorno de desarrollo.
3. Y probablemente la más importante de todas es que ASP provee de una mejor administración de las conexiones que se realizan a una base de datos determinada, puesto que puede mantener activa una o varias conexiones antes de que otro usuario haga una solicitud, aumentando el rendimiento de la aplicación; mientras que CGI no provee este mismo aprovechamiento.

VIII. BIBLIOGRAFÍA

[1] Bobadilla Jesús y Alcocer Alejandro. Creación de aplicaciones web en Windows NT Active Server Pages. Editorial ALFAOMEGA GRUPO EDITOR, S.A. 489 páginas.

[2] Sinclair Joseph y McCullough Carol. Creación de Bases de Datos en Internet. Ediciones Anaya Multimedia. Madrid, España, 1997. 504 páginas.

[3] Documentación de Microsoft Internet Information Server (incluida al instalarse el Option Pack 4 de NT).

Referencias en Internet:

[4] Sitios sobre programación en ASP:

http://www.asp.com

http://www.asptoday.com

http://www.learnasp.com

http://www.4guysfromrolla.com

[5] Sitio para encontrar manuales de programación, cursos, tutoriales, diccionario informático, etc.:

http://www.lawebdelprogramador.com

[6] Enciclopedia de términos informáticos:

http://www.whatis.com

[7] Sitio para descargar el intérprete de Perl y documentación del lenguaje:

http://www.ActiveState.com

[8] Sitio oficial del lenguaje Perl:

http://www.perl.com

[9] Páginas del módulo DBI y de la lista de drivers:

http://www.hermetica.com/technologia/perl/DBI

[10] Información sobre DBI:

http://www.fuge.com/dbi/

http://www.aranca.co.uk/technology/perl/DBI

http://www.jc.com/sql_stnd.html