Tesis de Licenciatura
En
Sistemas de Información

Materia: Preparación y Evaluación de Proyectos

Profesor: Jorge Peri

Director de Tesis: Javier Blanqué

Alumna: Myrian C. Hernández

Tema: Desarrollo de Intranets utilizando Internet Information Server, Active Server Pages y SQL Server

Resúmen:

Esta tesis tiene como objetivo explicar el uso de herramientas que utilizan tecnologías propias de Internet para el desarrollo de aplicaciones corporativas. Esta basada en la experiencia obtenida en el desarrollo de este tipo de aplicaciones (comunmente denominadas Intranets o Extranets), y la finalidad es transmitir los conocimientos adquiridos para que el desarrollo de futuras aplicaciones sobre esta plataforma resulte más simple.

También deseo, con esta tesis, presentar un modelo de esquema a seguir para el desarrollo de futuras implementaciones, es decir, que se podría tomar el producto final de este trabajo como una base a ser adaptada a cualquier problemática de negocios.

Contenidos:

I Introducción

Una Intranet es un sistema que puede ser desarrollado con cualquier finalidad, con la particularidad de utilizar herramientas y tecnologías propias de internet. A continuación se describirán los elementos que componen estas tecnologías y las herramientas a utilizar en el presente trabajo.

Componentes de una Intranet

Podemos pensar que una intranet es una aplicación del tipo cliente-servidor, con la particularidad de que el servidor se encuentra particionado en varias sub aplicaciones que pueden correr en una o varias maquinas (esto es lo que comunmente se denomina tecnologia de 3 capas). Así, del lado del servidor podemos ubicar los siguientes elementos:

- Web Server: es el componente que se encarga de recibir los pedidos que realizan las aplicaciones clientes y devolver las páginas con los resultados. Si comparamos una aplicación tradicional con una intranet, podríamos decir que el web server se ocupa de la interfaz entre el servidor de aplicaciones y el cliente, este resuelve los llamados y el protocolo que utiliza son las direcciones HTTP recibidas de los browsers y las páginas html que envía como resultado de las llamadas.

- Aplicaciones: son los componentes que dialogan con el web server, resolviendo todas las cuestiones lógicas referidas a la problemática de negocio y siempre le devuelven al Web Server una página HTML. Existen distintos tipos de aplicaciones en las intranets actuales, entre las más conocidas y utilizadas podemos mencionar:
  - ✓ CGI's: que son aplicaciones tradicionales con la particularidad de poder dialogar con el Web Server, para enviar y recibir información, pero son aplicaciones totalmente aisladas de estos, y
  - ✓ ASP's: consisten en bloques de scripting insertados en las páginas que conforman el sitio o aplicación, genéricamente hablando. Estas se diferencian de las anteriores porque son ejecutados por el mismo Web Server, en vez de ejecutarse como una aplicación aparte. Más adelante explicaré con más nivel de detalle en que consisten y cuales son sus ventajas, ya que la intranet que describo utiliza este tipo de aplicación.

- Base de Datos: es un tercer elemento del lado del servidor, donde se almacenarán los datos del sistema y muchas veces también parte de la lógica del negocio.

- Páginas HTML: son los bloques de información que circulan entre el Web Server y el Browser, y en concreto, no son más que simples archivos de texto con tags embebidos entre el texto que podría ser leído en forma natural, pero que le agregan características de formato. Podemos decir que son el protocolo en el que dialogan ambas aplicaciones. Cabe aclarar que dentro de las tecnologías que se manejan a través de Internet, no solo existe este tipo de protocolo (el que conocemos como HTTP o hiper text transfer protocol), tambien existen los protocolos FTP (file transfer protocol) y Gopher, pero la herramienta que vamos a describir solo utiliza HTTP para el envio de las direcciones de paginas que se solicitan y HTML como lenguaje en el que dichas paginas estan escritas. Las páginas pueden

imaginarse como los templates de la interfaz de una aplicación tradicional, con la diferencia de que se desarrollan en un lenguaje standard.

En resúmen, del lado del servidor tenemos distintos elementos atendiendo las distintas tareas que este debe abarcar:

Interfaz con la aplicación cliente, atendida por el Web Server.
Templates de la aplicación, utilizando las páginas HTML.
Motor de la aplicación, en nuestro caso localizado en scripts de las ASP's.
Almacenamiento y administración de la información contenida, en la Base de Datos.

Del lado del cliente solo consideramos un elemento, el ya mencionado Browser, que consiste en una aplicación que puede interpretar las páginas HTML para mostrarlas de un modo amigable al usuario. Esta aplicación no tiene grandes cualidades en cuanto a manejo de lógica incluída en la página, excepto por el uso de elementos adicionales, como lo son el intérprete de Javascript que la mayoría de los browsers conocidos traen incorporado, la posibilidad de que el Web Server envíe aplicaciones pequeñas que puedan ser ejecutadas en la máquina que contiene el browser (applets), etc. Todos estos elementos, si bien están relacionados con el browser, no son nativos, y una aplicación intranet ideal debería enviar páginas con HTML puro para que su uso sea 100% eficiente y compatible con todas las versiones y diseños de browsers en circulacion. En nuestro caso, se incluyeron pequeños scripts de JavaScript para ser ejecutados en el browser con la finalidad de minimizar el uso de las conexiones entre el Browser y el Web Server en los casos de validaciones mínimas que no requieren información que no se encuentre contenida en la página.

Herramientas a utilizar

Cada uno de los elementos que mencioné en el apartado anterior se relaciona con una herramienta determinada. A continuación vamos a mencionar las herramientas elegidas para el desarrollo de la intranet que utilice como base para este trabajo. Existen muchas otras herramientas para cubrir los distintos aspectos, pero estas se relacionan muy bien entre sí, tarea bastante complicada si consideramos que las tecnologías utilizadas son muy nuevas y todavía no hay muchos estándares totalmente definidos. Asi tenemos:

<u>En el server:</u>

- NT Server 4.0 o Windows 2000: como sistema operativo,
- Internet Information Server 3.0, 4.0 o 5.0: como Web Server,
- Active Server Pages: como servidor de aplicaciones,
- SQL Server 7.0: como base de datos y repositorio de la lógica de negocio del sistema a desarrollar.

<u>En el puesto de desarrollo:</u>

- NT WorkStation o Windows 2000 professional: como sistema operativo,
- Netscape Communicator 4.61: como browser de páginas web (esta aplicación también se instalará en los equipos de los usuarios de la intranet). Podria utilizarse tambien el Internet Explorer, pero este browser tiene caracteristicas propias de funcionamiento y no reconoce algunos de los estandares adoptados en los lenguajes que utilizamos (HTML y JavaScript),

inclusive sus mayores puntos de falla son los que se relacionan con el pedido de paginas '.asp', aun cuando esta tecnologia fue creada por el mismo fabricante.

- NoteTab Pro 4.61: como editor de las páginas que conforman el sitio y la aplicación. Entre otras muchas facilidades, este editor tiene control de ediciones incorporado, con lo cual no se necesita de una herramienta adicional para manejo de versiones y edicion compartida de archivos.
- SQL Client Tools: para acceder y administrar el SQL Server desde los puestos de desarrollo.

En el puesto de acceso al sistema:

- Cualquier equipo donde se pueda instalar un browser, con cualquier sistema operativo.
- Netscape Navigator o Internet Explorer: como browser de paginas Web.
- MS Office: basicamente Word y Excel, ya que se pueden realizar macros en estas herramientas para complementar algunas tareas que no se pueden realizar a traves del browser, principalmente las orientadas a manejo de impresion de pagina (cortes de pagina, margenes, etc.).
- Acrobat Reader: para leer documentos '.pdf', en caso de que exista la necesidad de publicar documentos en este formato (se utiliza para asegurar la no modificacion de los mismos por parte del usuario).

Intranets frente a Aplicaciones Tradicionales

En relación a las aplicaciones mas comunmente usadas, podemos mencionar las siguientes ventajas a favor de las intranets:

- ✓ Gran parte de lo que requiere una aplicación tradicional ya está resuelto: la conectividad entre cliente y servidor, la división de tareas de cada módulo, la interfaz de la aplicación, la aplicación cliente etc.

- ✓ El cliente es muy simple y estándar, no depende de los equipos que se utilicen del lado del cliente. Esto es de suma utilidad en empresas con muchos años de trabajo, donde la diversidad de equipos en funcionamiento dificulta la implementación de nuevos sistemas.

- ✓ La posibilidad de manejar todo tipo de información desde una sola aplicación, ya que desde una intranet podemos acceder a datos de una base, documentos, imágenes, planillas, etc., y la manera de acceder es la misma tanto para datos estructurados como para datos no estructurados.

- ✓ Permite el uso de prototipos, una etapa del desarrollo de sistemas muy importante, pero que hasta hoy era muy tediosa de implementar, debido a que requeria un esfuerzo inicial extra y no siempre se asemejaba al producto final. En una intranet, el uso del prototipo nos permite acortar el tiempo de desarrollo, pudiendo discutir con el usuario la funcionalidad a partir de ejemplos y no desde lo abstracto de una idea.

Como desventajas, podemos mencionar las siguientes:

- ✓ Al ser muy nuevas las tecnologías involucradas, los productos que se utilizan no están lo suficientemente probados, esto dificulta el desarrollo, debido a que muchas veces los

famosos bugs se encuentran del lado del software de base, lo que los transforma en difíciles de localizar y obviamente imposibles de corregir. Con el tiempo nos acostumbramos a esto y aprendemos a buscar otros medios para obtener lo que el soft de base no llega a cubrir correctamente (y como todo lo malo tiene un lado bueno, finalmente llegamos a desarrollar soft de base que nos permite darle capas de seguridad y robustez adicionales a la aplicacion!).

- ✓ No hay demasiados desarrolladores que estén aplicando estas técnicas actualmente en nuestro país, esto hace que quienes las usamos no tengamos demasiadas referencias o consultores cuando se presenta algun problema. Ademas, no utilizamos estas herramientas tal como sus desarrolladores aconsejan (pues consideramos haber hallado mecanismos más eficientes para el desarrollo de aplicaciones), con lo cual, ni quienes desarrollan las herramientas pueden entender, muchas veces, cuando se les reporta un bug, como es que llegamos a esa situación!.

También consideramos importante enumerar las ventajas de las ASP's respecto de los CGI's, justificando así el uso de las primeras para el desarrollo de este trabajo:

- ✓ Son fáciles de implementar (debido a que simplemente se escribe el código en las páginas, este es interpretado, no requiere compilación, etc.)

- ✓ Su procesamiento lo realiza directamente el Web Server, no se debe acceder a una aplicación adicional (con el costo que esto requiere, los tiempos, la conectividad, etc.)

- ✓ Los lenguajes de scripting utilizados para escribir aplicaciones en este ambiente son muy simples, no se requiere una gran experiencia en VBasic o Java para desarrollar.

II Especificaciones

Para comenzar, estos son los requerimientos basicos para generar el ambiente necesario para el desarrollo de aplicaciones:

Hardware:

- Server: PC Pentium II o superior, con 96 MB RAM (mínimo) y 2 GB disco (mínimo).
- Puestos de Desarrollo: PC Pentium o superior, con 32 MB RAM (mínimo).
- Puestos clientes: cualquier equipo PC, Mac o Unix, donde se pueda instalar un browser.

Software Server:

- Windows 2000 Server (o NT Server y WorkStation 4.0. con Option Pack 4.0 y Service Pack 4)
- SQL Server.

Software Puesto de Desarrollo y Cliente:

- Netscape Navigator 4.61 o superior.
- NoteTab Pro 4.61
- SQL Client Tools.

La elección de IIS, ASP's y SQL Server fueron determinadas originalmente por la elección de NT, debido a que estas están mejor integradas con el sistema operativo que cualquier otra. Tanto el IIS como el SQL Server con Windows NT pueden ver el mismo esquema de seguridad, lo que me permite definir solo una vez a los usuarios, comparten el Event Viewer, herramienta desde donde podemos controlar cualquier problema o mensaje que envien estas aplicaciones, y ambos pueden funcionar como servicios del server.

El haber elegido Netscape Navigator como browser se debe exclusivamente a la experiencia en el uso de los dos mas reconocidos en el mercado (este y el Internet Explorer). Netscape respeta mejor los estándares HTML, mientras que el Explorer cuenta con varios elementos totalmente propietarios para realizar funciones que el HTML estándar cubre de un modo mucho más simple. Además, Netscape tiene una mejor velocidad de respuesta para el usuario, soporta mejor el estándar Javascript y falla mucho menos en la ejecución de las ASP's que el Internet Explorer (esto es algo difícil de entender, primero cuando este úiltimo pertenece a los mismos fabricantes que las ASP's y segundo porque se supone que el browser no debería tener nada que ver con estas, ya que se ejecutan en el servidor. Lo cierto es que mientras en el Internet Explorer muchas de las páginas desarrolladas fallaban sin explicación alguna, en Netscape funcionaban correctamente!).

Por último, NoteTab fue el mejor editor de páginas que hallamos para simplificar la edición de las páginas, ya que me permite editar muchas páginas a la vez, realizar búsquedas entre todas estas, definir grupos de proyectos, y otras configuraciones que simplifican el desarrollo. Es importante destacar su similitud, en aspectos generales, al BBEdit, editor de paginas que solo cuenta con una version para Macintosh, y esto es una verdadera lastima, ya que

es el editor con el que todo desarrollador soño alguna vez!. Como contrapartida de esta herramienta podríamos haber utilizado el Visual Interdev, pero no me resultó práctico, ya que genera demasiado código innecesario y poco eficiente. Solo lo usamos para aprender que eran las ASP's, pero no lo recomendamos para el desarrollo de aplicaciones productivas.

III Herramientas

En este capítulo intentaremos describir las herramientas antes mencionadas, detallando sus características, parámetros de instalación y configuración y algunas recomendaciones basadas en la experiencia con cada una de ellas.

Para esto vamos a mantener la clasificación del capítulo anterior, es decir, vamos a describir dos ambientes: el servidor y el puesto de desarrollo (que podemos considerarlo como un super cliente, ya que contiene la aplicación que se instalará luego en cada uno de los equipos de quienes accedan a la intranet). Entonces tendremos:

Server	Puesto de Desarrollo
Windows 2000 Server o NT Server	Windows 2000 professional o NT WorkStation
Option Pack 4.0 (para NT)	Netscape Communicator
Service Pack 4 (para NT)	NoteTab Pro
SQL Server 7.0	SQL Client Tools

Windows 2000 Server y Professional (o NT Server y Workstation) son los sistemas operativos a utilizar tanto en el servidor como en los puestos de desarrollo, conviene instalarlo del modo más estándar posible (si es posible no habilitar el uso de dominios, a menos que sea realmente necesario. Esto ayuda a hacer más fácil el mantenimiento del servidor, y a encontrarnos con menor cantidad de conflictos en el futuro!). Algunos de los servicios que vamos a necesitar del SO son:

- DHCP Server: considerando que se recomienda el uso del protocolo TCP/IP para la configuración de la red, y es conveniente la asignación de direcciones dinámicas, esto reduce las tareas de asignación, administración y mantenimiento de cada uno de los equipos.

- DNS Server: para definir nombres legibles que el Web Server pueda interpretar como direcciones IP. Esto se puede realizar si se cuenta con este servicio instalado y configurado, que tiene como objetivo traducir las direcciones en lenguaje casi natural a direcciones IP. Por ejemplo, una direccion convencional puede ser 'www.netscape.com' y su direccion ip podria ser 192.168.5.5.

- IIS Admin Service: este se instala desde el Option Pack, al instalar el IIS Service en NT o como parte de la instalacion estandar desde Windows 2000 Server.

- World Wide Web Service: igual que el anterior se instala desde el Option Pack, y es el servicio que habilita el IIS.

El Option Pack es un agregado del NT, que contiene un conjunto de aplicaciones extras, la mayoría orientadas a Internet, y la intención de Microsoft con esta implementación es acortar las diferencias entre NT 4.0 y Windows 2000. Este producto agrega a la versión actual de NT muchas de las facilidades que trae incorporado Windows 2000. Entre ellas está incluído el IIS 4.0, quien a su vez contiene el motor de ASP's. Es recomendable, si se va a desarrollar una nueva aplicacion, considerar la instalacion de Windows 2000 directamente, es mucho mas simple que

NT + Spack + Opack y ya hay suficientes empresas utilizandolo como para arriesgarse a intentarlo!.

Los Service Pack's son algo clásico en los productos Microsoft, y son un conjunto de 'parches' o rutinas extra que se supone solucionan problemas reportados por los usuarios. De todos modos,los usuarios de aplicaciones sabemos que cada mejora o correccion en software siempre trae nuevos problemas, es decir, que cada nuevo service pack siempre traera nuevos errores a las aplicaciones que estemos usando.

SQL Server es un motor de bases de datos, contiene muchas de las herramientas con las que cuentan las grandes bases de datos (como Oracle o Sybase), y la política de Microsoft es competir con aquellas. Básicamente, aquí se pueden definir bases de datos, diseñar tablas, vistas, procedimientos almacenados, etc. Las SQL Client Tools forman parte de esta aplicación y permiten acceder a la administración y a los datos almacenados en SQL.

Netscape Communicator es el browser que vamos a utlizar como cliente de la aplicación. Tiene varias aplicaciones incluídas, un visualizador de páginas, un cliente de e-mail, una agenda, un editor visual de páginas HTML, una aplicación para comunicarse On-Line con otros usuarios de la misma aplicacion en Internet, y otras herramientas para navegar por Internet. Para el desarrollo de la Intranet basta con el visualizador de páginas, pero quisieramos mencionar que conviene implementar el cliente de mail como corporativo, para simplificar la capacitación de los usuarios respecto de distintas aplicaciones, considerando que la interfaz es similar para todos los productos Netscape; ademas, dependiendo de los usos de la intranet, se podrian desarollar caracteristicas especiales para comunicar a distintos sectores de la empresa, con clientes, proveedores, etc. desde el mismo sistema, si se cuenta con el cliente de mail incorporado al browser.

NoteTab Pro es un editor de páginas de texto, orientado a páginas HTML. No es visual, solo muestra el texto de cada pagina. Se eligio trabajar con editores no visuales para desarrollar ASP´s debido a que todavía no hay ningún editor visual que interprete los scripts embebidos; en el mejor de los casos, los editores que 'intentan' interpretarlos, agregan tags cuando no entienden algo, esto es muy perjudicial, ya que ciertas cosas que funcionan en el browser no funcionan en estos editores, y lo más probable es que terminen arruinando el código de scripting que uno haya escrito. Con esto, finalmente es mas eficaz desarrollar en un editor de paginas de texto y probar la aplicación directamente a traves del browser, evitando perdidas de tiempo innecesarias.

A continuación se detallan las características de instalación y configuración de cada uno de estos productos, junto con algunas recomendaciones. No vamos a detallar lo que sería la instalación del sistema operativo, solo vamos a describir los servicios necesarios para la implementación de la intranet.

Internet Information Server

IIS es un Web Server, esto es, un servidor que administra el acceso a páginas HTML. Este Web Server, en particular, trae incorporados un conjunto de librerías adicionales, que permiten la interpretación de scripting incluído en las páginas (esto es lo que denominamos ASP

´s) y la conexión con una base de datos (por medio de lo que describiremos en el apartado de lenguajes como ADO, el cual consiste en un conjunto reducido de sentencias necesarias para enviar y recibir datos de la base de datos). Esta aplicación funciona como un servicio dentro de NT 0 W2000 Server.

Instalación

Desde el NT 4.0 no se recomienda instalar el IIS durante la instalación del NT Server, ya que en ese momento no se pueden configurar sus características de seguridad (el IIS debe generar la carpeta donde buscará las páginas del site en una partición NTFS, y esto no es configurable desde la instalación de NT). Además, es recomendable instalar directamente la versión 4.0 que viene incluída en el Option Pack 4, para no tener luego problemas en el modo de instalación del producto.

Desde W2000 se instalara directamente durante la instalacion basica del producto, sin necesidad de pasos posteriores, ya que este sistema operativo formatea directamente el disco en formato NTFS.

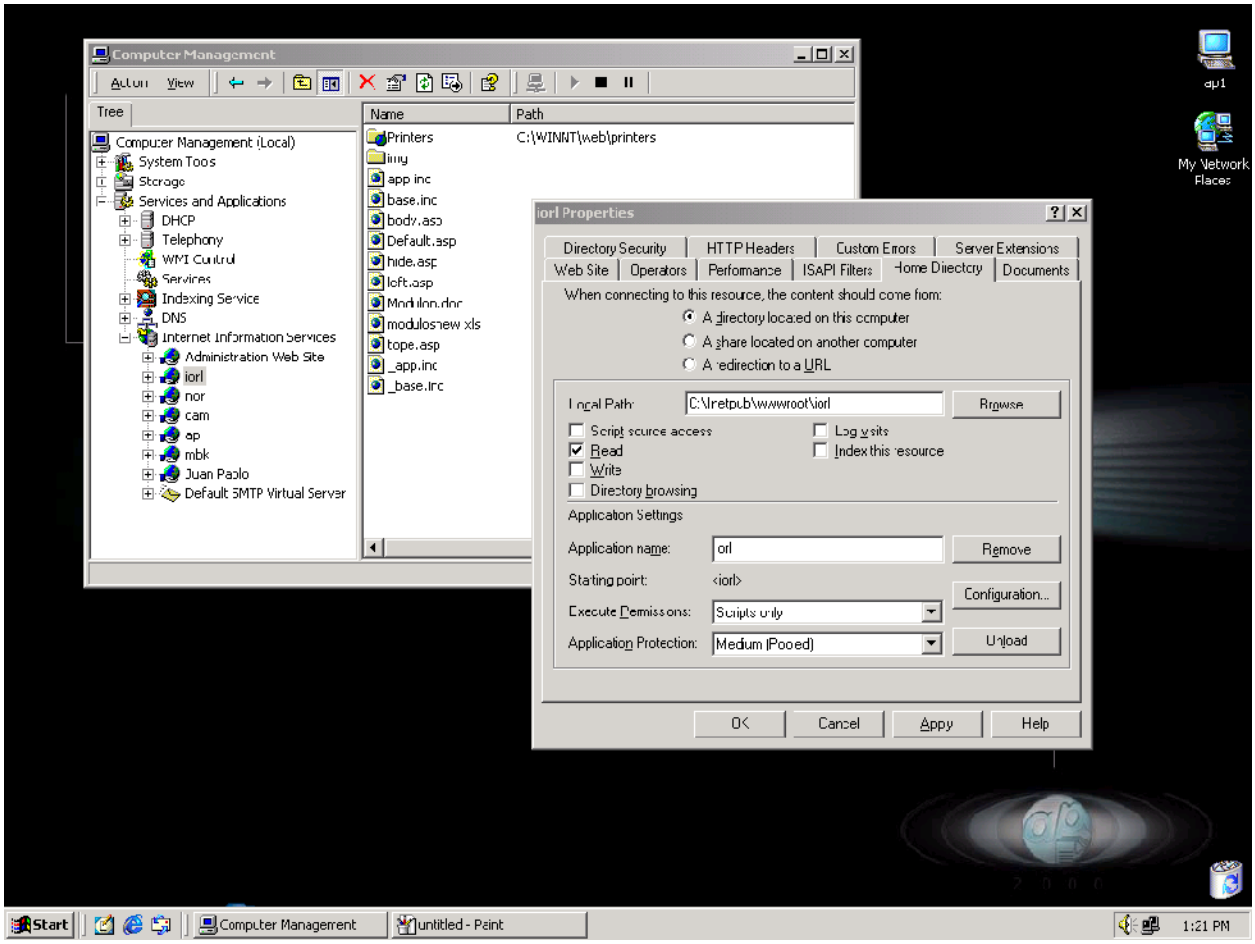
Al instalarse, IIS configura 2 WebSites, uno para la administración de este servicio y otro con la documentación del producto, al que le asigna la característica de default.

Configuración

Lo primero que deberíamos hacer para configurar nuestro propio Web es crear la carpeta donde vamos a poner las páginas html que van a conformar el sitio, que deberá estar dentro del path: 'partición NTFS:\inetpub\wwwroot', ya que aquí es donde el IIS almacena físicamente sus sitios. De todos modos se podrían ubicar en cualquier otro sitio, luego veremos como configurar el IIS para indicarle que la carpeta donde se encuentran las paginas estan en un directorio diferente al default.

La herramienta para administrar el IIS se denomina MMC (Microsoft Management Console) y fue diseñada para concentrar, a partir de W2000, las herramientas de administración de todos los productos Microsoft (desde N solo se pueden administrar desde ella el IIS y el SQL Server). La interfaz es similar a la usada en el NT Explorer, cuenta con una ventana donde incluye el árbol con las aplicaciones que puede administrar, que se puede expandir hasta ver en una segunda ventana los archivos que componen la aplicación (en el caso del IIS llegamos a ver las páginas que conforman el site).

Ahora, para definir nuestro WebSite, debemos ir, en NT, a Start\Programs\Windows NT Option Pack\Internet Information Server. Desde aquí se accede a la MMC, desde donde podremos configurar los parámetros necesarios para que las páginas puedan ser mostradas a través del IIS. En W2000 accedemos a la MMC desde el icono de la maquina, en el desktop, seleccionando la opcion Manage del menu que se activa al presionar la tecla derecha del mouse sobre este icono.



Para definir un nuevo Web, debemos presionar el botón derecho del mouse posicionados sobre alguno de los webs existentes (recordemos que el IIS siempre configura 2 durante su instalación, con lo cual siempre vamos a encontrar alguno ya configurado), entonces aparecerá un menú con los siguientes items:

- ✓ Explore: realiza una búsqueda en el disco del equipo para actualizar la información que muestra en la ventana de la derecha.
- ✓ Open: abre una nueva ventana del NT-Explorer a partir de la carpeta de inicio del site.
- ✓ Browse: desde aquí podemos ver las páginas en un browser, sin necesidad de saber cual es la página principal de la aplicación.
- ✓ Start: esta opción nos permite inicializar la aplicación.
- ✓ Stop: desde aquí podemos detener la ejecución de la aplicación asociada al sitio.
- ✓ Pause: sirve para detener la ejecución de la aplicación por un momento para luego retomarla en el mismo punto donde fue detenida.
- ✓ New: desde aquí podemos definir un nuevo sitio.
- ✓ Delete: podemos borrar un sitio (esto no significa que las páginas que lo conforman sean borradas del disco, solo estamos eliminando el link de una carpeta a un sitio web.
- ✓ Refresh: actualiza los datos de la ventana de la derecha.
- ✓ Properties: esta opción nos permitirá configurar el sitio.

Del menú anterior, debemos seleccionar la opción 'Properties', que nos abrirá una ventana con las siguientes solapas o secciones:

- Web Site: desde aquí se definen los siguientes parámetros:
  - ✓ Descripción (nombre del sitio),
  - ✓ IP Address,
  - ✓ TCP Port,
  - ✓ Conecciones (límites en cuanto a cantidad o tiempo de espera para cada intento),
  - ✓ Habilitar o no el Logging de la aplicación (no recomendamos habilitar este checkbox, debido a que consume demasiados recursos del equipo donde está instalado, afectando seriamente la performance del servidor. En nuestro caso, lo que hicimos fue incluir un archivo de log dentro de la aplicación, donde almacenar los eventos que consideramos importantes para realizar nuestros propios controles y estadísticas).

- Operators: aquí se indica que usuarios tienen acceso a la administración de los sitios web que se encuentran en el servidor (el IIS otorga este privilegio al administrador del servidor, y salvo excepciones bastaría con esto).

- Performance: donde se indican ciertos parámetros que hacen a la velocidad de respuesta del servidor.

- ISAPI Filters:

- Home Directory: donde se indican las siguientes características:
  - ✓ Como interpretar el path de inicio de la aplicación o del sitio (esto es: si es una carpeta ubicada en el mismo servidor, en otra máquina, o es una dirección de internet),
  - ✓ Path local (este ítem cuenta con un botón llamado 'Browse', que permite navegar por las maquinas accesibles de la red para localizar donde se encuentra la carpeta donde almacenamos las páginas de nuestro sitio),
  - ✓ Permisos de Acceso (read y write, se recomienda habilitar solo el primero),
  - ✓ Control de Contenidos:
    - Log access,
    - Directory browsing (no recomendamos habilitarla por cuestiones de seguridad, por ejemplo, si tenemos páginas que solo queremos que sean vistas por determinados usuarios de la aplicación, esto no se podría administrar si tenemos esta opción encendida),
    - Index this directory (me permite indexar los contenidos de las páginas que se encuentran por debajo de la carpeta indicada en 'Local Path',
    - Front Page Web (no recomendamos habilitarlo y tampoco se acostumbra instalar las Front Page Server Extensions, que vienen incluídas en el Service Pack).
  - ✓ Parámetros de la aplicación:
    - Nombre de la aplicación (no necesariamente debe ser el mismo que el del Web, aunque es recomendable, para asociarlos en el futuro. Con este nombre de aplicación, por ejemplo, podremos saber en SQL Server cuando algún usuario accedió a la base de datos desde el web.
    - Permite indicar si queremos que se ejecute utilizando un espacio separado en memoria, es decir, como un proceso aislado (no recomendamos esta práctica, ya que una de las principales ventajas de las ASP's es justamente el funcionar dentro del

IIS, optimizando los recursos y mejorando la performance de la aplicación, ya que ambas comparten la información.

- Permisos de scripting y ejecución (recomendamos marcar solo el de scripting, ya que el de ejecución no es necesario para utilizar las ASP's y al habilitarlo estaríamos permitiendo que el web ejecute cualquier aplicación que se tenga disponible, y esto es una vía libre para futuros ataques al sitio).

Dentro de estos parámetros hay un botón llamado 'Configuration', que abre una ventana con tres solapas:

- App Mappings:
- App Options: desde donde podemos indicar:
  - Habilitación del manejo de sesiones (esto es indispensable para desarrollar aplicaciones multiusuarios, ya que nos permite desarrollar sin preocuparnos por el manejo de elementos propios para cada usuario).
  - Habilitar buffering (esto solo es util si nuestras páginas van a tener elementos genericos o va a ser un web estático, en el caso de aplicaciones dinámicas no es útil, ya que no nos permitiría ver la última versión de la página si la misma fue solicitada un momento antes).
  - Habilitar Paths emparentados (es decir, manejar direcciones relativas al punto de inicio del web), conviene habilitar esta opción.
  - Definir el lenguaje default, que puede ser VBScript o JScript, en nuestro caso utilizaremos el segundo.
  - Indicar el tiempo máximo de espera para la ejecución de un script (debería ser suficiente con indicarle 60 segundos, pero siempre debemos considerar que puede haber procesos que demoren algo más, como por ejemplo Stored Procedures almacenadas en la base de datos, con lo cual conviene seleccionar el proceso más lento y a su tiempo de ejecución sumarle unos 10 segundos más para indicar el valor de este parámetro. Este tiempo siempre sera un estimado, pero lo importante es definir un tiempo razonable de espera, para que la aplicacion no siga procesando indefinidamente, por ejemplo, un loop con un corte de control equivocado.
- App Debbugging: contiene parametros para permitirle al IIS enviar mensajes de debug al server o al browser. Acostumbramos no habilitar esta opcion, ya que en tiempo de ejecucion consume un tiempo razonable y los mensajes enviados no son demasiado claros. Ademas, en la primer version del IIS no existia esta opcion, motivo por el cual nos acostumbramos a mostrar en las paginas las variables que queremos controlar o puntos de control de ejecucion necesarios para solucionar problemas en el codigo, por esto no consideramos necesario contar con esta opcion.

- Documents: permite establecer cual es el documento que se mostrara como default si en la direccion de acceso al sitio no se especifica el nombre de la pagina.

- Directory Security: desde aquí se definen las restricciones de acceso a los usuarios del web. Estos permisos estan separados en tres niveles:
  - Accesos Anónimos y Control de Autenticación: en este nivel tenemos tres maneras diferentes para permitir accesos:
    - ✓ Permitir accesos anónimos (para estos accesos, IIS utiliza una cuenta de NT creada por él mismo durante su instalación),

- ✓ Autenticación básica (IIS solicitará usuario y password cuando los accesos anónimos no estén permitidos y cuando las páginas se encuentren en una partición NTFS. En ambos casos el nombre y la password deben ser definidos en el esquema de seguridad de NT),
- ✓ Windows NT Challenge/Response (solicitará password cuando no se permitan accesos anónimos y cuando el usuario que esté intentando acceder no esté conectado a la red NT donde se encuentra instalado IIS).
- Comunicaciones Seguras,
- Restricciones a nivel de direcciones IP y nombres de dominios.

- HTTP Headers: permite indicar caracteristicas del contenido del sitio, como por ejemplo: tiempo de actualizacion de las paginas, clasificacion de contenidos, paginas con caracteristicas especiales, etc..
- Custom Errors: desde esta opcion se pueden relacionar errores estandares de HTTP con paginas personalizadas que expliquen en mayor detalle en que consisten los errores ocurridos. Existe un conjunto predefinido de estas paginas.

Para resumir todo lo detallado anteriormente podemos enumerar los pasos a seguir para configurar una aplicación:

- Crear una carpeta con al menos una página inicial en:
  'partición NTFS:\inetpub\wwwroot\carpeta'
- Ir al IIS Management Console.
- Definir un nuevo Web Site (desde el ícono del server o desde el ícono de otro web existente presionar botón derecho del mouse, seleccionar la opción 'New Site'); indicar nombre, dirección IP y permisos de acceso.
- Sobre el ícono del nuevo site presionar el botón derecho del mouse y seleccionar 'Properties'.
- Ir a la solapa 'Home Directory', presionar el botón 'Configuration' y modificar el timeout de las sesiones y de ejecución de scripts, y el lenguaje de scripting a utilizar (ya que por default asume VBScript).
- Ir a la solapa 'Directory Security' y realizar las restricciones deseadas (en particular es preferible deshabilitar la opción que permite accesos anónimos).

Siguiendo estos pasos, el Web Site debería quedar configurado y listo para ser accedido.

Recomendaciones para asignar privilegios a las ASP's

- Permitir solo lectura a las carpetas que contengan páginas html o imágenes, ya que estas no requieren el procesamiento del motor de las ASP's (de esta manera se resuelven y envian mas rápido al cliente).

- Permitir ejecución de scripts solo a las páginas que los contengan (sino el IIS no las podrá enviar resueltas al browser!).

- Solo permitir ejecución para carpetas que contengan aplicaciones o componentes que requieran la ejecución de una aplicación externa al Web Server, y en lo posible no utilizar aplicaciones externas, por seguridad no conviene habilitar esta opción.

- Solo permitir escritura en carpetas donde la aplicación deba escribir en archivos del file system de NT (por ejemplo, si quiero llevar un log propio de operaciones en un archivo de texto).

- No permitir, en lo posible, la opción de directory browsing, ya que un cliente puede recorrer el esquema del site y eventualmente consultar páginas que no pueden ser accedidas desde los links de la aplicación (por ejemplo, librerias de scripting).

Seguridad

Es conveniente utilizar SSL (Secure Sockets Layer) para asegurar la información que circula entre el Web Server y el browser, ya que esta herramienta se encarga de encriptar la información que se transmite, funcionando de la siguiente manera: cuando se inicializa una sesión segura se establece una comunicación entre el Web Server y el browser; el browser recupera el id del server y lo verifica con el certificado del mismo; si este es válido, emite un string de encripción y da comienzo a la sesión SSL.

Algo que debemos permitir en los browsers es la aceptación de cookies, ya que las ASP´s administran las sesiones mediante el uso de estas.

Consideraciones varias

- Setear a 0 el tamaño del cache para ASP's, para evitar problemas del tipo: modifique una página y no puedo ver la actualización a través del browser!.

- Setear en off la opción de scripting debugging en sites de producción (en general, recomendamos no utilizar esta característica, al menos en las versiones conocidas del IIS, ya que no brindan gran utilidad y lentifican enormemente la ejecución de las ASP's).

- Indicar que las aplicaciones que todavía están en etapa de desarrollo corran en espacio separado de memoria, para poder detectar posibles fallas en la utilización de recursos de manera externa al IIS.

Domain Name Service

Este es un servicio que forma parte del NT o W2000 server, se puede instalar incialmente o agregandolo como componente Add-On. Lo importante a considerar es que si en una misma red se instalan mas de un DNS pueden generarse conflictos en el caso de que cada uno de estos no este declarado en los demas.

Instalación

Si se instala junto con el servidor no hay que realizar ningun paso adicional, simplemente seleccionando los servicios de red vamos a encontrar un servicio denominado Domain Name Service.

Caso contrario, puede agregarse posteriormente, para eso se deben seguir los siguientes pasos:

- √ Insertar el CD de instalacion del server
- √ Seleccionar 'Install Add-On Components'
- √ Elegir 'Networking Services'
- √ Instalar
- √ En el caso de NT 4.0 es necesario bootear el equipo, para W2000 no es requerido.

Configuración

La configuracion minima requerida para incializar la utilizacion del DNS es la siguiente:

- √ Ir al administrador del equipo (Start\Programs\Administrative Tools\Computer Managemet),
- √ Seleccionar 'Services and Applications'
- √ Seleccionar 'DNS'
- √ Abrir: nombre equipo\Forward Lookup Zones
- √ Crear una nueva Zona (al presionar la tecla derecha del mouse sobre el item antes mencionado aparecera esta opcion)
    - Indicarle:
        - Tipo: Standard Primary
        - Nombre: nombre del server (ej.: ap)
        - Crear nuevo archivo con este nombre
- √ Ingresar a esta nueva zona creada y generar un nuevo Host (del mismo modo que se creo la zona)
    - Indicarle:
        - nombre del dominio (ej.: misitio)
        - IP Address asociada (ej.: 192.168.1.1)

De este modo, para ingresar al sitio que tenga instalado en este equipo, podre hacerlo ingresando la direccion 192.168.1.1 o el nombre misitio.ap

SQL Server

SQL Server es un motor de bases de datos, donde vamos a almacenar los datos contenidos en la aplicación, así como la mayor parte de la lógica del negocio. Esto último también podría declararse del lado de las ASP´s, pero en su momento se decidió así porque, si en algún momento, hubiera que cambiar alguna herramienta de las que conforman el sistema, lo último en

modificarse sería la base de datos, debido a que en principio es el software de base más probado de todos los que se mencionan en este trabajo (eso lo vuelve más robusto y estable que los demás componentes) y además la migración de los datos es la tarea más tediosa en el cambio de versión de un sistema, con lo cual, seguramente se eligiría cambiar cualquier otra de las herramientas antes que el SQL (en un principio no les teníamos mucha fé a las ASP´s, y pensabamos que si a futuro salía algo mejor podríamos migrar a ese algo y el trabajo estuvo orientado a dejar en estas la menor cantidad de código posible).

Instalación

La información que va a solicitar el wizard de instalación es la siguiente:

- ✓ Nombre del server NT o W2000,
- ✓ Cuenta del server que usará SQL al inicializarse,
- ✓ El modelo de seguridad a utilizar:
  - Standard: propio del SQL,
  - Windows NT integrado: reconociendo el login de NT,
  - Mixto: combinando ambos modelos,
- ✓ Ubicación y tamaño de la base de datos master (donde se guardarán referencias de todos los objetos que conforman las distintas bases que contendrá el SQL),
- ✓ El set de caracteres a utilizar (de esto dependerá la organización de los datos para las consultas, búsquedas y demás),
- ✓ Caracteristicas de ordenamiento,
- ✓ Protocolo de red que utilizará SQL (recomendamos el uso de TCP/IP, que es el más standard, simple y eficiente),
- ✓ Número de licencias adquiridas o usuarios concurrentes que va a soportar el server SQL,
- ✓ Grupos de usuarios que tendrán acceso a la base de datos.

Los pasos necesarios para la instalación del SQL Server son:

- ❖ Del cd de instalación ejecutar setup.exe.
- ❖ Seleccionar un modo de licencia (esto es indicar el modo de contabilización de las licencias que fueron adquiridas para acceder a la base de datos, si fueron pactadas por conexión o por computadora, lo más usado es comprarlas por conexión, y para desarrollar una intranet bastaría con una sola licencia, ya que el cliente necesario es uno solo).
- ❖ Indicar en que unidad de disco y carpeta se instalará la base (se debe especificar la ubicación y el tamaño inicial del Master device, que es el archivo que contendrá todas las bases de datos del sistema).
- ❖ Seleccionar las siguientes opciones de configuración:
  - Character set: las opciones son Code Page 850 (multilingual) o Code Page 437 (US English).
  - Sort order: algunas alternativas son: case sensitive o insensitive, binary sort order, uppercase sorts first, etc.
  - Protocolo de red: se puede optar por Multi-Protocol, NWLink o TCP/IP Sockets.
  - AutoStart SQL Server cuando arranca el equipo.
  - AutoStart SQL Executive cuando arranca el equipo.

- Especificar una cuenta de usuario para acceder a la base.

Al finalizar la instalación, esta habrá generado un grupo de programas (que se deberá buscar en start\programs\Microsoft SQL Server 7.0) dentro de los cuales, los más utilizados son:

- Books on Line: son una excelente herramienta de consulta para aprender a administrar y desarrollar aplicaciones en SQL Server,
- Enterprise Manager: es la herramienta que nos permite generar y administrar las bases de datos contenidas en SQL Server, al igual que los objetos que las conforman (tablas, vistas, etc.),
- Import and Export Data: es un utilitario que permite enviar datos desde y hacia el SQL Server. Esta aplicación se encuentra incorporada al Enterprise Manager,
- Profiler: desde esta aplicación podemos ver que comandos están siendo enviados al SQL, que usuario los envia, si fueron ejecutados con éxito, etc.,
- Query Analyzer: este ambiente nos permite enviar comandos al SQL, podemos realizar simples consultas, crear objetos, etc., siempre en lenguaje Transact-SQL.

Configuración

Para poder acceder al SQL Server debemos abrir el Enterprise Manager y registrar el servidor. Los datos requeridos para este paso son: nombre del server, nombre de usuario y password. El nombre de usuario default es SA, y en principio no requiere password, aunque, considerando que este usuario tiene todos los permisos asignados, es altamente recomendable asignarle una password que en lo posible no sea facil de acceder.

Creación de una Base de Datos

Para crear una base de datos se necesitan definir los siguientes elementos:

- el device donde se va a almacenar la base (esto se requiere en SQL 6.5, SQL 7.0 ya no lo utiliza). Esto se puede realizar desde el Enterprise Manager/Manage/Database Devices, aqui se abrira una ventana donde se ingresa el nombre del device y el espacio en disco a utilizar. En un device se puede almacenar mas de una base definida por el usuario (tambien hay bases propias del SQL Server, que se almacenan en devices diferentes). Lo que hace esto, en realidad, es crear un archivo en el path indicado, con el nombre del device y le asigna el tamaño indicado. Como recomendacion, es mas sano definir de antemano el tamaño que puede llegar a ocupar la base, ya que, si bien este dato se puede modificar sobre la marcha, la informacion se mantiene mejor organizada si puede almacenarse en forma secuencial y en un solo lugar del disco.

- la base de datos, se genera a través de un wizard que se encuentra en el Enterprise Manager/Manage/Databases. Los datos que pide esta ventana son: nombre de la base de datos, nombre del device, tamaño asignado (que no puede ser mayor que el tamaño asignado al device) y el nombre del device donde se desea almacenar el log de transacciones. En SQL 7.0 basta con posicionarse sobre la carpeta Databases dentro del Enterprise Manager para que el wizard mencionado anteriormente aparezca, en este caso, basta con indicar el nombre de

la nueva base, ya que en esta nueva version toda la problemática referida al espacio a utilizar la resuelve directamente el SQL sobre el SO.

- las tablas, views, stored procedures, tipos de datos definidos por el usuario, defaults para ciertos campos, reglas, constraints, indices, triggers y claves foraneas. Estos objetos son los que finalmente se van a utilizar para definir el modelo de datos, asegurar su integridad y desarrollar la aplicación que los administre. Cada uno de ellos tiene su propia interfaz grafica para generarlos y administrarlos, en SQL 6.5 tambien desde el Enterprise Manager/Manage. En SQL 7.0 esas interfaces graficas fueron modificadas sustancialmente (y no siempre para simplificar la vida del pobre programador!), y el nuevo Enterprise Manager ya no nos permite consultar la informacion contenida en la base, ni ejecutar queries, sino que para eso existe otra aplicación llamada Query Analyzer, desde donde tambien se pueden crear y modificar los objetos arriba mencionados, utilizando el lenguaje Transact-SQL.

Recomendaciones

Crear siempre, como mínimo 2 bases de datos: una para desarrollo y otra para producción (esto si se va a trabajar con el mismo servidor en los dos ambientes); y una manera muy cómoda de trabajar es usar los datos de producción para el desarrollo (si son una gran cantidad bastará con tomar solo una parte).

Super NoteTab

Super Notetab es un editor de texto que encontre en un site que recomienda productos para el desarrollo de ASP´s (www.asp.com). Este editor se encuentra en la categoria de Shareware, es decir, que su costo es minimo (u$s 5), y se puede bajar una versión demo de www.notetab.com.

La caracteristica para mi mas importante de este editor fue el manejo de multiples páginas de un modo unificado, esto me permite realizar acciones de busqueda, reemplazo, etc. en todas las páginas a la vez, una posibilidad muy importante cuando una aplicación esta formada por un conjunto de páginas con codigo fuente.

Algo interesante es un manejo de proyectos, es decir, me permite agrupar varias páginas en un proyecto (o categoria); asi, puedo mantener varias versiones de un desarrollo, como ser, la última versión de produccion y la versión de desarrollo, y trabajar con cada conjunto por separado.

Otra caracteristica importante es que administra el uso de una página por otros usuarios. Esto es imprescindible cuando se trabaja en equipos, porque me asegura que no perdere mis modificaciones ni las de quien haya abierta la misma página (al intentar acceder a alguna de mis páginas abiertas me avisara que esta fue modificada y me preguntara si quiero actualizarla antes de comenzar a modificarla).

Por supuesto, cuenta con todas las caracteristicas de los editores de texto HTML que hay en el mercado: maneja el sistema de coloreo del texto de acuerdo a los tags que va

encontrando (esto es util si la página contiene puro HTML, en mi caso, muchas veces tuve que olvidarme de esta caracteristica, debido a que interpreta que los tags comienzan con el caracter '<' y terminan con '>', y cuando en mi codigo de scripting utilizaba estos simbolos, el editor cambiaba el color del codigo que encontraba adelante!. Aun asi, es una buena caracteristica), me permite acceder a un browser desde el mismo editor, puedo tener en mi barra de herramientas botones que al utilizarlos inserten en la página el HTML necesario para agregar determinados objetos, facilidades para seleccionar colores, etc.

Instalación

Instalar el NoteTab Pro solo requiere ejecutar ntp40.exe, donde se indican como unicos parametros el lugar donde ubicar los archivos en el disco y el nombre de icono de programa para el sistema operativo. En mi caso, tengo otros dos ejecutables con upgrades a esta instalacion, cuyo mecanismo de instalacion es identico a este.

Configuración

Esta aplicación tiene un area de configuraciones que se accede desde View\Options, en el menu principal del programa. Las caracteristicas que acostumbro dejar habilitadas son las siguientes:

View: indicar que solo muestre Toolbar y Statusbar. La primera para acceder mas rapido a las tareas que mas frecuentemente realizo; y la segunda porque uso muy frecuentemente el contador de lineas para corregir problemas de sintaxis (que el browser detecta e informa indicando justamente la linea dentro de la pagina en ejecucion).

General: los items que dejo encendidos son:

- √ Save Position and Size
- √ Reload Open Documents (para el caso en que alguien mas este trabajando sobre el mismo documento, de este modo se controla la edicion de un mismo archivo por varios usuarios)
- √ Load Favorites on Startup (ya que puedo indicarle grupos de paginas, seleccionando uno favorito, y hacer que cada vez que levanto el programa ya los edite)
- √ Display Prompts

Files: Indico los directorios donde posicionarme inicialmente para abrir y grabar documentos (c:\inetpub\wwwroot).

Documents: indicarle que no realice 'Word Wrap', ya que esto dificulta el entendimiento de las lineas de programa!.

Tools: solo indico que realice 'Find Word at Cursor'

TabBar: le indico que no muestre las extensiones de los archivos ni sus iconos, ya que esto me ocupa demasiado espacio en la linea de solapas de archivos editados.

Netscape Communicator

Netscape es un programa que se utiliza para ver páginas HTML. Comunmente estas aplicaciones reciben el nombre de browsers, y se utilizan para navegar por Internet o como interfaz para las intranets. Incluye las siguientes aplicaciones:

- Navigator: para visualizar páginas html,
- Messenger Mailbox: para administrar la casilla de correo,
- Collabra Discussion Groups: para consultar newsgroups,
- Page Composer: para diseñar páginas html,
- Conference: para establecer conferencias con otros usuarios,
- Calendar: es una aplicación de agenda,
- IBM Host On-Demand: para acceder a la aplicación central de una empresa.

Si bien todos los productos Microsoft recomiendan el uso del browser Internet Explorer, la practica me demostro que no es lo mejor, comence haciendo caso a los que 'supuestamente saben!', pero a medida que avanzaba en el desarrollo me encontraba con situaciones inexplicables, tags standard de HTML no reconocidos por este browser, la recomendacion de utilizar tags 'especiales' (es decir, propietarios!), manejo poco eficiente de formateos de letras, tamaños y demas (los conocidos styles.css), y lo mas desagradable era ver que estas caracteristicas solo funcionaban para Internet Explorer. Decidi hacer pruebas con Netscape y descubri que, para empezar, levantaba las páginas mucho mas rapido (esto se puede evaluar desde una intranet, donde la velocidad de bajado de páginas es practicamente constante, no como el caso de internet, donde lo último que afecta la velocidad es el browser!), además, podía usar menos tags, más simples, eliminar los stylesheets y demás cosas 'ajenas' al HTML. Hoy puedo decir que el motivo por el cual recomiendan el uso del Internet Explorer en muchos sites es porque, al desarrollar con herramientas de Microsoft puras pierden el control del código que finalmente se genera, y este suele contener muchos de esos elementos no standard. Por cierto, todos estos agregados se pueden obtener perfectamente con el uso del HTML standard. Finalmente, Netscape no modifica las configuraciones de entorno del sistema operativo, como si lo hace el Internet Explorer.

No tiene mucho sentido explicar la instalacion de este programa, ya que desde internet se baja directamente el programa de instalacion, y tiene un conjunto minimo de parametros a setear durante la instalacion. En cambio, podemos mencionar algunas recomendaciones para hacer un uso mas eficiente del mismo.

Recomendaciones

Una manera de aprovechar las caracteristicas del browser es revisar cuales son sus opciones de configuración; en la última versión de Netscape vamos a encontrar las siguientes opciones:

- Appearance: aquí se ubican las características referidas al aspecto de cada ventana. Las más importantes son:

- ✓ Que ventana de la aplicación abrir cuando al inicio de una sesión (pueden ser; navegador, mensajería, newsgroups, editor de páginas web o NetCaster).
- ✓ Como ver los botones de la barra de navegación (podemos verlos con una leyenda sola indicando a que se refiere cada uno, con un dibujo solo para cada botón o combinar la leyenda y el dibujo). Por una cuestión de 'espacio libre' para visualizar mejor las páginas, recomendamos visualizar esta barra de botones en modo 'Text Only', ocupando así menos espacio útil.
- ✓ Fonts (es una sub-categoría, por lo tanto se selecciona desde el frame izquierdo): desde donde podemos elegir el tipo de letra con el que más nos guste visualizar las páginas, el tamaño, e indicar si queremos o no que el browser respete estos formatos o utilice los propios de la página.
- ✓ Colors (también es una sub-categoría): podemos indicar el color del texto, del fondo de página, de los links a otras páginas. Estas características serán tenidas en cuenta en los casos de que estos atributos no estén definidos en la construcción de la página.

- Navigator: Comprende las características exclusivas de la ventana de navegación. Podemos indicar las siguientes cuestiones:
  - ✓ Con que página se va a abrir originalmente la ventana (puede ser una página en blanco, una página indicada como 'Home' o en la última página visualizada en la sesión anterior de la aplicación).
  - ✓ Cual va a ser la página Home, es decir, cual será nuestro punto de partida. Netscape utiliza la dirección que le indicamos en este campo para dos elementos: cuando le indicamos, en el punto anterior, que inicialmente abra determinado sitio, y cuando presionemos uno de los botones de navegación que también se llama 'Home'. En nuestro caso, esta dirección estará apuntando a la página de entrada de la intranet del banco.

- Mail & Groups: esta categoría comprende las configuraciones de la ventana de mensajes del Netscape, y no la analizaremos debido a que no usaremos esta funcionalidad.

- Composer: aquí se ubican las configuraciones referidas a la ventana de composición de páginas web, tampoco entraremos en detalle en este punto, por no considerarlo necesario, pero nunca está de más saber todo lo que se puede hacer con cada programa!.

- Offline: desde aquí se le indica al browser si va a estar continuamente conectado a Internet o a la red interna o no. En nuestro caso estaremos conectados en línea, es decir, todo el tiempo, así que tampoco aquí tenemos nada interesante para configurar.

- Advanced: desde aquí se configuran los aspectos de seguridad del browser, los recursos de los cuales puede disponer, etc.

Integración de las distintas herramientas

Hasta aqui se descrbio el ambiente utilizado para el desarrollo de una intranet utilizando ASP's; ahora veremos una breve explicacion acerca de como interactuan estos componentes:

Tanto el IIS como el SQL Server se pueden ejecutar como servicios del NT Server, esto nos asegura que cada vez que se encienda el equipo estas dos aplicaciones seguro quedaran funcionando.

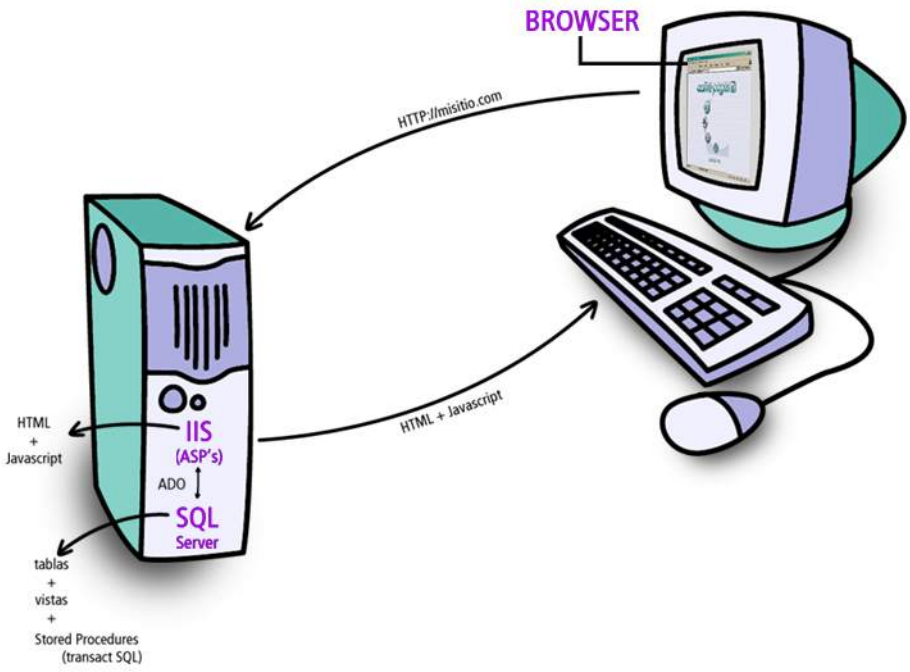
El IIS es, en principio, un simple servidor de páginas HTML, el browser solicita una página accediendo a determinada direccion IP, el server con dicha direccion detecta el pedido, verifica si la página solicitada requiere algun tipo de autenticacion (en tal caso envia al browser un pedido de usuario y password) y cuando la seguridad fue resuelta envia al browser la página solicitada. Esto es lo que hace cualquier Web Server.

El IIS contiene, ademas, un componente que puede interpretar codigo de scripting contenido en las páginas, este se denomina ASP's (Active Server Pages), esta incorporado al IIS en forma de dll, esto significa que se ejecuta dentro del ambiente de recursos del IIS.

La aplicación principal de la intranet estara, entonces, en forma de scripts dentro de las páginas que conformen el site. Desde estos scripts sera posible acceder a la base de datos, a través de una coleccion de objetos llamada ADO (ActiveX Data Objects), desde los cuales vamos a poder establecer una conexion con SQL Server (o cualquier otra base de datos que pueda ser accedida via ODBC), ejecutar un comando (pedido o envio de datos a la base) y obtener los resultados que me seran devueltos en un recordset.

Una vez obtenida la informacion que deseo, solo me resta agregarle los tags de HTML necesarios para formatearlos de acuerdo a como los quiera presentar al browser que solicito la página.

Graficamente, el esquema seria el siguiente:



Como se puede observar, el esquema de la aplicación es muy simple, y a pesar de que en la practica me encontre con serias limitaciones en las herramientas que utilice, creo que es una excelente manera de desarrollar aplicaciones de un modo eficiente y rapido.

IV Lenguajes

Las ASP's consisten, básicamente, en páginas de HTML con bloques de scripting embebido entre los tags de formateo. Estos scripts se pueden resolver en el web server (por ejemplo para comunicarse con una base de datos, comparar resultados, efectuar búsquedas, etc.) o en el mismo browser (por ejemplo para validar datos que ingrese el usuario, reaccionar frente a determinados eventos, etc.).

Así como pudimos observar dos ambientes para clasificar las herramientas, desde el punto de vista de los lenguajes a utilizar podemos enumerarlos en el siguiente orden:

- En el IIS:
  - ✓ **HTML** (Hiper Text Markup Language): es un lenguaje que consiste en la utilización de tags (marcadores encerrados entre signos'<' y '>'). Básicamente este lenguaje se utiliza para formatear el texto que conforma cada página, no contiene sintáxis alguna para la incorporación de lógica, solo por los tags <script> o <% %> que indican que lo que hay entre ellos es código de scripting para ser interpretado, ya sea por el Web Server o por el Browser,
  - ✓ **JScript:** es el lenguaje de scripting, que se escribe entre los tags <script> o <% %> para agregarle lógica a las páginas, que va a ser interpretado por el motor de las ASP's,
  - ✓ **ADO:** es un subset de las ASP's, formado por un conjunto de elmentos que permiten la conexión con la base de datos,
  - ✓ **JavaScript:** es un lenguaje adicional, que desde las últimas versiones los browsers pueden interpretar, utilizado basicamente para agregarle logica o dinamismo a la informacion contenida en las páginas de HTML, pero que es interpretado por el browser y no por el web server.

- En SQL Server:
  - ✓ **Transact-SQL Server:** es el lenguaje utilizado para generar objetos en la base de datos (algunos de estos objetos pueden ser creados desde entornos gráficos, pero por ejemplo los Stored Procedures deben ser generados siempre en este lenguaje).

A continuación paso a describir en que consisten y para que se utilizan cada uno de estos lenguajes.

HTML

Este lenguaje de tags fue pensado para ser universal, esto es, que cualquier browser interpretara de la misma manera a cada uno de sus tags, logrando así que una misma página sea vista de la misma manera en cualquier equipo y software. Pero el traslado de esta idea a la realidad genero tres problemas:

1. Si bien hay un organismo que se ocupa de estandarizar los tags, al ser una idea tan reciente, se van agregando nuevos tags, y cada versión de browser los va incorporando, pero hacia atrás, estas aplicaciones no son compatibles (un clásico ejemplo fue la implementación de

tags para el armado de tablas en HTML, que solo son soportados a partir de las versiones 2.0 del Netscape y del Explorer).

2. Cada desarrollador de browsers tiende a implementar sus propios tags de HTML, cuando las ventajas que estos desean incorporar aún no fueron contempladas por los estándares de HTML. Con esto, las aplicaciones tienden a desarrollarse orientadas a un determinado browser (lo aconsejable siempre es usar tags estándar, resignando las nuevas características a cambio de poder ser accedidos desde cualquier browser).

3. Algo todavía no logrado por los browsers es la no utilización de elementos propios del sistema operativo donde se estén ejecutando, como por ejemplo, la paleta de colores. Así, una misma página, accedida desde un mismo browser, pero en equipos con sistemas operativos diferentes, se va a ver distinta. Este problema no nos impide ver correctamente la página; en mi opinión esto no es un problema, sino un límite realista de la excelente idea de generar páginas universales!.

Para comenzar, debemos saber que el lenguaje HTML se utiliza para formatear contenidos de documentos, por esto vamos a hablar de páginas HTML, es decir, documentos formateados con este lenguaje. Estos tags de formateo son interpretado por los browsers, y los documentos pueden, de todas maneras, leerse desde cualquier editor de páginas de texto, donde se podran detectar dichos tags como palabras en inglés encerradas entre símbolos '<' y '>'.

Estos tags pueden clasificarse dentro de las siguientes categorias:

- Tags para definir la estructura del documento: son los que se utilizan para diferenciar las distintas secciones del documento dentro del archivo que conforma la pagina.
- Tags de encabezado: que permiten enumerar ciertas caracteristicas de la pagina.
- Tags indicadores de bloques de texto, que nos permitiran distinguir parrafos, encabezados, etc.
- Indicadores de listas.
- Indicadores de formateo de texto.
- Marcadores a otras páginas y a otros puntos de la misma u otra pagina.
- Tags que permiten insertar imágenes en la pagina, mapearles punteros a otros puntos, etc.
- Tags para el armado de tablas.
- Tags para insertar elementos de formularios (campos para ingreso de texto, popups, checkbox, etc.).

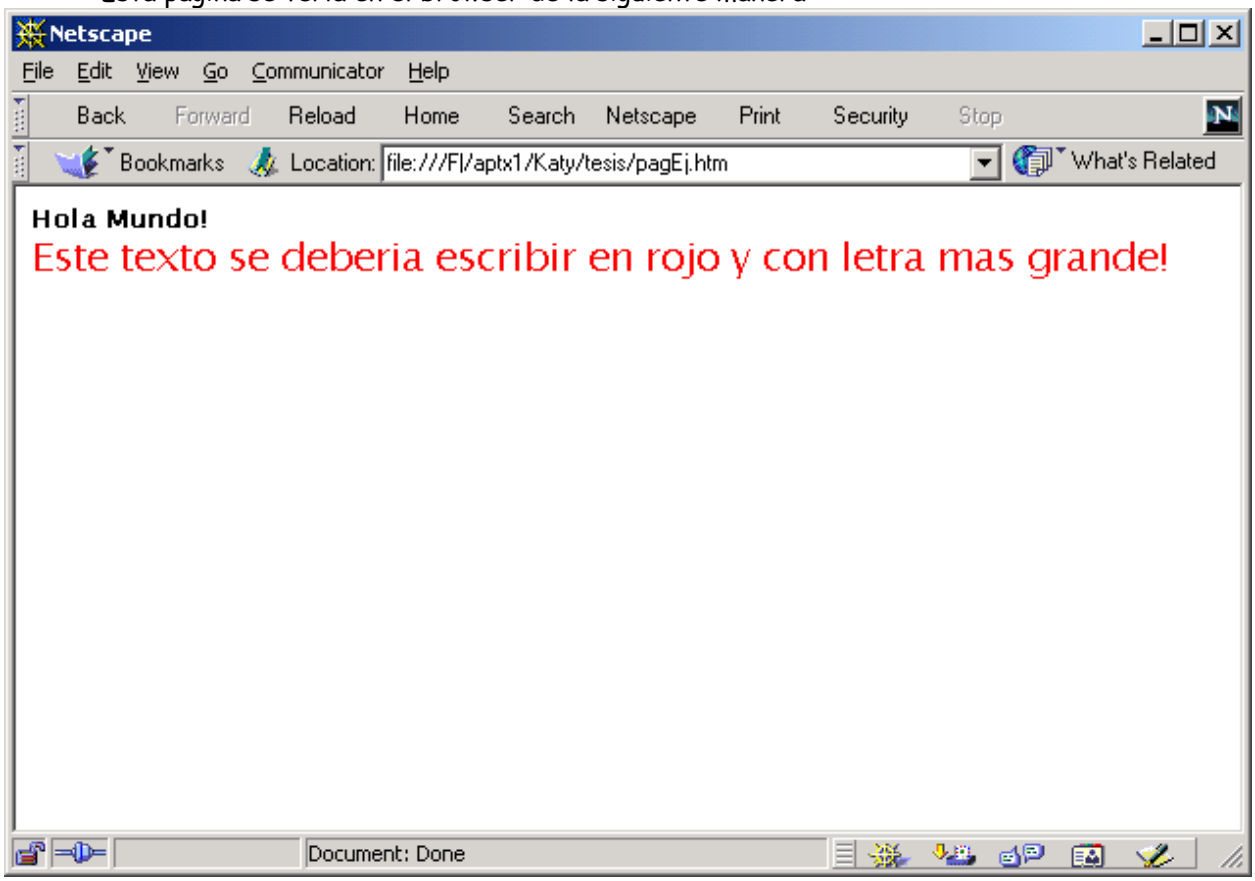
En el apendice A se detallan cada uno de los tags estandares que pueden ser incluidos en cada una de estas secciones.

Una página HTML podría contener, entonces, el siguiente texto:

```
<HTML>
<HEAD>
  <TITTLE>Ejemplo 1<TITTLE>
</HEAD>
<BODY>
  <B>Hola Mundo!</B><BR>
  <FONT COLOR='RED' SIZE=+2>
```

```
    Este texto se deberia escribir en rojo y con letra mas grande!
  </FONT>
</BODY>
</HTML>
```

Esta página se vería en el browser de la siguiente manera:



Como se puede ver, solo el texto que no forma parte de un tag es mostrado como contenido de la página. Los demás tags son necesarios para la correcta descripción de una página HTML:

- <HTML> se utiliza para indicar que todo lo que se encuentre antes del tag </HTML> es el contenido de la página; es decir, si agrego texto afuera de estos tags, el browser no lo considerará parte de la página y se limitará a ignorarlo.

- <HEAD> y </HEAD> contienen información correspondiente al encabezado de la página; en este ejemplo, lo único que se indica es el título (que el browser mostrará en el nombre de la ventana), pero se pueden indicar otras características.

- <BODY> y </BODY> encerrarán el contenido de la página, es decir, lo que el browser mostrará como contenido de la página.

- <B> y </B> indican que el texto que se encuentre entre ellos se debe mostrar en negrita, así como <FONT> y </FONT> indican el color y tamaño en que se debe mostrar el texto que contienen. Hay otros muchos tags de este tipo, la mayoría con atributos que permiten

formateos mucho más sofisticados (como el COLOR y el SIZE del tag FONT). La totalidad de estos tags se encuentran en el apéndice A.

En este ejemplo, se puede ver que en la 'location' del browser aparece el path indicando donde se encuentra el archivo en mi disco (file:///Katy/Desktop %20Folder/Tesis/Ejemplo1.html); cuando escribamos páginas '.ASP' debemos considerar que solo podremos llamarlas a través de la dirección del Web Server que las interprete, para que el script sea resuelto antes de llegar al browser.

Recomendaciones

- El uso de las comillas al indicarles valores a los atributos de un tag (es indistinto que sean simples o dobles) no es necesario a menos que dicho valor contenga algun espacio o caracter '>'.

- Si se colocan varios espacios entre palabras, el browser solo interpretará uno, ignorando los demás.

- Todos los caracteres de nueva línea, enter y tab son ignorados por el browser, aún así, se recomienda usarlos para mantener el código de las páginas más claro y ordenado.

- Todas las páginas HTML deben contener la extensión .htm o .html, aún cuando el sistema operativo donde se encuentre el Web Server no requiera el uso de extensiones (como en el caso de las macs); esto es para que el browser las pueda interpretar. Mas adelante veremos que hay otras extensiones que también son permitidas (como .cgi, .asp) pero estas deben explicitarse como páginas especiales que el browser pueda interpretar, las únicas comprendidas de manera implícita por el browser son las .htm o .html.

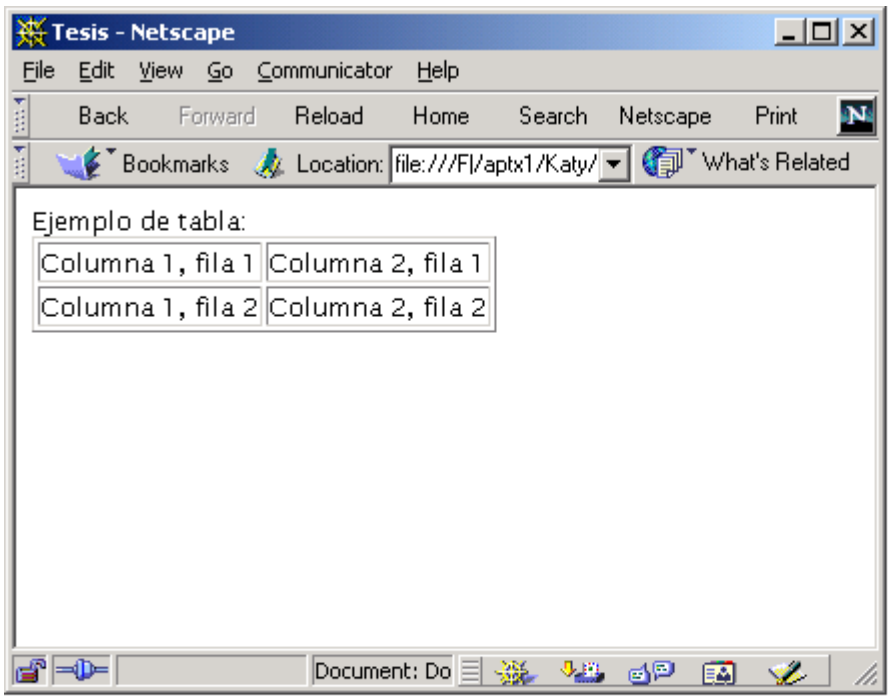
Ejemplos

- Página conteniendo un listado (para ver como se arma una tabla)

```
<html>
<head><title>Tesis</title></head>
<body>
<p>Ejemplo de tabla:
<table border=1>
        <tr>
                <td>Columna 1, fila 1</td>
                <td>Columna 2, fila 1</td>
        </tr>
        <tr>
                <td>Columna 1, fila 2</td>
                <td>Columna 2, fila 2</td>
        </tr>
</table>
</body>
```

```
</html>
```

Vista en el browser:

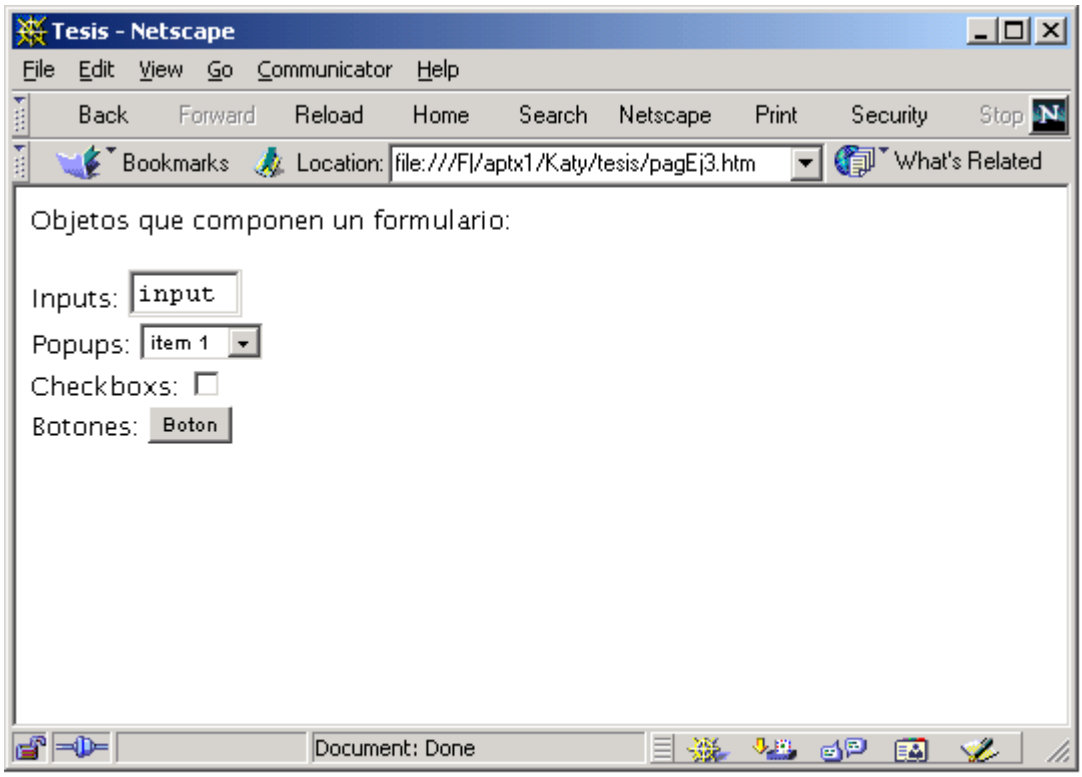


- Página conteniendo un formulario (para ver como se envian los datos al servidor)

```
<html>
<head><title>Tesis</title></head>
<body>
<p>Objetos que componen un formulario:

<form name=body action=app.asp method=POST>

Inputs: <input name=input type=text size=6 value='input'><br>

Popups: <select name=popup>
                <option value=1 selected>item 1
                <option value=2>item 2
                <option value=3>item 3
</select><br>

Checkboxs: <input name=checkbox type=checkbox><br>

Botones: <input name='Boton' type=button value='Boton'><br>
```

```
</form>
</body>
</html>
```

Vista en el browser:



JScript

Este es un lenguaje interpretado, basado en objetos y de scripting que contiene algunas de las características de los lenguajes tradicionales orientados a objetos (como C++ y Java). Algunas de las limitaciones de este lenguaje respecto de los tradicionales son:

- ✓ No se pueden crear aplicaciones separadas en este lenguaje, siempre dependerán de un Web Server que las interprete.

- ✓ No es un lenguaje fuertemente tipeado, es decir, no es obligatorio declarar las variables que se vayan a utilizar, ni asignarles un tipo de datos preestablecido, sino que el interprete se lo asignará de acuerdo al tipo de información que se quiera almacenar en ellas (aún así siempre es recomendable declarar e inicializar las variables, para tener un mayor control sobre los distintos elementos de la aplicación).

Como cualquier otro lenguaje, este se compone de variables, sentencias, bloques de sentencias y comentarios. A continuacion se describen cada uno de estos elementos.

Variables

Las variables se utilizan para almacenar los distintos datos o valores necesarios para realizar una determinada accion con ellos dentro del codigo en el que estan insertadas. La sentencia para declarar variables en Jscript es 'var', y en la misma declaracion las variables pueden ser inicializadas (esto es altamente recomendable para evitar reacciones confusas en el codigo que las vaya a utilizar. Ademas, Jscript les asigna el valor 'undefined' si no estan inicializadas, y este dato no es compatible con ningun otro tipo de datos, esto puede generar problemas a futuro en el funcionamiento de la aplicación que estemos desarrollando).

Ejemplos:

```
var mim = "Hola mundo!";        //string
var ror = 3;                    //integer
var nen = true;                 //boolean
var fif = 2.718281828;          //float
```

Algunas consideraciones para los nombres:

- El primer carácter debe ser una letra o un simbolo '_' o '$',
- Los demas caracteres pueden ser letras, numeros o los simbolos antes mencionados,
- No se pueden utilizar las palabras reservadas del lenguaje (esto es, todas aquellas que se utilizan para escribir una sentencia).

Sentencias

Una sentencia es, en principio, una línea de código, pero aquí (al igual que en C) se puede escribir más de una sentencia por línea, separándolas con un ';'. Cada sentencia puede representar una expresión (booleana o numérica) o una asignación.

Ejemplo:

```
var today = new Date();
```

Bloques

Los bloques de instrucciones o sentencias se encierran entre llaves ({}), y en general encierran funciones o acciones a seguir por las distintas salidas de una condición.

Ejemplo:

```
var theMoments = "";
var theCount = 42;
while (theCount >= 1)  {
 if (theCount > 1)  {
    theMoments = "Only " + theCount + " moments left!";
 }  else  {
    theMoments = "Only one moment left!";
 }
theCount--;       // Update the counter variable.
```

```
}
theMoments = "BLASTOFF!";
```

Tanto las sentencias como los bloques de sentencias pueden contener 'Controladores de Flujo del Programa', que no son mas que sentencias condicionales, en base a las cuales una corrida del programa puede seguir una linea u otra de codigo, repetir un conjunto de instrucciones, etc. Los condicionales soportados por Jscript son: If ... else, For ... in, While, Do while, Switch.

Funciones

Las funciones son bloques de sentencias a las que se puede acceder utilizando el nombre que las contiene, generalmente devuelven un valor, aunque esto no es obligatorio, y pueden tambien modificar el valor de variables que hayan sido creadas afuera de ellas. Tambien se les puede pasar informacion a las funciones, a través de sus parametros.

Ejemplo:

```
Var hola = 'hola mundo';

Function reemplazar_Str(str,x,y){
 str = str.reeplace(x,y);
 return str
}
hola = reemplazar_Str(hola,'hola','chau')
```

El resultado de ejecutar este codigo nos devolveria el string 'chau mundo'.

Comentarios

Finalmente, los comentarios son frases escritas en lenguaje natural que se delimitan con los caracteres '//' para indicar que desde esa posicion hasta el final de la línea hay un comentario o '/* ... */' donde los puntos suspensivos representan el comentario que se quiere guardar en el documento.

Ejemplos:

```
var hola = "Hola Mundo";  //Aquí comento el significado de la linea

/* aquí agrego varias lineas de codigo, para, por ejemplo, comentar la accion que
desarrolla una funcion, que parametros espera, cual es la informacion de salida,
etc.*/
```

Objetos

Como dijimos anteriormente, este lenguaje esta basado en el uso de objetos, es decir, en colecciones de propiedades y metodos, donde una propiedad es un valor asignado a una caracteristica del objeto y un metodo es una funcion que se ejecuta sobre las propiedades del objeto.

Jscript soporta tres tipos de objetos:

- Intrinsecos: Array, Boolean, Date, Function, Global, Math, Number, Object, y String,
- Creados por el usuario,
- Objetos propios del browser.

Palabras Reservadas

Son palabras que debemos evitar usar como nombres de variables, ya que son interpretadas como elementos propios del lenguaje (indicadores de sentencias, de declaraciones, tipos de datos, etc.)

La lista de objetos existentes en el lenguaje y como crear objetos propios se encuentra en el apendice B; y para mas informacion acerca del lenguaje, al instalar el IIS se genera un web site default con la documentacion original, se puede acceder a través de la siguiente direccion: http://ip_adress_o_dns/iishelp/JScript/htm/Jstutor.htm

ADO

ADO (ActiveX Data Objects) es es una colección de objetos que se utilizan para acceder a cualquier base de datos a la que se pueda conectar a través de OLE DB provider. Esta coleccion de objetos esta compuesta por los siguientes objetos:

- **Connection:**
  Mantiene la información referida a la conexión con el proveedor de datos.

  Propiedades:
  - ✓ Attributes: indica características de un objeto. Los valores posibles a ser asignados son:
    - AdXactCommitRetaining: para indicarle a la base de datos que ejecute un Commit Transaction,
    - AdXactAbortRetaining: para indicar un Rollback.
  - ✓ CommandTimeout: indica cuanto tiempo tiene asignada la ejecución de un comando como máximo.
  - ✓ ConnectionString: contiene la información necesaria para establecer la conexión con la base de datos.
  - ✓ ConnectionTimeout: contiene la cantidad de segundos que puede quedar el IIA esperando antes de cancelar la ejecucion de un comando. Es muy util configurarlo cuando se esta desarrollando, ya que puede suceder que un proceso no necesariamente termine en modo normal, o haya algun problema en la base de datos por lo cual el comando puede quedarse esperando indefinidamente, excepto cuando esta propiedad esta configurada.
  - ✓ CursorLocation: contiene un puntero a la librería desde la cual va a administrar el uso de cursores en un recordset. Los valores posibles son:
    - AdUseClient: utiliza el cursor propio de ADO,
    - AdUseServer: utiliza el cursor propio de la base de datos.

No es una propiedad que haya usado, debido a que no realizo operaciones de base de datos del lado de las ASP's; prefiero realizarlas en Stored Procedures dentro de SQL Server.

- ✓ DefaultDatabase: permite indicar una base de datos default para el objeto conexión. Esto tampoco lo utilizo, ya que siempre apunto a una misma base de datos, pero seria una manera de utilizar, por ejemplo, distintas bases de datos cuando dentro de una intranet diferenciamos varias aplicaciones. En mi experiencia me resulto mas practico generar una sola base de datos, para asi compartir muchos de los objetos creados que son comunes a todas las aplicaciones.
- ✓ IsolationLevel: indica el nivel de aislacion de una conexion. Se utiliza cuando se manejan transacciones batch, y permite decir si otras transacciones pueden ver los cambios que se estan realizando o no en un conjunto de datos.
- ✓ Mode: indica los posibles permisos para acceder a los datos recuperados por la conexion. Esta propiedad no la utilizo debido a que solo accedo a los datos en modo lectura, debido a que no modifico datos desde el ambiente de las ASP's sino a través de Stored Procedures.
- ✓ Provider: indica el nombre del proveedor de la base de datos a la que va a acceder la conexion.
- ✓ State: indica el estado actual de la conexión, esto es, si esta abierta o cerrada.
- ✓ Version: indica el numero de versión de ADO.

Metodos:

- ✓ BeginTrans: comienza una nueva transaccion.
- ✓ CommitTrans: graba los cambios y finaliza la transaccion abierta, pudiendo tambien iniciar una nueva transaccion.
- ✓ RollbackTrans: cancela los cambios realizados desde el comienzo de la transaccion y la finaliza, pudiendo tambien abrir una nueva.
- ✓ Close: cierra una conexión.
- ✓ Execute: ejecuta un query determinado, que puede ser una sentencia de SQL o un Stored Procedure.
- ✓ Open: abre una conexión a una fuente de datos.

- **Command:**

Mantiene la información relacionada con un comando, como el string que contiene el query a ejecutar, parámetros relacionados, definiciones, etc. No es necesario utlizar este objeto para acceder a la base de datos, a menos que se quieran definir parámetros de un stored procedure para recibir como respuesta parámetros de salida (es decir, llamar a stored procedures que devuelvan valores en variables en vez de recordsets). No recomendamos el uso de comandos, debido a que en la versión anterior de ADO el esquema de devolución de parámetros de salida no funcionaba, y gracias a eso descubrimos que es más eficiente, para el control posterior, que todos los llamados a la base de datos devuelvan un recordset (esto es, podemos asegurarnos siempre si obtuvimos la respuesta deseada de la base de datos).

Propiedades:

- ✓ ActiveConnection: indica mediante que conexión se va a ejecutar el comando a enviar a la base de datos.
- ✓ CommandText: contiene el query a ejecutar.
- ✓ CommandTimeout: setea el tiempo a esperar la ejecucion de un comando en la base de datos antes de devolver un mensaje de que no se pudo ejecutar.

- ✓ CommandType: indica el tipo de comando a enviar. Los valores posibles son:
  - ✓ AdCmdText: el texto enviado es una definicion textual de un comando.
  - ✓ AdCmdTable: el texto es el nombre de una tabla.
  - ✓ AdCmdStoredProc: el texto es un stored procedure (este es que vamos a utilizar en la aplicación que vamos a desarrollar mas adelante).
  - ✓ AdCmdUnknown: Default. Indica que el tipo de texto a enviar no fue definido.
- ✓ Name: contiene el nombre del comando.
- ✓ State: indica el estado del comando, es decir, si esta abierto o cerrado.

Metodos:

- ✓ CreateParameter: crea un nuevo parametro para el comando a ejecutar.
- ✓ Execute: indica la ejecucion del comando.

- **Parameter:**

Guardan los parámetros a enviar a un stored procedure al momento de ejecutar un comando.

Propiedades:

- ✓ Attributes: indica determinadas caracteristicas del parametro, como ser si el parametro acepta valores nulos, datos signados o datos binarios.
- ✓ Direction: indica si es un parametro de ingreso, salida o en ambos sentidos de envio de datos.
- ✓ Name: guarda el nombre del parametro.
- ✓ NumericScale: indica la escala numerica de los posibles valores que pueda almacenar el parametro.
- ✓ Precision: indica el grado de precision de los valores en caso de ser numericos
- ✓ Size: indica el maximo de caracteres admitidos en el contenido del parametro.
- ✓ Type: indica el tipo de dato que contiene el parametro.
- ✓ Value: asigna un valor al parametro.

Metodos:

- ✓ AppendChunk: se utiliza para agregar textos extensos a un parametro.

- **Recordset:**

Almacena los datos enviados por la base de datos (podemos imaginarlo como una grilla de Excel, donde los datos se encuentran organizados en filas y columnas). Este objeto es la interfaz principal con los datos

Propiedades:

- ✓ AbsolutePage: indica en que pagina se ubica el recordset.
- ✓ AbsolutePosition: especifica la posicion ordinal de un registro dentro de un recordset.
- ✓ ActiveConnection: indica que conexión se utilizo para generar el recordset.
- ✓ BOF: indica la posicion inicial del recordset (antes del primer registro).
- ✓ Bookmark: es un marcador de un registro dentro de un recordset.
- ✓ CacheSize: es el numero de registros que se pueden almacenar localmente en memoria.
- ✓ CursorLocation: configura la ubicación del motor de cursores (es decir, si se usan los de la base de datos o los propios de ADO).
- ✓ CursorType: indica que tipo de cursor utilizar en un recordset.

- ✓ EditMode: indica el estado de edicion de un registro (esto se utiliza si se trabaja sobre los recordsets. En este trabajo los recordsets solo se leen, no se opera sobre ellos).
- ✓ EOF: marca el fin del recordset, es decir, despues del ultimo registro.
- ✓ Filter: guarda los criterios de busqueda del query que genera el recordset.
- ✓ LockType: indica el tipo de lockeo a utilizar durante la edicion de los registros del recordset.
- ✓ MarshalOptions: indica cuales registros deben ser actualizados en la base de datos, esto es: todos o solo los modificados.
- ✓ MaxRecords: permite indicar un maximo de registros a recibir en un recordset.
- ✓ PageCount: indica la cantidad de páginas que contienen el recordset
- ✓ PageSize: define cuantos registros conforman un recordset.
- ✓ RecordCount: guarda la cantidad de registros que contiene el recordset.
- ✓ Source: contiene la fuente desde donde se extrajeron los registros que conforman el recordset.
- ✓ State: es el estado del recordset, esto es: indica si esta abierto o cerrado.
- ✓ Status: indica el estado de un registro, es decir: si fue updateado correctamente, si fue modificado, si es nuevo, etc.

Metodos:

- ✓ AddNew: agrega un registro nuevo al recordset.
- ✓ CancelBatch: cancela la ejecucion de un update en modo batch.
- ✓ CancelUpdate: cancela los cambios realizados en un registro.
- ✓ Clone: realiza una copia del recordset.
- ✓ Close: cierra un recordset abierto.
- ✓ Delete: borra un registro perteneciente al recordset.
- ✓ GetRows: copia varios registros del recordset en un array.
- ✓ Move: cambia la posicion del puntero de un recordset a un determinado registro.
- ✓ MoveFirst: apunta al primer registro del recordset.
- ✓ MoveLast: apunta al ultimo registro del recordset.
- ✓ MoveNext: va al registro siguiente del que esta apuntando actualmente.
- ✓ MovePrevious: se mueve al registro anterior al actual.
- ✓ NextRecordset: apunta al recorset siguiente (esto se utiliza cuando se abren varios recordsets al mismo tiempo, no es el caso de la aplicación detallada en este trabajo).
- ✓ Open: abre un nuevo recordset, cuyo contenido es la respuesta a un determinado comando.
- ✓ Requery: re-ejecuta el query que genero el recordset.
- ✓ Resync: refresca los datos del recordset actual.
- ✓ Supports: indica que metodos soporta un determinado recordset.
- ✓ Update: graba los cambios realizados en el recordset en la base de datos.
- ✓ UpdateBatch: graba los cambios en modo batch.

- **Error:**

Almacena los mensajes y errores producidos durante la conexión con la base de datos. Este objeto tampoco es necesario para conectarse a una base de datos, de hecho, tuve que implementar controles propios para el control de la información que me devolvía la base de datos, debido a que cuando intenté utilizar este objeto fallaba muy frecuentemente (tiene

problemas con la limpieza de variables, y nunca logré capturar correctamente los mensajes necesarios para establecer los controles adecuados).

Propiedades:

- ✓ Description: string que describe el error ocurrido.
- ✓ HelpContext: indica el tema relacionado al error en el archivo indicado en HelpFile.
- ✓ HelpFile: indica el archivo help asociado al error ocurrido.
- ✓ NativeError: contiene el error especifico informado por el provider.
- ✓ Number: guarda el numero de error reportado.
- ✓ Source: es el nombre del objeto que genero el error.
- ✓ SQLState: contiene el codigo de error correspondiente al ANSI SQL standard.

Metodos: no tiene.

- **Field:**

Representa una columna particular del recordset.

Propiedades:

- ✓ ActualSize: indica la longitud actual del valor de un campo del recordset.
- ✓ Attributes: contiene los atributos propios del campo en la base de datos.
- ✓ DefinedSize: guarda el tamano definido en la base de datos del campo.
- ✓ Name: contiene el nombre del campo.
- ✓ NumericScale: indica la escala numerica utilizada en el campo.
- ✓ OriginalValue: guarda el valor original del campo, en caso de haber sido modificado en el recordset.
- ✓ Precision: contiene el grado de precision en el caso de que el campo contenga un valor numerico.
- ✓ Type: contiene el tipo de dato definido en la base de datos para el campo.
- ✓ UnderlyingValue: es el valor original del dato en la base de datos; la diferencia con OriginalValue es que este contiene el valor que originalmente estaba en la base, mientras que UnderlyingValue puede ver los cambios realizados desde que fueron tomados los datos.
- ✓ Value: contiene el valor del campo.

Metodos:

- ✓ AppendChunk: permite actualizar datos contenidos en textos largos o binarios.
- ✓ GetChunk: permite tomar datos desde campos de tipo texto o binario.

- **Property:**

Es la colección de propiedades propias del proveedor de base de datos utilizada.

Propiedades:

- ✓ Attributes: contiene los atributos soportados por el provider de datos, por ejemplo: si determinada propiedad es o no soportada, si es requerida u opcional, si es de solo lectura o puede ser modificada, etc.
- ✓ Name: contiene el nombre de la propiedad.
- ✓ Type: contiene el tipo de dato correspondiente a la propiedad.
- ✓ Value: contiene el valor asignado a la propiedad.

Metodos: no tiene.

Estos objetos pueden ser creados y utlizados independientemente de su jerarquía, es decir, no es necesario crear todos los objetos para acceder a la información que necesitamos. Support for stored procedures with in/out parameters and return values.

En princpio, este ambiene nos permite varias facilidades en el manejo de información enviada y recibida de la base de datos (como control de cantidad de registros devueltos, soporte para multiples conjuntos de datos de respuesta, realizar updates en modo batch, etc.), pero no aconsejo utlizar estas facilidades, ya que pueden ser perfectamente implementadas del lado de la base de datos, de hecho ese es el ambiente natural para realizar estas tareas. Además, este ambiente se encuentra del lado de las ASP´s, y en un principio no era una herramienta muy robusta, muchas de sus características fallaban y opté por dejar las cosas en este ambiente lo más simples posible.

Ciertas propiedades pueden utilizarse sin ser explicitadas, como por ejemplo la propiedad Item. Entonces se cumplen las siguientes igualdades:

Command.Properties.Item(0) = Command.Properties(0)
Command.Properties.Item("Name") = Command.Properties("Name")

son idénticas.

Podemos agregar que la colección Properties es la que ADO asume si no se especifica ninguna, entonces, las sentencias anteriores se podrían expresar como:

Command(0)
Command("Name")

Así podemos simplificar el código a la hora de escribir llamadas a la base de datos.

Transact-SQL

Este lenguaje se utiliza dentro de la base de datos, y esta fuertemente orientado a la manipulacion de los datos que esta contiene. A pesar de ser muy rudimentario y contar con una sintaxis reducida, permite realizar practicamente todas las operaciones necesarias para crear una base de datos, todos sus objetos (tablas, vistas, stored procedures, etc.) y administrar el ingreso de informacion a la misma, realizando validaciones y calculos en caso de ser necesario.

Los elementos que encontramos en este lenguaje son, estructuralmente, similares a los de la mayoria de los lenguajes convencionales:

- Identificadores: son los nombres de los distintos objetos que componen la base de datos: tablas, vistas, stored procedures, columnas, server, etc.

- Tipos de Datos: definen que tipo de dato puede contener una variable, columna de tabla o parametro. En algunos casos, al procesar informacion, no es necesario especificar algunos tipos de datos, siendo que estos pueden ser derivados del tipo de dato que se esta utilizando (esto es, por ejemplo, al realizar una consulta a una tabla, se asumiran los tipos de datos

declarados al crear la tabla, no es necesario volver a especificarlos para ejecutar la consulta).

Al definir un tipo de dato estamos asignando 4 atributos al objeto en cuestion:

- Tipo de informacion que contendra el objeto (si sera texto, numero o binario),
- Tamaño del objeto (los campos de tipo caracter se definen por cantidad de caracteres, los de tipo imagen, binario o numerico se definen en bytes),
- La presicion de un valor numerico (es decir, el numero de digitos que podra contener ese objeto)
- La escala del numero (cantidad de digitos a la derecha del punto decimal)

Los tipos de datos que soporta Transact SQL son:

- binary: almacena datos en formato binario, de tamaño fijo, soporta un maximo de 8000 bytes
- bit: solo soporta los valores 0 y 1
- char: contiene valores de tipo caracter, de tamaño fijo, puede contener hasta de 8000 caracteres
- cursor: referencia a un cursor
- datetime: contiene datos de fecha y hora entre el 1/1/1900 y el 6/6/2079 con presicion de milisegundos
- decimal: contiene valores decimales de presicion fija entre –10E38 –1 y 10E38 -1
- float: contiene numeros con presicion flotante entre –1.79E308 y 1.79E308
- image: contiene valores binarios (2.147.483.647 bytes maximo) representando imagenes
- int: contiene numeros enteros entre –2.147.483.648 y 2.147.483.648
- money: permite almacenar valores monetarios (con 4 digitos decimales) con valores entre –922.337.203.685.477,5808 y 922.337.203.685.477,5808
- nchar: guarda caracteres en formato Unicode, es de tamaño fijo y soporta un maximo de 4000 caracteres
- ntext: guarda caracteres en formato Unicode, es de tamaño variable y soporta un maximo de 1.073.741.823 caracteres
- nvarchar: guarda caracteres en formato Unicode, es de tamaño variable y soporta un maximo de 4000 caracteres
- real: valores de presicion flotante entre -3.40E38 y 3.40E38
- smalldatetime: contiene datos de fecha y hora, con presicion hasta el minuto
- smallint: contiene valores enteros entre –32.768 y 32.768
- smallmoney: contiene valores monetarios con valores entre –214.748,3648 y 214.748,3648
- text: contiene caracteres, es de tamaño variable, y su capacidad maxima es de 2.147.483.647 caracteres
- timestamp: identificador unico de objetos en la base de datos. Seria un buen candidato para claves unicas, de no ser porque se modifica cada vez que un objeto es modificado!
- tinyint: valores enteros entre 0 y 255
- varbinary: almacena datos en formato binario, de tamaño variable, y hasta un maximo de 8000 bytes
- varchar: contiene caracteres (hasta 8000), es de tamaño variable
- uniqueidentifier: identificador unico de objeto

- Funciones: son bloques de sentencias ya definidas en la base de datos, que pueden recibir parametros y retornan un valor o un conjunto tabular de valores. Ejemplos de funciones en SQL Server son Datediff, sum, etc.

  Las funciones que contiene el lenguaje se pueden clasificar de la siguiente manera:
  - √ funciones Aggregate: aplican operaciones que combinan multiples valores retornando un nuevo valor. (AVG, COUNT, DISTINCT, GROUP BY, HAVING, MAX, MIN, NULL, STDEV,STDEVP, SUM, VAR, VARP).
  - √ funciones de Configuracion: retornan valores de parametros de configuracion de la base de datos(@@CONNECTIONS, @@DATEFIRST, @@DBTS, @@LANGUAGE, @@LANGID, @@LOCK_TIMEOUT, @@MAX_CONNECTIONS, @@MAX_PRECISION, @@OPTIONS, @@NESTLEVEL, @@REMSERVER, @@SPID, @@SERVERNAME, @@SERVICENAME, @@TEXTSIZE, @@VERSION).
  - √ funciones de Cursor: retornan informacion sobre el estado de un cursor (@@CURSOR_ROWS, @@FETCH_STATUS, CURSOR_STATUS).
  - √ funciones de Fecha y Hora: permiten manipular valores de tipo datetime y smalldatetime(DATEADD, DATEDIFF, DATENAME, DATEPART, DAY, GETDATE, MONTH, YEAR).
  - √ funciones Matematicas: ejecutan operaciones geometricas, trigonometricas, etc (ABS, ACOS, ALL, AND, ANY, ASCII, ASIN, ATAN, ATN2, AVG, CAST, CEILING, CONVERT, COS, COT, COUNT, DEGRESS, EXP, FLOOR, LOG, LOG10, POWER, RAND, ROUND).
  - √ funciones de Metadatos: devuelven informacion sobre los atributos de la base de datos y sus objetos (Abort, AcquireConnection, Add, AddColumn, AddConstraint, Clear, Commit, etc.).
  - √ funciones de Set de Filas: devuelven conjuntos de filas que pueden ser utilizadas de igual modo que los datos de una tabla en una sentencia de transact-SQl (CONTAINSTABLE, OPENQUERY, FREETEXTTABLE, OPENROWSET).
  - √ funciones de Seguridad: retornan informacion sobre usuarios y roles (ISMEMBER, ISSRVROLEMEMBER, SUSERID, SUSERNAME, SUSERSID, SUSERSNAME, USERID, USER).
  - √ funciones de String: permiten manejar informacion que contenga datos de los siguientes tipos: char, varchar, nchar, nvarchar, binary y varbinary (ASCII, CHAR, CHARINDEX, DIFFERENCE, LEFT, LEN, LOWER, LTRIM, NCHAR, PATINDEX, REPLACE, QUOTENAME, REPLICATE, REVERSE, RIGHT, RTRIM, SOUNDEX, SPACE, STR, STUFF, SUBSTRING, UNICODE, UPPER).
  - √ funciones del Sistema: operan sobre objetos y opciones del sistema (APPNAME, CASE, CONVERT, COALESCE, CURRENT_TIMESTAMP, CURRENT_USER, DATALENGTH, @@ERROR, FORMATMESSAGE, GETANSINULL, HOST_ID, HOST_NAME, IDENT_INCR, IDENT_SEED, @@IDENTITY, IDENTITY, ISDATE, ISNULL, ISNUMERIC, NEWID, NULLIF, PARSENAME, PERMISSIONS, @@ROWCOUNT, SESSION_USER, STATS_DATE, SYSTEM_USER, @@TRANCOUNT, USER_NAME).
  - √ funciones de Estadistica del Sistema: retornan informacion acerca de la performance del server de base de datos (@@CPU_BUSY, @@IDLE, @@IO_BUSY, @@PACK_RECEIVED, @@PACK_SENT, @@PACKET_ERRORS, @@TIMETICKS, @@TOTAL_ERRORS, @@TOTAL_READ, @@TOTAL_WRITE).
  - √ funciones de Texto e Imagenes: permiten manipular informacion de tipo text e image (PATINDEX, TEXTVALID, TEXTPTR).

- Expresiones: son unidades de sentencias que seran procesadas por el motor de base de datos. Por ejemplo, 'select * from usr' es una sentencia. Estas pueden contener constantes, funciones que retornan algun valor, referencias a columnas o variables.

- Operadores: son simbolos que permiten modificar condiciones o columnas en una sentencia. Estos se clasifican de la siguiente manera:
  - √ Aritmeticos: ejecutan operaciones matematicas basados en dos expresiones numericas (+, -, *, /, %).
  - √ De Asignacion: el unico operador de asignacion que soporta Transact-SQl es el signo '=', por medio del cual se asignan valores especificos a las variables.
  - √ Bitwise: son aquellos operadores que permiten manipular expresiones del tipo entero (& and, | or, ^ or exclusivo).
  - √ De Comparacion: estos operadores evaluan la relacion entre dos expresiones. No pueden ser usados con campos de tipo text, ntext o image (=, >, <, >=, <=, <>, !=, !<, !>).
  - √ Logicos: son aquellos que evaluan el valor de verdad de una expresion. Retornan solo dos valores posibles: falso o verdadero (ALL, AND, ANY, BETWEEN, EXISTS, IN, LIKE, NOT, OR, SOME).
  - √ De Concatenacion de Strings: Se utiliza el simbolo + y permite sumar strings o caracteres a un string o caracter.
  - √ Unarios: se consideran unarios aquellos operadores que afectan a una sola expresion por vez (+, -, ~).

- Comentarios: son trozos de texto incluidos entre las sentencias de un script. Como en todos los lenguajes, estos se utilizan para clarificar parte del codigo para su posterior interpretacion. El separador de comentarios utilizado para incluir comentarios entre las sentencias es la suma de dos signos '-' (--).

- Palabras reservadas: son palabras que solo pueden ser utilizadas por el propio lenguaje, y no se pueden utilizar como nombres de objetos o variables.

V Desarrollo de una Aplicación

El objetivo de este capitulo es generar cada uno de los componentes necesarios para llegar a conformar una intranet muy básica, con la finalidad de comprender como interactúan cada uno de los componentes involucrados para conformar un solo sistema.

Los pasos a seguir para lograr este objetivo son:

- Aprender a generar páginas,
- Inicializar un Web Site,
- Agregarle lógica a las páginas,
- Integrar la información estática con datos almacenados en una base de datos,
- Agregar lógica del lado del cliente

A continuación paso a exlicar como llevar a cabo cada una de estas tareas. Para esto se aplicara lo visto en la descripción del lenguaje HTML, pero generando páginas que conformen un site modelo. Esto es: una página inicial, una que muestre información detallada referida a lo que querramos mostrar y una donde tomar datos del visitante, para almacenarlos luego en la base de datos.

Generación de Páginas

Inicialmente, vamos a generar tres páginas:

√ Default.htm: conteniendo la información de presentación de nuestro site (a que se refiere, que productos o servicios ofrece y links a las demas páginas).

√ Info.htm: donde se muestre informacion adicional de lo que nuestro site ofrece,

√ Form.htm: para tomar los datos de quien visite nuestro site y desee comunicarse con nosotros.

Las paginas mencionadas se encuentran al final de este trabajo.

Inicialización del Web Site

Los pasos a seguir para generar el web site son los siguientes:

- Crear una carpeta en inetpub\wwwroot\ con un nombre que refiera a nuestro site (en nuestro caso sera: inetpub\wwwroot\tesis),
- Poner dentro de esta carpeta las páginas antes generadas,
- Ir al Internet Service Manager para declarar el nuevo site:
    - Seleccionar el server, presionar boton derecho del mouse y clickear 'New Web Site',
    - Ingresar el nombre del site (tesis)

- Elegir la IP Address donde va a responder el site (siempre que haya mas de un site definido debe apuntar cada uno a una direccion IP diferente),
- Indicar donde se encuentra la pagina de inicio del site (en nuestro caso es d:\inetpub\wwwroot\tesis),
- Definir si queremos o no acceso anonimo a nuestro site,
- Definir permisos de acceso (en nuestro caso solo de lectura y scripting).
- Seleccionar el nuevo site, presionar el boton derecho del mouse y clickear 'Properties',
- Deshabilitar la opcion 'Enable Logging' (consume demasiados recursos y podemos tener nuestro propio log incorporado a la aplicación mas adelante!),
- Seleccionar la solapa 'Home Directory'
- Darle un nombre apropiado a la aplicación asociada al site (esto es util, por ejemplo, para saber desde el log del SQL Server que site esta accediendo en el caso de que mas de uno tenga acceso),
- Presionar el boton 'Configuration' e ir a la solapa 'App Options',
- Subir la cantidad de minutos antes de cortar la conexión de un usuario (acostumbro indicar 1440 minutos, que es el equivalente a un dia),
- Cambiar el lenguaje default de ASP a JScript,
- En la solapa 'Documents' ingresar el nombre de la pagina de inicio del site (en nuestro caso default.htm)

➢ Ir al DNS Server y asociar la IP asignada al nuevo site con un nombre (en el ejemplo: www.tesis.com.ar).

Con estos pasos es suficiente para dar por generado un nuevo site, si ahora escribimos la direccion en el browser vamos a poder acceder a nuestro sitio al igual que accedemos a cualquier sitio de internet.

Logica del lado del Browser

Hasta ahora hemos trabajado como si se tratara de un site estatico tradicional, lo que hicimos hasta aquí no tiene nada particular: son solo páginas de HTML publicadas a través de un Web Server.

Una manera simple de agregar logica a un site es incorporar JavaScript del lado del browser. Para eso debemos saber lo siguiente:

✓ El codigo JavaScript debe estar encerrado entre tags:
'<script language=JavaScript> codigo que querramos incorporar </script>'

✓ Las rutinas JavaScript son accedidas a través del manejo de eventos producidos sobre los objetos, por medio de llamadas a sus metodos o desde llamadas a rutinas dentro del codigo de scripting. Estos eventos se detallan en el apendice B, capitulo de Eventos.

Podemos ver un ejemplo en la pagina formJavaScript.htm, donde encontraremos el bloque de rutinas JavaScript:

```
Function popID(popup){
 For(i=0;i< popup.length;i++){    // recorro todos los items del popup
  if(popup.options[i].selected) {return popup.options[i].text;alert(popup.options[i].value)}
```

```
// verifico si alguno de los items fue seleccionado, de ser asi retorno ese item
 }
 alert(popup.name+' invalido');  // aviso si no hay ningun item seleccionado
 return '';
}

function tomarDatos(nombre,origen,destino,obs){
 if (nombre.value==''){alert('Debe ingresar su nombre!');return};
 str= 'Hola '+nombre.value+'\n';
 str+='Entendemos que usted estaria viajando desde '+origen+' hacia la zona '+destino+'\n';
 str+='A la brevedad le enviaremos informacion al respecto.\n';
 str+='Gracias por confiar en nosotros.\n';
 alert(str);
}
```

Y el evento en el boton que lleva a ejecutar estas rutinas:

```
<input name='Enviar' type=button value='Enviar Datos'
onClick='tomarDatos(nombre,popID(origen),popID(destino),obs)'>
```

Asi, al presionar el boton que envia los datos, la pagina genera un cartel de Alert informando que los datos fueron recibidos y prometiendo una proxima respuesta, previa verificacion de que el usuario haya ingresado un nombre y haya seleccionado los lugares de origen y destino, imprescindibles para saber que debemos enviarle!. Algo a considerar en este ejemplo es que todo esto se ejecuto en el browser, entonces, ningun dato fue enviado al server, asi no puede quedar este ejemplo, ya que estamos perdiendo la informacion ingresada.

Lógica del lado del Web Server

Ahora bien, lo que mas nos interesa en este trabajo es explicar en que consiste un site que utiliza ASP's, asi que ahora pasaremos a agregar logica en el Web Server, que es donde podemos ubicar a las ASP's.

En principio podemos observar que respecto de cómo se escribe el codigo de las ASP's, no hay mayor diferencia con Javascript. Esto es porque trabajamos en JScript, tambien podriamos utilizar VBScript, pero me parece mejor el scripting Jscript, ya que como lenguaje es mucho mas robusto y claro, y ademas se ejecuta mas rapido. Ambos lenguajes son similares en cuanto a sintaxis, objetos y caracteristicas. De esta manera puedo escribir del mismo modo rutinas para ser ejecutadas tanto en el Web Server como en el Browser.

La diferencia radica, justamente, en donde se ejecuta el codigo: el JavaScript viaja al Browser junto con el HTML, y se ejecuta ahí mismo, sin necesidad de que el Web Server se entere de lo que esta sucediendo. JScript, en cambio, se ejecuta en el Web Server, es decir, que cuando la pagina viaja al browser, el codigo ya fue resuelto, y no se puede ver si desde alli vieramos el codigo fuente de la pagina.

Veamos esto en un ejemplo: si abrimos la pagina formJScript.asp podemos ver el siguiente codigo:

Codigo JavaScript:

```
<script language=JavaScript>  //RUTINAS DE JAVASCRIPT (SE ENVIAN AL BROWSER,
JUNTO CON EL HTML)_____________________
function popID(popup){
 for(i=0;i< popup.length;i++){                // recorro todos los elementos del popup
  if(popup.options[i].selected) {return popup.options[i].text;alert(popup.options[i].value)}
 }
 alert(popup.name+' invalido');               // aviso si no hay ningun elemento seleccionado
 return '';
}

function tomarDatos(nombre,origen,destino,obs){
 if (nombre.value==''){alert('Debe ingresar su nombre!');return};
 str= 'Hola '+nombre.value+'\n';
 str+='Entendemos que usted estaria viajando desde '+origen+' hacia la zona '+destino+'\n';
 str+='Confirma estos datos?.\n';
 if (confirm(str)) enviar_al_Web(nombre.value,origen,destino,obs.value)
}

function rep(str,x,y){
 str1=str+'';
 for(i=0;i<str1.length;i++){
  if(str[i]=='%' && str[i+1]!='2')str1=str1.replace('%','%25');
  if(str[i]==' ')str1=str1.replace(' ','%20');
  if(str[i]=='&')str1=str1.replace('&','%26');
  if(str[i]=='#')str1=str1.replace('#','%23');
  if(str[i]==x)str1=str1.replace(x,y)
 }
 return str1
}

function enviar_al_Web(nombre,origen,destino,obs){
 location="formJScript.asp?st=1&nombre="+rep(nombre,'','')+"&origen="+rep(origen,'','')
+"&destino="+rep(destino,'','')+"&obs="+rep(obs,'','')
}
</script>
```

En este ambiente hemos agregado la funcion enviar_al_Web, cuyo objetivo consiste en enviar al Web Server la solicitud de una nueva 'location' o pedido de pagina, enviando a la vez los datos ingresados al formulario.

Codigo JScript:

```
<%       //RUTINAS DE JSCRIPT (SE EJECUTAN ANTES DE ENVIAR LA PAGINA AL
BROWSER, EN EL SERVER)_____________________________
function enviar_Confirmacion(st,nombre,origen,destino,obs){
 if (st==1){
```

```
  serverStr ='Gracias '+nombre+' por utilizar nuestros servicios.<br>';
  serverStr+='Sus datos han sido enviados al depto de informes y a la brevedad le sera
enviada una respuesta.'
 }
}
enviar_Confirmacion(Request('st')+'',Request('nombre')+'',Request('origen')
+'',Request('destino')+'',Request('obs')+'')
%>
```

En este tramo de la pagina encontraremos lo nuevo en relacion a lo que vinimos viendo hasta ahora, este codigo se ejecuta antes de que la pagina sea solicitada, y no viaja al Browser como el codigo que vimos en el apartado anterior. Siempre que llamemos a esta pagina, antes de enviarla al Browser, el Web Server resolvera el codigo encerrado entre los tags '<% %>', en nuestro caso la llamada a la funcion 'enviar_Confirmacion', y escribira el contenido de las variables o llamadas a funcion definidas entre '<%= %>', en el ejemplo <%= serverStr %>.

La primera vez que esta pagina es solicitada (desde index.htm) el valor de 'st' es igual a 0 (la llamada es formulario de pedidos., donde se pasa como parametro de la pagina la variable st con el valor 0 explicitado), conteniendo solo el html de la primera opcion del '<% if (Request('st')==0){ %>'. Una vez que los datos del formulario fueron ingresados, y cuando el usuario presiona el boton de 'Enviar Datos', se ejecuta la funcion de JavaScript que termina con un llamado al Web Server, solicitando esta misma pagina, pero con 'st' conteniendo el valor 1. En este momento, el Web Server vuelve a ejecutar el JScript contenido en la pagina y la devuelve, pero esta vez solo la porcion '<% } else { %>' de la pagina es enviada al browser.

Algo para destacar es el uso de una funcion propia del ambiente llamada 'Request', que se utiliza para recuperar los datos enviados al Web Server. A esta altura deberiamos tener claro que las ASP's, si bien se ejecutan en el Web Server, deben considerarse una aplicación aparte para poder aclarar algunos conceptos, tal es el caso del uso de Request, que permite el paso de los datos que se envian al Web Server al ambiente de las ASP's. Esto tambien lo encontrariamos si trabajaramos con CGI's.

Acceso a Archivos del Sistema Operativo

En el apartado anterior vimos como recuperar datos ingresados por el usuario para luego procesarlo, ahora veremos como podriamos guardar esos datos para un uso posterior, en el ejemplo, podria ser para luego enviar una respuesta; en la practica el uso que le doy a esta opcion es para guardar la clave de acceso a la base de datos (en IIS 3.0 era posible leer el codigo ASP de las páginas y considere muy inseguro dejar alli la clave) y para armar mi propio archivo de log de transacciones y/o errores (el motivo por el cual no guardo estos datos en la base de datos es para el caso de tener problemas en la conexión a la base, para poder tener alguna referencia de lo que puede estar pasando).

Veamos como quedaria la rutina 'enviar_Confirmacion' si le agregamos esta funcionalidad (esto lo tenemos en la pagina llamada jscript2Text.asp):

```
function enviar_Confirmacion(st,nombre,origen,destino,obs){
```

```
 if (st==1){
  var fsObj,arch,pathArch;
  pathArch = 'd:\datos.txt';
  fsObj=Server.CreateObject("Scripting.FileSystemObject");
  arch=fsObj.OpenTextFile(pathArch,8);
  arch.WriteLine(nombre+'\t'+origen+'\t'+destino+'\t'+obs);
  arch.close();
  serverStr ='Gracias '+nombre+' por utilizar nuestros servicios.<br>';
  serverStr+='Sus datos han sido enviados al depto de informes y a la brevedad le sera
enviada una respuesta.'
 }
}
```

Aquí estamos usando objetos propios de JScript para el manejo de entidades propias del sistema de archivos de NT:

- ✓ Server.CreateObject("Scripting.FileSystemObject"): en esta linea estamos creando un objeto dentro del ambiente las ASP's que contendra un objeto del sistema de archivos.
- ✓ OpenTextFile: es un metodo del objeto antes mencionado que permite abrir el archivo indicado en su primer parametro (la informacion detallada de objetos, propiedades y parametros se encuentra en el Apendice B).
- ✓ WriteLine: es otro metodo que permite escribir una linea de texto en un archivo abierto previamente.
- ✓ Close: es el ultimo metodo utilizado, nunca debemos olvidarnos de cerrar correctamente cualquier archivo que abramos!.

Integración con Bases de Datos

Ahora nos acercamos mas al modo de operar de una aplicación real, es decir, guardar la informacion en una base de datos. En nuestro caso usaremos SQL Server, pero desde las ASP's podriamos acceder a cualquier base de datos que permita conexiones via ODBC.

En principio, los elementos que deben existir previamente en la base de datos son:

- ♦ La tabla donde almacenaremos los datos, en nuestro caso se llama pedidosInfo y tiene la siguiente estructura:

```
CREATE TABLE pedidosInfo (
        id int IDENTITY (1, 1) NOT NULL ,
        nombre varchar(100) NOT NULL ,
        origen varchar(50) NOT NULL ,
        destino varchar(50) NOT NULL ,
        obs varchar(255) NULL
) ON [PRIMARY]
GO
```

- Un Stored Procedure que reciba los datos, realice los controles necesarios, grabe los datos en la tabla y retorne un Ok en caso de que los datos hayan sido grabados correctamente. Este ultimo dato no deberia ser necesario si los objetos ADO funcionaran correctamente, ya que se deberia poder chequear la colección de errores del objeto comando, pero como este suele fallar (pierde algunos datos durante la ejecucion del comando, por ejemplo), prefiero asegurarme un determinado retorno a chequear luego de haber enviado el comando; de esta manera me aseguro de que todo funciono del modo esperado:

```
Create proc guardarPedido      @nombre        varchar(100),
                                    @origen                varchar(50),
                                    @destino       varchar(50),
                                    @obs           varchar(255)
as
--Realizo los controles que considere necesarios
if exists (select * from pedidosInfo where nombre=@nombre and origen=@origen and
destino=@destino) begin
        select 'Solicitud ya ingresada' error

RETURN
End

insert pedidosInfo (nombre,origen,destino,obs)
 select @nombre,@origen,@destino,@obs
GO
```

Y en nuestra pagina formJScript2Sql.asp vamos a encontrar lo siguiente:

```
<%
function sql(qry){
 rSt=Server.CreateObject("ADODB.Recordset");
 con=Server.CreateObject("ADODB.Connection");
 con.CommandTimeout = 900;
 con.Open('filedsn=tesis.dsn;user=sa;password=sa;APP=viajes');
 serverStr = qry;
 rSt.Open(qry,con);
 serverStr = rSt('msg').Value;
 rSt.Close();
 con.Close();
}

function enviar_Confirmacion(st,nombre,origen,destino,obs){

 SERVERSTR ='';
 if (st==1)
  sql('guardarPedido "'+nombre+'","'+origen+'","'+destino+'","'+obs+' "')
}

enviar_Confirmacion(Request('st')+'',Request('nombre')+'',Request('origen')
```

```
+'',Request('destino')+'',Request('obs')+'')
%>
```

Veamos cuales fueron las modificaciones:

- A la rutina enviar_Confirmacion la modificamos para que, en caso de que lo que estemos enviando sean datos validos, llame a la funcion sql, y le pasamos como parametro una llamada a un stored procedure de SQL con sus correspondientes parametros,

- Creamos una funcion llamada 'sql', que recibe como parametro un query a ser ejecutado en la base de datos. En esta rutina vamos a encontrar, en su mayoria, comandos ADO:
  - ✓ 'rSt' y 'con' son objetos ADO, el primero es un objeto recordset y el segundo un objeto conexión. Aquí tambien podriamos haber creado un comando, pero no es necesario, asi que siguiendo la ley que dice que 'la mejor linea de codigo es la que no se escribe' tratamos de usar la menor cantidad posible de elementos!.
  - ✓ 'CreateObject', 'CommandTimeout', 'Open' y 'Close' son todos metodos propios de los objetos antes mencionados.
  - ✓ Y por ultimo, en la linea 'serverStr = rSt('msg').Value;' tomamos el valor devuelto por el comando ejecutado y lo asignamos a la variable que luego mostraremos en la pagina.

Como se puede apreciar, nada de esto se ve muy complicado. Lo mas importante es siempre tener claro donde se ejecuta cada porcion de codigo, ya que es imprescindible para poder comprender como se suceden los eventos, donde enviar cada dato, etc. A continuacion pasare a explicar la estructura de una aplicación real, que de hecho esta funcionando en dos empresas.

VI Un modelo de aplicación real

Introducción

Esta aplicación surgio de la necesidad de implementar un sistema para la administracion de fondos de inversión (Consultatio Asset Management). La empresa donde iba a ser implementado se estaba actualizando tecnologicamente, motivo por el cual existia una gran libertad para elegir en que ambiente trabajar, y el objetivo era aplicar tecnologias nuevas; aun cuando todo lo referente a Internet era muy nuevo en el pais (comienzos de 1997), la gerencia de sistemas decidio intentar desarrollar una intranet, a pesar de no tener demasiados conocimientos en el tema.

Como condicion inamovible el Web Server deberia ser el IIS, ya que se apuntaba a homogeneizar plataformas y NT Server se habia impuesto como estandar. Por esta decision tambien se prefirio usar SQL Server, ya que siendo de la misma linea de productos nos aseguraba menos problemas que cualquier otra en terminos de interfaz y seguridad. Asi quedo definida la plataforma a utilizar.

En relacion a conocimientos y experiencia previa, el gerente tenia conocimientos de NT 4.0 y SQL Server 6.5. Y yo contaba con la experiencia de haber participado en el equipo de desarrollo de PC Banking, la aplicación del site del ex Banco Frances (hoy BBV) que permitia realizar operaciones a los clientes a través de Internet. En este caso se habia utilizado la plataforma Macintosh, con WebStar como Web Server, Hypercard para el desarrollo de CGI's que administraban la informacion contenida en los formularios del site y 4D para acceder al host central del banco.

Al comienzo del proyecto tuvimos la intencion de utilizar Visual Interdev para el desarrollo del site, pero al ver el codigo generado decidimos escribir nuestras propias rutinas 'a mano', ya que Interdev no era para nada eficiente ademas de generar demasiados bugs en el codigo, y en última instancia, teniamos que meternos en el codigo igual. Tambien en la primera versión del sistema desarrollamos las ASP's en VBScript, ya que todas las politicas de Microsoft apuntaban a darle mas importancia a este lenguaje de Scripting que a JScript, ademas en esa epoca JavaScript estaba naciendo y no era seguro que se estandarizara como finalmente sucedio. Al escribir la segunda versión del sistema preferimos este lenguaje, que ya se veia mas solido y traia unas cuantas ventajas sobre VBScript.

A continuacion paso a explicar como se diseño la aplicación y los elementos que la componen.

Definicion del Modelo

Un objetivo a alcanzar con el desarrollo de la segunda versión fue lograr que la aplicación fuera totalmente parametrizable, para poder aplicarla a futuro en otra empresa (debido a que ya teniamos la propuesta de desarrollar una intranet para un banco). Asi, tratamos de abstraernos de la problemática en si del negocio para llegar a detectar elementos comunes en aplicaciones que se puedan aplicar a cualquier otro tema.

Para comenzar, podemos simplificar la vision externa de cualquier sistema como un conjunto de pantallas con dos funciones basicas:

- ✓ Permitir el ingreso de informacion al sistema
- ✓ Mostrar informacion contenida y/o procesada por el sistema

Asi, desde el punto de vista de la interfaz, podemos pensar en dos 'templates' genericos que cubran cada una de estas necesidades: Formularios y Listados.

Dentro de la aplicación, podemos hablar tambien de un conjunto generico de actividades que se deben llevar a cabo, y podemos aprovechar la clasificacion anterior para organizarlas:

Para el ingreso de la informacion necesitamos como minimo realizar las siguientes acciones:

- ✓ Validar el ingreso de datos,
- ✓ Confirmar que los datos ingresados cumplan con las relaciones existentes entre estos y las demas entidades del sistema,
- ✓ Almacenar la informacion ingresada en caso de ser correcta,
- ✓ Informar al usuario que la operación se ha realizado con éxito o si ha surgido algun error.

Y para mostrar informacion contenida en el sistema debemos como minimo:

- ✓ Recibir un requerimiento de informacion,
- ✓ Verificar que los criterios del pedido sean coherentes,
- ✓ Hacer la selección y/o procesamiento correspondiente de datos en base a la consulta realizada
- ✓ Devolver el conjunto de datos de respuesta, en caso de existir, o un mensaje de aviso al usuario de que no se encontraron datos, en caso contrario.

Una vez detectadas las acciones a realizar, quedaba por ver como realizar cada paso dentro de los ambientes que disponiamos, estos eran: páginas HTML, codigo de scripting que se ejecuta en el Browser, scripting en el Web Server y tablas, vistas y stored procedures en SQL Server. Finalmente, el esquema obtenido fue el siguiente:

- ✓ En el ingreso de datos:

Validar el ingreso de datos	JavaScript en el Browser
Confirmar la integridad de los datos	JSCript para conectarse con SQL y Stored Procedures en SQL Server
Almacenar la informacion	Tablas en SQL Server (accedidas desde los stored procedures)

Informar el resultado al usuario	JScript para generar páginas HTML

✓ En la solicitud de informacion:

1612

Recibir el requerimiento	JScript para conectarse con SQL
Verificar coherencia del pedido	Stored Procedures en SQL Server
Seleccionar y/o procesar los datos	Stored Procedures en SQL Server
Mostrar la informacion	JScript para generar el HTML

En base a esto, terminamos teniendo, a grandes rasgos, dos ambientes de desarrollo que se deberian comunicar entre si:

Las páginas que conforman el site, donde se incluye:

- ✓ El HTML que las compone,
- ✓ El JavaScript que se ejecutara en el Browser,
- ✓ El JScript que se ejecutara en el Web Server.
- ✓ El codigo ADO por medio del cual la aplicación contenida en las páginas se comunicara con el SQL Server.

SQL Server, conteniendo:

- ✓ Las tablas donde se almacenaran los datos,
- ✓ Los stored procedures, a través de los cuales se realizara el envio y recepcion de la informacion, y que principalmente contendran la logica del negocio aplicada en el procesamiento de los datos.

Consideraciones de Seguridad

Antes de seguir explicando el modelo considero importante explicar porque las cosas se hicieron de esta manera, y muchos de los factores tuvieron relacion con la seguridad, un tema muy delicado en las tecnologias relacionadas con Internet, en principio por la caracteristica publica de la red pero ademas porque la mayoria de las herramientas utilizadas son muy nuevas y no estan lo suficientemente probadas como para poder confiar totalmente en ellas.

- Las ASP's contienen un conjunto de elementos incorporados para facilitar su uso, y de hecho son muy utiles en su mayoria. Uno de esos objetos, que como idea parece muy util pero no utilizo es un archivo llamado 'global.asa', que contiene un conjunto de sentencias que permitirian inicializar y administrar la aplicación y las sesiones de cada usuario. En la versión 4.0 este archivo no cumple correctamente su funcion, motivo por el cual considere mas seguro incluir rutinas propias que me permitan cumplir estas funciones (son las funciones iniApp e iniSession, que seran explicadas dentro del contenido de las páginas que conforman el site).

- Utilizo un archivo (ubicado fuera del arbol de directorios al que tiene acceso el Web Server) para guardar el string de datos que me permiten conectarme con la base de datos, debido a que, entre otras cosas, debo enviar en este string un nombre de usuario, password, nombre de una base de datos, etc., y si estos datos estuvieran escritos dentro del codigo cualquiera

que pudiera acceder (de manera inesperada) al codigo de las páginas tendria todos los datos necesarios para acceder a la base de datos.

- La logica del negocio siempre estara del lado del SQL Server, nunca en las páginas. Esto surgio en principio por el poco convencimiento de la utilizacion de las ASP's, y no asi del SQL Server; sabiamos que era mas probable querer cambiar las primeras, y para tal efecto seria mas simple si dejabamos lo menos posible a resolver de ese lado. Por otra parte, todo lo que sean reglas del negocio consiste basicamente en almacenamiento y procesamiento de informacion, y si esta se encuentra en el SQL Server es esperable que el medio mas eficiente para manipularla sea este mismo. Finalmente, cuando decidimos pasar a una segunda versión pudimos confirmar que fue una buena determinacion, ya que el codigo a migrar fue minimo!.

Funcionalidad de las Páginas

Organización

Las páginas que conforman la aplicación son las siguientes:

- Index.htm: es el portal o pagina de inicio del site, donde se ubica el logo de la empresa y las caracteristicas de publicidad que se consideren necesarias, asi como avisos a todos los usuarios del sistema, etc.

- Default.asp: es la pagina de definicion de frames.

  - ✓ Menu.asp: es la pagina que se muestra en el frame superior de la ventana, y contendra links a las distintas aplicaciones contempladas por la intranet, de acuerdo al perfil del usuario que se conecte.

  - ✓ Hide.asp: esta pagina se muestra debajo de la anterior, solo contiene los datos del usuario y la fecha, pero contendra tambien las rutinas Javascript que sean necesarias para la logica del lado del browser.

  - ✓ SMenu.asp: es la pagina que se muestra en el frame izquierdo de la ventana, y contiene items relacionados a la aplicación a la cual haya entrado el usuario.

  - ✓ Body.asp: es la pagina central de la ventana, donde se mostraran las distinas pantallas que conforman la aplicación (edicion de datos mediante formularios, listados, etc.).

  - ✓ Fijos.asp: es la pagina que se muestra al pie de la ventana, y contiene links a prestaciones generales del site, que no dependen del perfil del usuario que se conecte.

- Base.inc: es una pagina que no se muestra a través del browser, sino que contiene las rutinas basicas que seran ejecutadas por el Web Server.

- App.inc: es similar a la anterior, pero contiene las rutinas propias de la aplicación, es decir, que recibe los datos enviados desde el browser, los procesa y envia una pagina de resultado.

Contenidos

Como se pudo ver en la seccion anterior, el site esta conformado por frames, dentro de los cuales se muestra informacion a medida que el usuario va accediendo a los distintos links. Ahora veremos como interactuan los contenidos de las páginas antes mencionadas.

Default.asp

```
<!-- #INCLUDE VIRTUAL="app.inc" -->
<% /*a_('ini',false);*/iniApp() %>
<html>
<script language=JavaScript>
innerWidth=750;
innerHeight=500;
</script>
<head><title>Mercobank</title></head>
<frameset rows="60,40,*,100" framespacing=0 frameborder=no border=0 marginheight=0 marginwidth=0 noresize>
        <frame name=menu src=menu.asp scrolling=no marginheight=3 marginwidth=3 noresize>
        <frame name=varios src=varios.asp scrolling=no marginheight=3 marginwidth=3 noresize>
        <frameset cols='125,*' framespacing=0 frameborder=no border=0 marginheight=2 marginwidth=2 noresize>
                <frame name=smenu src=smenu.asp scrolling=auto marginheight=3 marginwidth=3 noresize>
                <frame name=body src=body.asp scrolling=auto marginheight=3 marginwidth=3 noresize>
        </frameset>
        <frame name=fijos src=fijos.asp scrolling=no marginheight=3 marginwidth=3 noresize>
</frameset>
</html>
```

Veamos que elementos vale la pena destacar aquí:

- <!-- #INCLUDE VIRTUAL="app.inc" -->
  En esta linea estamos indicando que a dicho codigo ASP se incluya el contenido de la pagina mencionada (en este caso app.inc). Para ser exactos deberiamos incluir todas las páginas '.inc' a las que deba acceder el IIS, pero en este caso no es necesario incluir tambien base.inc, ya que esta esta incluida a partir de app.inc.

- <% /*a_('ini',false);*/iniApp() %>
  Aquí estamos incorporando codigo que ejecutara el Web Server antes de enviar esta pagina al Browser. Lo que hacemos es llamar a la rutina iniApp(), que inicializa las variables globales de la aplicación si se esta ejecutando por primera vez (el comentario '/*a_('ini',false);*/' se encuentra aquí para inicializar la variable que indica si es o no la primera vez que se esta inicializando la aplicación, y este codigo se utiliza durante el desarrollo para incializar la aplicación en caso de que un cambio de codigo asi lo requiera), e inicializa las variables de la sesion de un nuevo usuario, cada vez que se conecte al sistema.

- <script language=JavaScript>
  innerWidth=750;
  innerHeight=500;
  </script>
  Aquí estamos incluyendo codigo que se ejecutara en el Browser, en este caso en particular solo estamos indicando el tamaño default de la ventana, en las páginas siguientes veremos rutinas Javascript mas interesantes.

  Lo demas es codigo HTML que define los frames y sus respectivas caracteristicas.

Menu.asp

```
<!-- #INCLUDE VIRTUAL="app.inc" -->
<html>
<body background='../img/fondos/menu.jpg'>
<table border='0' width='750'>
<tr>
 <td width='120' align=left valign=top>
  <a href='index.htm' target='_top'>
   <img src='../img/menu/mbk10.jpg' border='0'>
  </a>
 </td>
 <td width='650' align=center valign=middle><%= s('s_Menu') %></a></td>
</tr>
</table>
</body>
</html>
```

Aquí lo unico por destacar es el tag '<%= s('s_Menu') %>', cuya funcionalidad es mostrar, en ese lugar del codigo HTML el contenido de la variable de sesion llamada 's_Menu', la cual adquiere determinado valor cuando se ejecuta la funcion iniApp. Como vemos, esta pagina actua como un template fisico, es decir, su contenido sera variable y dependera del estado de la aplicación. Al ingresar al site, la variable 's_Menu' estara vacia, y luego de que el usuario se conecte contendra links a las distintas aplicaciones que soporte la intranet.

A esta altura deberia explicar algunas cuestiones que hacen a la nomenclatura de los objetos en JScript:

- Las variables que seran compartidas por todos los usuarios de la aplicación durante su ejecucion comenzaran con los caracteres 'a_', como por ejemplo 'a_db' contendra el nombre de la base de datos a la que se conectara la aplicación para obtener y almacenar datos.

- Las variables que se generaran para el uso de cada usuario, es decir, de sesion, comenzaran con los caracteres 's_', por ejemplo la variable 's_User' contiene el nombre del usuario que se encuentre conectado al sistema. Las ASP's generaran una instancia de cada una de estas variables por cada usuario que acceda, administrandolas de manera independiente.

- Las variables globales (similares a las de aplicación, con la diferencia de que no se mantienen luego de cada ejecucion que realice el Web Server sobre las páginas) se distinguen con los caracteres 'g_', como ejemplo podemos mencionar la variable g_fmtList, que en realidad es un array de formatos para los listados que muestren informacion requerida a la base de

datos. Esta informacion no es necesario mantenerla constantemente en memoria, ya que son valores constantes y ya se encuentran explicitados en las páginas, con lo cual es suficiente con que esten disponibles en ellas para ser accedidos cada vez. En la practica, el uso de estas variables se debe a que necesitabamos objetos de tipo array y ni las variables de sesion ni las de aplicación nos brindaban esta facilidad.

Hide.asp

```
<!-- #INCLUDE VIRTUAL="app.inc" -->
<html>
<script language=JavaScript>
tit='';nw='';pexc=0;foldersTree = '';
function optChk(opt,s){return (opt==1 && s=="")}
function fok(s,w){
 y=s.substr(0,2);
 m=s.substr(2,2);
 d=s.substr(4);
 ok=(y>=0 && y<=99 && m>=1 && m<=12 && d>=1 && d<=31);
 if(ok && w==1){
  dd=new Date(y,m-1,d);
  dw=dd.getDay();
  ok=(dw!=0 && dw!=6)
 }
 return ok;
}
function c(){
 fields="";
 for(ci=0;ci<c.arguments.length;ci++){
  if(c.arguments[ci]=="")return false;
  fields+=c.arguments[ci]+","
 }
 return fields.substr(0,fields.length-1)
}
function u(x){y = x+'';return (y=='undefined' || y=='null' || y==' ')?'':y}
function a(x){alert(x.name+"  invalido."); x.focus();x.select()}
function q(x){return "'"+x+"'"}
function qd(x){return '"'+x+'"'}
function popID(x){
 for(m=0;m<x.length;m++){
  if(x.options[m].selected){
   if(!tit.match(x.options[m].text) && x.name!='ft')
    tit+=x.options[m].text+' - ';
   return x.options[m].value
  }
 }
 a(x);
 return false;
}
function b(x){return (x.checked)?1:"0"}
function z(x,opt){
 s=x.value;
 if(optChk(opt,s)) return "null";
 if(s!="") return q(u(s));
 a(x);
 return false
}
function n(x,opt){
 s=x.value;
 if(optChk(opt,s)) return "null";
```

```
 if(s>0) return rep(s,",","");
 a(x);
 return false
}
function np(x,opt){
 s=x.value;
 if(optChk(opt,s)) return "null";
 if(x.value<=0){
  alert(x.name+' debe contener un valor positivo');
  return false
 }
 n(x,opt)
}
function nt(x,opt){
 s=x.value;
 if(optChk(opt,s)) return "null";
 if(x.value<=0 || x.value > 100){
  alert(x.name+' debe contener un valor porcentual');
  return false}n(x,opt)
 }
 a(n);
 return false
}
function f(x,opt,w){
 s=x.value;
 if(optChk(opt,s)) return "null";
 if(fok(s,w)){
  tit+=s+' - ';
  return q(s)
 }
 a(x);
 return false
}
function fw(x,opt){return f(x,opt,1)}
function p(x,opt){
 s=popID(x);
 if(s.toLowerCase()=="null")s="";
 if(optChk(opt,s)) return "null";
 if(s!="") return s;
 a(x);
 return false
}
function pi(x,opt){
 return (x.type=="text" || x.type=="hidden")?n(x,opt):p(x,opt)
}
function cambiarGif(obj,cual,frame){
 alert("../img/"+frame+"/"+parent.frame.obj.name+cual+".gif");
 obj.src="../img/"+frame+"/"+obj.name+cual+".gif"
}
function env(fu,pa){
 if((pa+'').search('false')!=-1)return;
 pa=rep(pa+'','x','x');
 tit=rep(tit,' ','%20');
 parent.body.location="body.asp?func="+fu+"&pars="+pa+"&tit="+tit+"&nw="+nw
}
function rep(str,x,y){
 str1=str+'';
 for(i=0;i<str1.length;i++){
  if(str[i]=='%' && str[i+1]!='2')str1=str1.replace('%','%25');
  if(str[i]==' ')str1=str1.replace(' ','%20');
  if(str[i]=='&')str1=str1.replace('&','%26');
```

```
  if(str[i]=='#')str1=str1.replace('#','%23');
  if(str[i]==x)str1=str1.replace(x,y)
 }
 return str1
}
//Arbol de SMENU_____________________________
function generateTree(){
 items = new Array(<%= s('s_itemsArbol') %>);
 padre = new Array(<%= s('s_padreArbol') %>);
 aux = '';
 niveles = ('<%= s("s_cMod")%>'=='IBanking')?0:1;
 if (items.length==0) foldersTree = '';
 else {
  foldersTree = folderNode('<%= s("s_cMod")%>',niveles);
  for (i=0;i < items.length;i++)
   if (niveles==0)
    if (padre[i]!=0) aux = appendChild(foldersTree,leafNode(items[i]))
    else appendChild(aux,items[i])
   else appendChild(foldersTree,items[i]);
 }
}//_____________________________________
function folderNode(name,niveles){return new Array(0,0,niveles,name)} function leafNode(name){return new
Array(0,0,1,name)}
function appendChild(parent,child){return parent[parent.length]=child}
function redrawTree(){
 var doc = top.smenu.window.document;
 doc.clear();
 doc.write("<body background=../img/fondos/smenu.jpg text=#FFFFFF link=#FFFFFF alink=#FFFFFF
vlink=#FFFFFF>\n");
 doc.write("<table border=0 cellspacing=2 cellpadding=1>\n");
 if (foldersTree!='') redrawNode(foldersTree,doc,0,1,"");
 doc.write("</table>\n");
 doc.close();
}//_____________________________________
function redrawNode(foldersNode,doc,level,lastNode,leftSide){
 var j=0;var i=0;spc="";spc6='<%= spc(3) %>';
 if (level>0) spc='<%= spc(2) %>';
 displayIconAndLabel(foldersNode,doc,spc);
 if (foldersNode.length > 4 && foldersNode[0]){
 if (!foldersNode[2]){
  level+=1;
  for (i=4; i< foldersNode.length;i++)
    if(i==foldersNode.length-1){
      redrawNode(foldersNode[i], doc, level, 1, leftSide)}
    else{
       redrawNode(foldersNode[i], doc, level, 0, leftSide)}
  }else{
   doc.write("<tr>\n\t<td>");
   for (i=4; i< foldersNode.length;i++)
       doc.write("\n\t\t"+foldersNode[i]+"<br>");
   doc.write("\n\t</td>\n</tr>\n");
  }
 }
}//_____________________________________
function displayIconAndLabel(foldersNode,doc,spc){
 img=(foldersNode[1])?"open":"closed";
 doc.write("<tr><td nowrap>\n\t"+spc+"<A href='javascript:parent.varios.openBranch(\"" + foldersNode[3] +
"\")'><img src='../img/tree/"+img+"folder.gif' width=22 height=22 border=0><img
SRC='../img/smenu/"+foldersNode[3]+"off.gif' border=0></a>\n\t</td>\n</tr>\n");
}//_____________________________________
function closeFolders(foldersNode){
```

```
 if (!foldersNode[2])
   for(i=4;i< foldersNode.length;i++)closeFolders(foldersNode[i]);
 foldersNode[0] = 0;
 foldersNode[1] = 0;
}//_____________________________________
function clickOnFolderRec(foldersNode, folderName){
 if (foldersNode[3]==folderName){
  if (foldersNode[0])
   closeFolders(foldersNode);
  else{
   foldersNode[0]=1;
   foldersNode[1]=1;
  }
 }else{if(!foldersNode[2])
   for(i=4; i< foldersNode.length; i++)
     clickOnFolderRec(foldersNode[i], folderName)}
}//_____________________________________
function openBranch(branchName){
 clickOnFolderRec(foldersTree, branchName);
 if ('<%= s("s_cMod") %>'=='IBanking') parent.body.location.href="../ibanking/"+branchName+".htm";
 timeOutId = setTimeout("redrawTree()",100);
}//_____________________________________
function initializeTree(){
 generateTree();
 redrawTree();openBranch('<%= s("s_cMod") %>')
}
initializeTree();
</script>

<body background='../img/fondos/varios.jpg' text='#0000A0' link='#0000A0' alink='#0000A0' vlink='#0000A0'>
<form name=body action=body.asp method=POST>
<table border='0' width='750'>
<tr>
 <td width='115' align=left valign=middle>
  <a href="../ibanking/btel.htm" target=body>
   <img src=../img/publicidad/telefono.gif border=0>
  </a>
 </td>
 <td width='200' align=left valign=middle>
  <%= lk(s('s_UserD'),'body.asp?func=home','body') %>
 </td>
 <td width='202' align=center valign=middle> </td>
 <td width='203' align=center valign=middle><%= s('s_listAct') %></td>
 <td width='30' align=right valign=middle><%= s('s_hoy') %></td>
</tr>
</table>
</form>
</body>
</html>
```

Esta pagina, aunque contiene mucho mas codigo Javascript, es basicamente similar a la anterior: consta de un conjunto de rutinas que se ejecutaran del lado del Browser, el template de la pagina en HTML y las variables JScript que completaran con los valores correspondientes los lugares que ocupan.

Lo que puede resultar interesante en esta pagina es que a partir de ella se escribira en SMenu.asp el arbol con los items correspondientes a cada aplicación. Para esto comenzamos observando que dentro de los tags de '<SCRIPT></SCRIPT>', ademas de las funciones, hay una linea que llama a la funcion initializeTree, esto significa que esta funcion se ejecutara cada vez

que se acceda a la pagina. Dicha funcion tiene como objetivo generar y mostrar en el frame smenu un arbol con items correspondientes a la aplicación que el usuario haya seleccionado (para esto, este frame debera refrescarse cada vez que el usuario ingrese a una aplicación diferente). Las demas rutinas de esta pagina son accesorias para esta funcion, excepto 'optChk, fok, c, u, a, q, qd, popID, b, z, n, np, nt, f, fw, p, pi y rep' que son rutinas usadas desde los campos de los formularios para hacer validaciones muy basicas sobre los datos ingresados por el usuario, 'env' que se utiliza para enviar los datos desde las páginas al Web Server, y 'cambiarGif' que modifica el nombre de la imagen a mostrar en caso de que el usuario pase por encima de la misma con el mouse (es decir, cubre la funcion de rollover). La definicion de cada una de estas funciones con sus respectivos parametros se encuentra en el apendice E.

SMenu.asp

```
<html><body background=../img/fondos/smenu.jpg></body></html>
```

Esto se debe a que solo existe como template fijo para este frame, su contenido es generado cada vez desde la pagina varios.asp, y depende de la aplicación que seleccione el usuario.

Body.asp

```
<!-- #include virtual="app.inc" -->
<html>
<% root(Request('func')+'',Request('pars')+'',Request('tit')+'',Request('nw')+'') %>
<script language=JavaScript>
<%= s('s_script') %>
</script>

<body background=../img/fondos/body.jpg text=#0000A0 link=#008040 alink=#800040
vlink=#800040 onLoad="if (document.forms[0].length)
document.forms[0].elements[0].focus();<%= s('s_msg') %>">

<form name=body action=body.asp method=POST>
<%= s('s_titBody')+'<br>'+s('s_layBody') %>
</form>
<%  s_('s_msg') %>
</body>
</html>
```

Esta pagina puede entenderse como pantalla principal del sistema, ya que aquí se mostraran los formularios para ingresar informacion, las páginas de texto, los listados, etc. Toda esta informacion estara contenida en la variable 's_layBody'. Tambien aquí se ubicaran algunas rutinas adicionales de JavaScript (en 's_cript'), que se generaran junto con las demas variables en caso de ser necesarias.

Algo que vale la pena destacar aquí es la llamada a la funcion root. Cada vez que el usuario presione un boton o clickee un link, el Browser estara realizando una llamada al Web Server,

esperando como resultado una pagina a mostrar en lugar de esta; que para nuestra aplicación sera siempre la misma pagina, ya que solo varian los contenidos de las variables que el Web Server genera. Asi, ante alguna de las acciones antes mencionadas, el Web Server recurrira a esta pagina, ejecutara root() y al retorno de esta funcion las variables contendran los valores requeridos por el usuario.

Por ultimo podemos ver que en el tag '<BODY>' estamos indicando una accion que se ejecutara al cargar la pagina:

onLoad="if (document.forms[0].length) document.forms[0].elements[0].focus(); <%= s('s_msg') %>

Con esto estamos diciendo que, en principio, si la pagina contiene un formulario se debe posicionar el cursor en el primer campo de este, y ademas estamos mostrando el contenido de la variable 's_msg', la cual se carga con un mensaje de alerta si hubo algun problema con el envio de datos al Web Server o a la base de datos. Luego, esta variable se limpia, tambien durante la carga de la pagina, para que no quede continuamente mostrando la ventana de alerta.

Fijos.asp

```
<!-- #INCLUDE VIRTUAL="app.inc" -->
<html>
<body bgcolor=#FFE322 text=#004080 link=#800040 alink=#800040 vlink=#800040>
<form name=body action=body.asp method=POST>
<center><%= s('s_Fijos') %></center>
</form>
</body>
</html>
```

Aquí como vemos no hay nada nuevo, solo el template de la pagina en HTML y la incorporacion de la variable 's_Fijos', que contiene los links a las aplicaciones generales del sistema.

App.inc

```
<!-- #INCLUDE VIRTUAL="base.inc" -->
<% function root(f,p,t,nw){
/* Raiz de la aplicacion, aqui llega cada pedido realizado x el browser y deriva
                                        f: sProc a ejecutar en sql
                                        p: parametros del sProc
                                        t: titulo de la pagina siguiente
                                        nw: abrir la pagina siguiente en una ventana nueva */
 clear();f=u(f);p=u(p);fp=f+' '+p;
 switch (f){
  case 'webini':      admin(fp);                                  break;
  case 'webLogin':  sessionIni(fp);                 break;
  case 'webLogout':sessionIni('');                  break;
  case 'webChPass':             pass(fp,1,'ChPass');                        break;
  case 'webPassCh':             pass(fp,2,'home');              break;
  case 'Find':                  find(f,p,t,1,nw);               break;
  case 'pin':                   find(f,p,'',1,'Generar Pines');break;
```

```
  case 'List':                  list(fp,nw);                                break;
  case 'comboInp':  comboDraw(p,'');            break;
  case 'CliFind':   comboDraw('s_popcli',fp);   break;
  case 'lisFind':   comboDraw('s_poplis',fp);   break;
  case 'AbmPops':   abm(fp,p,1);                        break;
  case 'Abm':                   abm(fp,p,2);                                break;
  case 'newMod':    modIni(p,nw);               break;
  case 'Load':      modLoad(fp,0);              break;
  case 'Imp':                   modLoad(fp,1);              break;
  case 'reLoad':    modAct(fp,1);               break;
  case 'edClose':   modClose(f,p);              break;
  case 'Save':      modSave(f,p);               break;
  case 'newPag':    layouts(p);                         break;
  default:          layouts('home');
 }
}//_____________________________________________________
function admin(fp){sql(fp,3);layouts('Admin')}
function sessionIni(fp){
if (fp!='')
        {sql(fp,2,'c')}
else
        {iniSession()};

s_('s_script',"parent.menu.location.href='menu.asp';parent.fijos.location.href='fijos.asp';parent.varios.location.hre
f='varios.asp';");
        if (s('s_msg')!='') return;
        layouts('home')
}//_____________________________________________________
function pass(fp,step,lay){
 sql(fp,((step==1)?2:0),'c');
 if(s('s_msg')!='')return;
 layouts(lay)
}//_____________________________________________________
function modIni(p,nw){layouts(p,1,nw)}
function modLoad(fp,imp){
 s_('s_toImp',u(imp));
 sql(s('s_cMod')+fp.replace('Imp','Load'),2,'f');
 layouts(s('s_cMod')+'Abm',1)
}//_____________________________________________________
function modAct(fp,first){
 sql(fp.replace('reLoad',s('s_cMod')+'Save'),2,'f');
 layouts(s('s_cMod')+'Abm',u(first))
}//_____________________________________________________
function modClose(f,p){modCS(f,p)}
function modSave(f,p){
 f=((s('s_cMod')=='Admin' || s('s_cMod')=='Contactos')
                                ?f+s('s_tabla')
                                :s('s_cMod')+f);modCS(f,p)
}//_____________________________________________________
function modCS(f,p){
 if(s('s_cModBack')=='idx')
  sql(f+' '+p);layouts(s('s_cMod'),1)
 else
  find(f,p)
}//_____________________________________________________
function find(f,p,tit,first,docs){
 fp = f+' '+p;format='';
 if (u(docs)!=''){
  format='doc';
  det='';
  tit=docs;
```

```
 s_('s_cModQry',fp+((tit=='Contactos')?',+s('s_User'):''))
 }
 else {
  if(u(first)==1){
   s_('s_cModQry',((s('s_cMod')=='v_cobro')?'':s('s_cMod'))+fp);
   s_('s_cModBack','list');
   s_('s_nList',((p.indexOf(',')==-1)?p:p.substr(0,p.indexOf(','))))
  }
  else{sql(fp);if(s('s_msg')!='')return}
  det=tit;tit=s('s_cMod');
 }
 sql(s('s_cModQry'),1,((s('s_cMod')=='v_cobro')?'doc':format));
 if(s('s_msg')!='')return;
 switch (s('s_cMod')+s('s_nList')){
  case 'Racing22':
s_('s_listAct',bot('Preparar','parent.varios.env',"'List',parent.body.c(parent.body.chkID(),'1',"+s('s_User')+")"));
   jsF('chkID');
   Session('s_script')+='parent.varios.location.href="varios.asp";';
   break;
  case 'Racing4':
   s_('s_listAct',bot('Generar
Certificado','parent.varios.env',"'List',parent.body.c(parent.body.chkID(),'2',"+s('s_User')+")"));
   jsF('chkID');
   Session('s_script')+='parent.varios.location.href="varios.asp";';
   break;
  default: s_('s_listAct');
 }
 templB(tit,'list',u(det)+s('s_tbl'));
}//_____________________________________________________
function list(fp){
 sql(s('s_cMod')+fp+','+s('s_User'));if(s('s_msg')!='')return;
 layouts(s('s_cMod'));
}//_____________________________________________________
function layouts(templ,modo,modoLinks){          /* Armado de paginas a retornar al browser
                                                templ: nombre del template
                                                modo: 1,''; 1: init de la pagina */
 layBody(templ,modo);
 if (templ=='Listados' || templ=='Normas' || templ=='Contactos' || templ=='IBanking'){
  sql('subMenu '+templ,5,modoLinks);
  s_('s_script','parent.varios.location.href="varios.asp";');
  s_('s_itemsArbol',g_itemsArbol);
  s_('s_padreArbol',g_padreArbol);
 }
 else{Session('s_script')+='parent.smenu.location.href="smenu.asp";'}
}//_____________________________________________________
function layBody(templ,modo){
 switch (templ+''){
  case 'home':
            lay=(!s('s_ss'))?row2A('Usuario',inp('usr','',8,'Blur','pass.focus()',''))
+row2A('Password',inpB('pass','','password',8,'Change','parent.varios.env',"'webLogin',parent.varios.c(parent.vario
s.z(usr),parent.varios.z(pass))"))
            :row1C(lkForm('Desconectarse','webLogout',''))+row1C(lkForm('Cambiar
Password','webChPass',qs(s('s_User'))))+((s('s_cumples')!='')?row1C('Cumplea&ntilde;os del mes: ')
+row1C(s('s_cumples')):'');
            templB('mbk','idx',lay);break;
  case 'err': templB('mensaje','vacia');break;
  case 'ChPass':
            lay=row2A('Anterior',inpB('ant','','password',8,'Blur','pn.focus',''))
            lay+=row2A('Nueva',inpB('pn','','password',8,'Blur','pnc.focus',''))

lay+=row2A('Confirma',inpB('pnc','','password',8,'Change','parent.varios.env',"'webPassCh',parent.varios.c("+s("s_
```

```
User")+",parent.varios.z(ant),parent.varios.z(pn),parent.varios.z(pnc))"));
            templB('Cambio de Password','idx',lay);break;
 //__________INDEX_________________
 case 'Admin':
            combo('s_poplis',0,'','lisFind');
            lay =row1L(b('Parametros del Sistema'));
            lay+=row2A(spc(3),lkAbm('App','App')+lkAbm('Permisos','Permisos')+lkAbm('Formularios','Form')
+lkAbm('Estados','St'));
            lay+=row1L(b('Tablas Base'));
            lay+=row2A('',lkAbm('Banco','Bco')+lkAbm('Tipo_DGI','DgiT')+lkAbm('Tipo_Doc','DocT')
+lkAbm('IBank','IBank')+lkAbm('Provincias','Pcia'));
            lay+=row2A('',lkAbm('Sucursales','Sucursal')+lkAbm('Gerencias','Gerencia')+lkAbm('Oficinas','Oficina')
+lkAbm('Empleados','Emp'));
            lay+=row1L(b('App Fijas'));
            lay+=row2A('',lkAbm('Listados','Listado')+lkAbm('Sistemas','Sistema')+lkAbm('Normas','Norma')
+lkAbm('Tipos de Normas','NormaT')+lkAbm('Contactos','Contacto'));
            lay+=row1L(b('F.Racing'));
            lay+=row2A('',lkAbm('Clientes','FRcli')+lkAbm('Origen','FRorigen'));
            lay+=row1L(b('SOS'));
            lay+=row2A('',lkAbm('Equipos','sosEquipo')+lkAbm('Impacto','sosImpacto')
+lkAbm('Objeto','sosObjeto')+lkAbm('Prioridad','sosPrioridad')+lkAbm('Problemas','sosProblem'));
            lay+=row1L(b('Generar Pin:')+'
Rango:'+inp('desde','',6)+inp('hasta','',6)+bot('Generar','parent.varios.env',"'pin',c(n(desde),n(hasta))"));
            lay+=hr();
            lay+=row1C(lkForm('Recargar Popups','webini')+spc(3)+lkJs('Consultar Ediciones','Find','Ed'));
            lay+=row1C(spc()+s('s_dbg'));
            templB('Administracion','idx',lay,templ);break;
 case 'Racing':

sql('rcapAnt',4,'s_frcapfh','Change',"parent.varios.env('Find',c(11,null,null,null,null,null,null,q(p(s_frcapfh))))");
            combo('s_popcli',0,'','CliFind');
            lay=layIdx();
            lay+=tFh('Alta','fh')+row2A('Cliente',posPopS('s_popcli',''),' Origen ',a('a_FRorigen'));
            lay+=row2A('Importe',inp('imp'),'Nro',inp('nro'));
            lay+=row2A('Estado',str2Pop('!null|!1|Pagado!2|Impago','st'),chk('total'),'Total');
            lay+=row2A('Capit.:  % Int',inp('cint'),'% Gastos',inp('cgs')+chk('Grabar'));
            lay+=row2A('Listado',ar2Pop('list','Change','if (p(list,1)!=\'null\')
parent.varios.env',"'Find',c(p(list),"+qs(s('s_User'))
+",pi(s_popcli,1),p(a_frorigen,1),np(imp,1),np(nro,1),p(ft,1),f(fd,1),f(fh,1),n(cint,1),n(cgs,1),p(st,1),b(total),b(Grabar))",
'','Detalle Operaciones','Totales x Cliente','Capitalizacion','Marcar Certificados'));
            lay+=row2A('Capit.Anteriores',s('s_frcapfh'));
            lay+=row1L(b('Formularios Unipago'));
            lay+=row1L(spc(5)+lkForm(b('Generar Formulario Unipago'),'Imp',c('0',s('s_User'))));
            lay+=row1L(spc(5)+lkForm(b('Marcar para Imprimir Unipago'),'Find',c('22',s('s_User'))));
            lay+=row1L(b('Certificados'));
            lay+=row1L(spc(5)+lk(b('Imprimir Certificados'),'../docs/Certificados%20Racing.doc','',''));
            templB(templ,'idx',lay,templ);break;
 case 'SOS':
            lay= layIdx();
            lay+=tFh('Alta','Sop.Externo','Derivacion','Finalizacion');
            lay+=row2A('Soporte Asignado',a('a_soporte'));
            lay+=row2A('Categoria',a('a_problem'),'Equipo',a('a_equipo'));
            lay+=row2A('Prioridad',a('a_prior'),'Objeto',a('a_obj'));
            lay+=row2A('Soporte Externo',inp('se'),'Impacto',a('a_impacto')+' Estado '+a('a_st'));
            lay+=row2A('Listado',ar2Pop('list','Change','if (p(list,1)!=\'null\')
parent.varios.env',"'Find',c(p(list),"+s('s_User')
+",p(ft,1),f(fd,1),f(fh,1),p(a_problem,1),p(a_equipo,1),p(a_prior,1),p(a_obj,1),z(se,1),p(a_impacto,1),p(a_soporte,1),p(a
_st,1))","','Detalle Operaciones'));
            templB(templ,'idx',lay,templ);break;
 case 'v_cobro':
            lay =row1L(b('Buscar Plazas, Plazos y Precios'));
```

```
            lay+=row2A('Codigo
Postal',inpB('cp','','text',5,'Change','parent.varios.env',"'Find',c('v_cobro','null,'+z(cp))"));

lay+=row2A('Localidad',inpB('loc','','text',35,'Change','parent.varios.env',"'Find',c('v_cobro','null,null,'+z(loc))"));
            templB(templ,'idx',lay,templ);break;
 case 'Listados':
            if (modo==1){
             s_('s_cMod',templ);s_('r_id',0);
             sql('Find Listados,-1,'+s('s_User'),1,'doc');
            }
            lay=row1L(b('Buscar por Titulo de Listado'));

lay+=row2A('Titulo',inp('descr','',30),bot('Buscar','parent.varios.env',"'Find',c('Listados',null,null,z(descr))","javas
cript:parent.varios.nw='Encontrados';"));
            templB(templ,'idx',lay,templ);break;
 case 'Normas': templB(templ,'idx','',templ);break;
 case 'Soporte':
            s_('s_cMod',templ);jsF('chkID');s_('r_id',0);
            sql('Find Soporte,'+s('s_User'),1,'doc');
            lay=row1C(s('s_tbl'));
            lay+=row1R(bot('Cumplida','parent.varios.env',"'List',parent.body.chkID()"));
            lay+=hr();
            lay+=row1L(b('Ingresar Reclamo'));
            lay+=row2A('Problema',a('a_problem'));
            lay+=row2A('Equipo',a('a_equipo'),'Descr.Equipo',inp('eqdescr','',45));
            lay+=row2A('Detalle',inp('obs','',80));
            lay+=hr();
            lay+=row1L(b('Instaladores'));
            lay+=row2A(lk('Acrobat Reader','../soft/acrobat.zip','','')
+spc(5)+lk('Timbuktu','../soft/timbuktu.zip','','')+spc(5)+lk('WinZip','../soft/winzip.exe','',''));
            templB(templ,'idx',lay+botsCG(templ),templ);break;
 case 'Contactos':
            if (modo==1){
             s_('s_cMod',templ);s_('r_id',0);
             sql('Find Contactos,-1,'+s('s_User'),1,'doc');
            }
            lay=row1L(b('Buscar por Nombre de Contacto'));

lay+=row2A('Nombre',inp('descr','',30),bot('Buscar','parent.varios.env',"'Find',c('Contactos',null,null,z(descr))","ja
vascript:parent.varios.nw='Encontrados';"));
            lay+=hr();
            lay+=row1C(s('s_tbl'));
            lay+=hr();
            lay+=row1L(b('Ingresar Contacto Personal Nuevo'));
            lay+=row2A('Nombre',inp('nombre','',60));
            lay+=row2A('Direccion',inp('direccion','',60));
            lay+=row2A('Telefonos',inp('telefonos','',60));
            lay+=row2A('E-Mail',inp('mail','',60));
            lay+=row2A('Obs',inp('obs','',60));
            templB(templ,'idx',lay+botsCG(templ),templ);break;
 case 'IBanking': templB(templ,'idx','',templ);break;
 //__________FORMS_____________________________________________________________________________
_____________________________
 case 'RacingAbm':
            if (modo==1) combo('s_popcli',0,'','CliFind');
            lay= row2A('Nro',r('r_#'));
            lay+=p2C('Cliente','popcli',i1R('Apellido','ap',20)+i1R('Nombre','nom',20)+inpB('r_cli',r('r_cli'),'hidden'));
            lay+=i2R('Dom','dom',35,i1R('CP','cp',5)+inpB('s_toImp',r('s_toImp'),'hidden'));
            lay+=i2R('Loc','loc',30,i1R('Pcia','pcia',20));
            lay+=i2R('Telefono','tel',20,p1R('Tipo Doc','doct')+i1R('Nro','docn',8));
            lay+=hr();
```

```
            lay+=i2F('Fecha','fh',i1R('Obs','obs',50));
            lay+=i2R('Importe','imp',15,p1R('Origen','frorigen')+i1R('','origendet',20)+((s('s_toImp')==1)?
inpB('r_pagado','0','hidden'):chkR('pagado')));
            templForm(s('s_cMod'),lay);break;
 case 'SOSAbm':
            lay= row2A('Nro',r('r_#'));
            lay+=p2R('Problema','problem',p1R('Equipo','equipo')+i1R('Descr.Equipo','eqdescr',45));
            lay+=i2R('Detalle','obs',80);
            lay+=hr();
            lay+=p2R('Prioridad','prior',p1R('Objeto','obj')+p1R('Impacto','impacto'));
            lay+=p2R('Sop.Asignado','soporte',p1R('Estado','st')+i1F('Finalizacion','ffinal'));
            lay+=hr();
            lay+=i2R('Sop.Externo','sopext',20,i1F('Fecha','fsopext')+i1R('Obs','seobs',40));
            lay+=hr();
            lay+=i2R('Deriv','deriv',20,i1F('Fecha','fderiv'));
            templForm(templ,lay);break;
 case 'App':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=row2A('Tipo',str2Pop('!null|!fija|fija!var|var','AppT'));
            lay+=i2R('Pars','pars',50);
            templForm(templ,lay);break;
 case 'Permisos':
            lay= row2A('Nro',r('r_#'));
            lay+=p2R('Depto','Depto');
            lay+=p2R('Oficina','Oficina');
            lay+=p2R('Tarea','App');
            templForm(templ,lay);break;
 case 'Ed':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Usr','usr',10);
            lay+=p2R('Form','form');
            lay+=i2R('ID','fid',5);
            lay+=i2R('Fecha y Hora','edit',20);
            templForm(templ,lay);break;
 case 'Form':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'St':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'Bco':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=i2R('Cod.Host','codHost',10);
            templForm(templ,lay);break;
 case 'Sucursal':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Codigo','cod',4);
            lay+=i2R('Nombre','descr',25);
            lay+=i2R('Domicilio','dom',20);
            lay+=i2R('CP','cp',5);
            lay+=i2R('Localidad','loc',20);
            templForm(templ,lay);break;
 case 'Contacto':
```

```
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Nombre','nombre',60);
            lay+=i2R('Direccion','direccion',60);
            lay+=i2R('Telefonos','telefonos',60);
            lay+=i2R('E-Mail','mail',60);
            lay+=i2R('obs','obs',60);
            templForm(templ,lay);break;
case 'Emp':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Apellido y Nombres','ap_nom',60);
            lay+=i2F('F.Nacimiento','fnac',p1R('Tipo Doc','doct')+i1R('Nro','docn',8));
            lay+=i2R('Domicilio','domp',60);
            lay+=i2R('Localidad','locp',30,i1R('CP','cpp',5));
            lay+=i2R('Telefonos','fonop',50);
            lay+=hr();
            lay+=i2R('Legajo','legajo',6,i1R('Interno','interno',5)+i1R('Puesto','puesto',30));
            lay+=p2R('Gerencia','gcia',p1R('Depto','depto'));
            lay+=p2R('Oficina','oficina',p1R('Sucursal','sucursal'));
            lay+=i2F('F.Ingreso','fingreso',i1F('f.Egreso','fegreso'));
            lay+=hr();
            lay+=i2R(b('Sistema')+spc(3)+'Login','login',8,'Pass '+inpB('r_pass',r('r_pass'),'password',8));
            templForm(templ,lay);break;
case 'DgiT':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=i2R('Cod.Host','codHost',10);
            templForm(templ,lay);break;
case 'DocT':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=i2R('Cod.Host','codHost',10);
            templForm(templ,lay);break;
case 'IBank':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=p2R('Hija de','IBankP');
            templForm(templ,lay);break;
case 'Pcia':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=i2R('Cod.Host','codHost',10);
            templForm(templ,lay);break;
case 'Listado':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Archivo','arch',12);
            lay+=i2R('Titulo','descr',50);
            lay+=p2R('Sucursal','sucursal');
            lay+=i2R('Obs','obs',50);
            lay+=p2R('Sistema','sistema');
            lay+=i2R('Programa','prog',15);
            templForm(templ,lay);break;
case 'Sistema':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=i2R('Origen','exist',10);
            templForm(templ,lay);break;
```

```
 case 'Norma':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Archivo','arch',30);
            lay+=i2R('Titulo','descr',80);
            lay+=p2R('Tipo','normat',chkR('live'));
            templForm(templ,lay);break;
 case 'NormaT':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'FRcli':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Apellido','ap',20);
            lay+=i2R('Nombre','nom',20);
            lay+=i2R('Dom','dom',35);
            lay+=i2R('CP','cp',5,i1R('Loc','loc',30));
            lay+=i2R('Pcia','pcia',20);
            lay+=i2R('Telefono','tel',20);
            lay+=p2R('Tipo Doc','doct',i1R('Nro','docn',8));
            templForm(templ,lay);break;
 case 'FRorigen':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            lay+=i2R('Cod','cod',10);
            templForm(templ,lay);break;
 case 'sosEquipo':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'sosImpacto':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'sosObjeto':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'sosPrioridad':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 case 'sosProblem':
            lay= row2A('Nro',r('r_#'));
            lay+=i2R('Ab','ab',10);
            lay+=i2R('Descr','descr',30);
            templForm(templ,lay);break;
 default: templB(templ,row1C('Página en Construcción: '+b(templ)));
 }
}//_____________________________________________
function abm(fp,tabla,fase){
 if (fase==1){
  s_('s_tabla',tabla);
sql(fp.replace('Pops',''),4,tabla,'Change','parent.varios.env',"'Abm',c('"+tabla+"',p(document.forms[0].elements[0]))
");
  templB(tabla,'abm');
```

```
 }
 if (fase==2){x=(tabla.indexOf(',')==-1)?tabla.length:tabla.indexOf(',');sql(fp,2,'f');layouts(tabla.substring(0,x),1)}
 if (fase==3){sql(fp);layouts(tabla/*s('s_cMod')*/,1)}
}//_____________________________________________________
function templB(tit,tipo,cont,cMod){
switch(tipo){
  case 'idx':          s_('s_cMod',cMod);
                       s_('s_newMod',1);
                       s_('s_cModBack',tipo);
                       s_('s_layBody',(u(cont)=='')?'':tblForm(cont));break;
  case 'form':         s_('s_cMod_',tipo);s_('s_layBody',tblForm(cont));break;
  case 'list':         s_('s_cMod_',tipo);s_('s_layBody',tag('center',cont));break;
  case 'abm':          s_('s_cMod_',tipo);s_('s_layBody',tblForm(row1C(lkForm('Nuevo','Abm',s('s_tabla')
+',0','800040'))+row2A(s('s_tabla'),s(s('s_tabla')))+row1C(lkForm('Volver','newMod','Admin','800040'))));break;
  case 'vacia': s_('s_layBody',spc());break;
  default: s_('s_layBody','Aplicacion en Desarrollo');break;
 }
 s_('s_titBody',img('../img/titulos/'+u(tit)+'.gif',tit));
}//_____________________________________________________
function templForm(templ,lay){return templB(templ,'form',lay+botsCG(templ))}
function tblForm(rows){return tag('center',tbl(trd(tbl(br()+rows+br(),'100',0,0,0,'')),'',4))}
%>
```

En esta pagina podemos ubicar el core de la intranet, esto es, que esta pagina contiene la rutina que recibe los pedidos del Browser (root(f,p,t,nw)), las acciones a seguir según cada llamado y la carga de las variables que conforman las páginas de la aplicación (s_Menu, s_titBody, s_layBody, etc.). Dichas rutinas son: root, admin, modIni, modLoad, modAct, modClose, modSave, modCS, find, list, layouts, layBody, abm, templB, templForm y tblForm; y todas tienen como objetivo el armado de los templates HTML necesarios para formatear el contenido dinamico de la pagina. Cada una de estas funciones esta descripta en el Apendice E.

Base.inc

```
<!-- #INCLUDE VIRTUAL="js.inc" -->
<%//_____________Variables_globales_____________________________________
var g_totRow,g_fmtList=new Array(),g_bFields=new Array(),g_itemsArbol=new Array(),g_padreArbol=new Arra
//Listados_____________________________________________________
g_fmtList['Listados']=';#,z;tit,a;fecha,a;hora,a;sistema,a;programa,a;';
g_fmtList['Normas']=';#,z;descr,l_blank;';
g_fmtList['Contactos']=';id,z;mail,m;';
g_fmtList['visa']=';#,z;periodo,a;doc,a;nombre,a;dom,a;loc,a;pcia,a;';
g_fmtList['Admin']=';Fecha_Hora,f;';
//Consultas____________________________________________________
g_fmtList['Racing1']=';#,h;t,z;sel,k;';
g_fmtList['Racing22']=';Importe,d0;cBarra,f;sel,k;t,z;';
g_fmtList['Racing11']=';t,z;';
g_fmtList['Racing2']=';t,z;';
g_fmtList['Racing3']=';t,z;';
g_fmtList['Racing4']=';Importe,d0;cBarra,f;sel,k;t,z;';
g_fmtList['SOS1']=';#,h;sel,k;';
g_fmtList['v_cobro']=';#,h;Alicuota,d0;';
//Campos Asociados a los botones de ABM's_____________
g_bFields['App']='z(r_ab),z(r_descr),p(appt,1),z(r_pars,1)';
```

```
g_bFields['Permisos']='p(a_depto),p(a_oficina),p(a_app)';
g_bFields['Form']='z(r_ab),z(r_descr)';
g_bFields['St']='z(r_ab),z(r_descr)';
g_bFields['Bco']='z(r_ab),z(r_descr),z(r_codHost,1)';
g_bFields['Sucursal']='z(r_cod),z(r_descr),z(r_dom,1),z(r_cp,1),z(r_loc)';
g_bFields['Emp']='n(r_legajo,1),z(r_ap_nom),z(r_interno,1),f(r_fnac,1),z(r_domp,1),z(r_cpp,1),z(r_locp,1),z(r_fo
g_bFields['Contacto']=s('s_User')+',z(r_nombre),z(r_direccion,1),z(r_telefonos,1),z(r_mail,1),z(r_obs,1)';
g_bFields['Contactos']=s('s_User')+',z(nombre),z(direccion,1),z(telefonos,1),z(mail,1),z(obs,1),b(comp)';
g_bFields['DgiT']='z(r_ab),z(r_descr),z(r_codHost,1)';
g_bFields['DocT']='z(r_ab),z(r_descr),z(r_codHost,1)';
g_bFields['IBank']='z(r_ab),z(r_descr),p(a_ibankp,1),b(r_live)';
g_bFields['Pcia']='z(r_ab),z(r_descr),z(r_codHost,1)';
g_bFields['Listado']='z(r_arch),z(r_descr),p(a_sucursal,1),z(r_obs,1),p(a_sistema,1),z(r_prog)';
g_bFields['Sistema']='z(r_ab),z(r_descr),z(r_exist,1)';
g_bFields['Norma']='z(r_arch),z(r_descr),p(a_normat),b(r_live),f(r_fvig,1),f(r_fact,1)';
g_bFields['NormaT']='z(r_ab),z(r_descr)';
g_bFields['FRcli']='z(r_ap,1),z(r_nom,1),z(r_dom,1),z(r_cp,1),z(r_loc,1),z(r_pcia,1),z(r_tel,1),p(a_doct,1),z(r_doc
g_bFields['FRorigen']='z(r_ab),z(r_descr),z(r_cod,1)';
g_bFields['sosEquipo']='z(r_ab),z(r_descr)';
g_bFields['sosImpacto']='z(r_ab),z(r_descr)';
g_bFields['sosObjeto']='z(r_ab),z(r_descr)';
g_bFields['sosPrioridad']='z(r_ab),z(r_descr)';
g_bFields['sosProblem']='z(r_ab),z(r_descr)';
//Campos Asociados a los botones de Forms_____________
g_bFields['Soporte']='p(a_problem),p(a_equipo),z(eqdescr),z(obs),'+qs(s('s_User'));
g_bFields['Racing']='z(r_cli),z(r_ap,1),z(r_nom,1),z(r_dom,1),z(r_cp,1),z(r_loc,1),z(r_pcia,1),z(r_tel,1),p(a_doct,1
g_bFields['SOS']=qs(s('s_User'))+',p(a_problem),p(a_equipo),z(r_eqdescr),z(r_obs),z(r_sol,1)'+((s('s_sosUsr')=
g_bFields['RRHH']='n(r_emp),f(r_fh,1),z(r_metaa,1),p(r_metaac,1),z(r_metab,1),p(r_metabc,1),z(r_metac,1),p(r
pe,1),p(r_vdp,1),p(r_vm,1),z(r_ej5,1),z(r_res,1),z(r_calif,1),z(r_pfuertes,1),z(r_ad,1),z(r_ar,1),z(r_cemp,1),b(r_fi
//_____________FORMATEOS HTML_____________________________________________
function tag(tag,x,pars){return ('<'+tag+((u(pars)!='')?' '+pars:'')+'>'+x+'</'+tag+'>').replace('  ',' ')}
//_____________TABLA HTML_____________________________________________
function tbl(rows,wi,bo,cspc,cspd,bgc){return tag('table',rows,((u(wi)!='')?'width='+wi+'%':'')+((u(bo)!='' || bo==0
function tl(label){return td(b(label),'nowrap align=center')}
function layIdx(label1,label2){return row1A(lkForm(b((u(label1)!='')?label1:'Ingresar Nueva'),'Load',c('0',s('s_U
function altas(label,tabla){return row1A(lkForm(b(label),'AbmPops',tabla))}
function tFh(){for(tFhi=0,pop=opt('null','');tFhi<tFh.arguments.length;tFhi++)pop+=opt(tFhi+1,tFh.arguments[tFh
function tr(cells,pars){return '\n'+tag('tr',cells,pars)}
function td(str,pars){return '\n\t'+tag('td',str,pars)}
function trd(cells,args){return tr(td(cells,args))}
function row(){for(rowi=0,cells='';rowi<row.arguments.length;rowi++)cells+=row.arguments[rowi];return tr(td(spc
function row1A(cont,align){align=(u(align)!='')?align:'left'; return trd(spc()+cont+spc(),'colspan=4 nowrap align='+
function row2A(cell1,cell2,oLabel,oObj){return row(td(u(cell1)+spc(),'valign=top align=right nowrap')+td(u(cell2)+
function row2Aw(cell1,cell2,oLabel,oObj){return row(td(u(cell1)+spc(),'valign=top align=right nowrap')+td(u(cell2)
function i2R(label,name,size,otros,ev,fun,pars){name='r_'+name;return row2A(label,inp(name,r(name),size,ev,fun
function i1R(label,name,size,ev,fun,pars){name='r_'+name;return spc()+label+spc()+inp(name,r(name),size,ev,fun,p
function i2F(label,name,otros){name='r_'+name;return row2A(label,inp(name,date2ISO(r(name)))+u(otros))}
function i1F(label,name){name='r_'+name;return label+spc()+inp(name,date2ISO(r(name)))}
function p2R(label,name,otros){return row2A(label,posPopA('a_'+name,r('r_'+name))+u(otros))}
```

```
function p1R(label,name,nname){id=(u(nname)!='')?r('r_'+nnname):r('r_'+name);pop=posPopA('a_'+name,id); return
function p2C(label,name,otros){return row2A(label,posPopS('s_'+name,'')+u(otros))}
function p1C(label,name){return label+spc()+posPopS('s_'+name,'')}
function a2R(label,name,rows,cols,otros){return row2A(label,u(otros)+tag('textarea',r(name),'name='+name+' ro
function hr(){return trd(tag('hr','','width=80%'),'colspan=4')}
//_____FORMS HTML_____
function inpB(name,valu,type,size,event,func,args,chk,src){return '\n\t\t<input name='+name+' type='+type+' siz
function inp(name,valu,size,event,func,args){return inpB(name,valu,'text',size,event,func,args)}
function chk(name,st,value,label){return (u(g_totRow)==1)?'':inpB(name,value,'checkbox',1,'','','',((st==1 || st==
function chkR(name,label){return chk('r_'+name,r('r_'+name),r('r_'+name),(((u(label)==''))?name:label))}
function select(x,name){return tag('select',x,'name='+name)}
function opt(ID,descr,sel){return '<option value='+ID+((sel==1)?' selected':'')+'>'+descr}
function bot(name,f,args,jsExtra,img){return '\n\t<input name="'+name+'" type=button value="'+name+'"'+evHdl
function botImg(name,f,args,path){return inpB(name,name,'image','','Click',f,args,'',path)}
function botsCG(form,extra){return row1A(bot('Cancelar','parent.varios.env',"'edClose","+s('r_id'))+spc(5)+((r('
//_____LINKS HTML_____
function lk(label,href,target,oc){return tag('a',label,'href='+href+((u(target!=''))?' target='+target:'')+u(oc))}
function lkHtm(label,pathImg,frame){return lk(((u(pathImg)=='')?label:img(pathImg+label+'off.gif',label)),qs('../
function lkForm(label,func,pars,color,target){return lk(((u(color)!='')?font(label,color):label),'body.asp?func='+((
function lkJs(label,func,pars,pathImg,frame){return lk(((u(pathImg)=='')?label:img(pathImg+label+'off.gif',label
function lkNW(label,func,pars){return lk(label,'nw.asp','_blank','parent.varios.nw='+qs(label)+';parent.varios.env(
function lkMail(str){return tag('a',font(str,'',-1),'href="mailto:'+str+'"')}
function lkAbm(label,table){return (lkForm(label,'AbmPops',table)+spc(2))}
function hlI(name,gif,nw,target,frame){return lk(img(gif,name),'body.asp?func=newMod&pars='+name+'&nw='+u(
//_____LISTAS HTML_____
function dl(str){return '\n'+tag('dl',str)}
function dt(str,coment){return '\n<dt>'+b(str)+u(coment)+'\n'}
function dd(str){return '\n<dd>'+str}
function li(str){return '\n<li>'+str}
function dtNW(str,func,pars){return dt(lkNW(font(str,'FFFFFF'),'doc',func,pars))}
function liNW(str,func,pars){return li(lkNW(str,'doc',func,pars))}
//_____DATOS HTML_____
//___Forma:
function font(str,color,size){return tag('font',str,((u(color)!='')?'color=#'+color:'')+((u(size)!='')?' size='+size:''
function strSize(str,size){return (size>15 && g_totRow!=1)?font(str,'',-1):str}
function colorT(n,toler,decs){if(u(toler)=='')toler=0;n=nFmt(n,decs);return(n<toler)?font(n,'FF0000'):((n==toler)
function b(str){return tag('b',str)}
function br(){return trd(spc(),'colspan=4')}
function img(src,name,otros){return "<img src='"+rep(src,' ','%20')+"' border=0 name='"+name+"'"+u(otros)+">"}
//___Contenido:
function nu(nustr,type){nustr2=(nustr+'').toLowerCase();return ((nustr2=='undefined' || nustr2=='null' || nustr2
function qs(str){return "'"+str+"'"}
function qsD(str){return '"'+str+'"'}
function spc(n){n=(u(n)=='')?1:n;for(spci=0,str='';spci<n;spci++,str+=' ');return str}
function rpad(str,car,cant){return (str+rpt(car,cant-str.length)).substr(0,cant)}
function lpad(str,car,cant){return (rpt(car,cant-str.length)+str).substr(0,cant)}
function rpt(car,cant){for(rpti=1,str='';rpti<=cant;rpti++,str+=car);return str}
function rep(str,x,y){for(repi=0,str2=str+'';repi<str2.length;repi++,str2=str2.replace(x,y));return str2}
function cantChar(str,car){for(cantChari=0,n=0;cantChari<str.length;cantChari++)if(str.charAt(cantChari)==car)
```

```
function c(){for(ci=0,str='';ci<c.arguments.length;ci++){if(u(c.arguments[ci])=='')return false;str+=c.arguments[c
function dFmt(n){return (((n+'').length>1)?'':'0')+n}
function date2ISO(val){dv=new Date(nu(val));if(val=='' || val==' ' || dFmt(dv.getYear())==NaN)return val;r
function nFmtM(n){return nFmt(n,2)}
function nFmt(n,decs){/*        formatea un nro
                                n: nro
                                decs: cantidad de decimales*/
 if (u(n)=='' || nu(n)=='') return '';
 nro=n+'',entC='';
 p=nro.indexOf('.');if (p==-1) p=nro.length;
 dec=rpad(nro.substr(p+1),'0',decs);
 ent=nro.substring(0,p);
 for (z=0;ent.length>0;z++,ent=ent.substr(0,ent.length-3))entC=','+ent.substr(ent.length-3)+entC;
 entC=entC.replace('-,','-');
 return entC.substr(1)+((decs>0)?'.':'')+dec;
}
function getDate(){
 dia=new Date();
 hoy= dia.getFullYear();
 hoy+=(dia.getMonth()>9)?(dia.getMonth()+1):'0'+(dia.getMonth()+1);
 hoy+=(dia.getDate()>9)?dia.getDate():'0'+dia.getDate();
 hoy+=' - '+((dia.getHours()>9)?dia.getHours():'0'+dia.getHours());
 hoy+=':'+((dia.getMinutes()>9)?dia.getMinutes():'0'+dia.getMinutes());
 hoy+=':'+((dia.getSeconds()>9)?dia.getSeconds():'0'+dia.getSeconds());
 return hoy
}
//_______VARIABLES_____________________________________
function rs2Var(scp){/* Pasa un rset a variables de session
                        scp=c => tantas variables como filas tenga el recordset, con el nombre = a la primer colum
                        scp=f => tantas variables como columnas tenga el recordset (caso de abm's) */
 if (scp=='c'){
  while (!rSt.EOF){
   switch(rSt(0).Value){
    case 'Menu':        Session('s_Menu')+= '\n\t\t'+lk(img('../img/'+rSt(0).Value+'/'+rSt(1).Value+'off.gif',rSt
    case 'Fijos': Session('s_Fijos')+='\n\t\t'+lk(img('../img/'+rSt(0).Value+'/'+rSt(1).Value+'off.gif',rSt(1).Value
    case 'Cumple':      Session('s_cumples')+=rSt(1).Value+'<br>';break;
    case 'hoy':         s_('s_hoy',date2ISO(rSt(1).Value));break;
    default:            s_('s_'+rSt(0),rSt(1).Value);break;
   }
   rSt.MoveNext();
  }
 }
 else for(rs2Vari=0;rs2Vari<rSt.Fields.Count;rs2Vari++) {s_('r_'+(rSt.Fields(rs2Vari).Name+'').toLowerCase(),rS
}//_____POPUPS_____________________________________________
function combo(name,id,qry,func){/* maneja el combo (input-popup)
                                name: nombre del combo
                                id: 0 (input), id de una opcion ya elegido (edicion de un formulario ya ingresado)
                                qry: '' (input), texto a buscar para armar el popup
                                Retorna:
```

```
                                        id=0 y qry='': input vacio
                                        id=0 y qry!='': popup con las opciones traidas de la consulta
                                        id!=0: popup con una opcion en blanco y la ya seleccionada (edicion de un fo
 if (u(func)=='') func=qry.substring(0,qry.indexOf(' '));
 func=qs(u(func));
 if(id==0 && u(qry)=='')s_(name,inp(name,'',6,'Change','comboCh',func+','+name)+b('?'));
 if(id==0 && u(qry)!='')sql(qry,4,name,'Change','comboCh',func+','+name);
 if(id!=0)s_(name,select(opt('null','')+opt(id,qry,1),name+' onChange="comboCh('+func+','+name+')"'));
}//_____________________________________________________________
function comboDraw(name,qry){
 if (s('s_cMod')=='Admin'){
  switch(name){
   case 's_poplis': return abm(qry,'Listado',1);
   case 's_popcli': return abm(qry,'Cliente',1);
  }
 }
 switch(name){
  case 's_popcli': fonC='CliFind';break;
  case 's_poplis': fonC='lisFind';break;
  case 's_popemp': fonC='empFind';break;
 }
 combo(name,0,qry,fonC);
 if (s('s_cMod')=='mp' && s('s_cMod_')=='abm'){return modAct('modSave '+s('r_id')+'',0)}
 layouts(s('s_cMod')+((s('s_cMod_')=='index')?'':'Abm'),0)
}//_____________________________________________________________
function rs2Pop(name,event,func,args){
 str=opt('null','',true);cantItems=0;
 while (!rSt.EOF){
  str+= opt(rSt(0),rSt(1));
  lastID=rSt(1).Value;cantItems++;
  rSt.MoveNext();
 }
 s_(name,select(str,name.toLowerCase()+evHdl(event,func,args)));
 if(cantItems==1) posPopS('s_'+name,lastID);
}//_____________________________________________________________
function rs2PopA(){/*  Pasa un rset a popup de aplicacion */
 str=opt('null','',true);name='';
 while (!rSt.EOF){
  if (name!==(rSt(0).Value+'').toLowerCase()){
   a_(name,select(str,'a_'+name));
   name=(rSt(0).Value+'').toLowerCase();
   str=opt('null','',true)
  }
  str+= opt(rSt(1),rSt(2));
  rSt.MoveNext();
 }
 a_(name,select(str,'a_'+name));
}//_____________________________________________________________
function ar2Pop(){/*    Convierte en popup un array
```

```
                        Pars: 0:name,1:event,2:func,3:args */
 for(ar2Popi=4,str='';ar2Popi<ar2Pop.arguments.length;ar2Popi++)str+=opt((ar2Popi==4)?'null':ar2Popi-4,ar2Pop.a
 return select(str,(ar2Pop.arguments[0]+'').toLowerCase()+evHdl(ar2Pop.arguments[1],ar2Pop.arguments[2],ar2P
}//_____________________________________________________________________
function str2Pop(str,name){/* Convierte un string en popup (el string viene en un campo del recordset)
                              str: el string a modificar
                              name: nombre del popup*/
 str=rep(str,'!','<option value=');
 str=rep(str,'|','>');
 return select(str,name.toLowerCase());
}//_____________________________________________________________________
function str2PopOpt(str,name){
 items=cantChar(str.toString(),'!');
 pos=(ifField('l'+name))?rSt('l'+name):r('r_'+name.substr(2));
 switch (items){
  case 0: return spc()+inpB(name,'','hidden');break;
  case 1: id=str.substring(str.indexOf('!')+1,str.indexOf('|'));descr=str.substring(str.indexOf('|')+1,str.length);
  default: return posPop(str2Pop(str,name),pos);
 }
}//_____________________________________________________________________
function posPopA(name,id){return posPop(a(name),id)}
function posPopS(name,id){return posPop(s(name),id)}
function posPop(obj,id){/*      Posiciona un popup
                              obj: nombre del popup
                              id: posicion a marcar
Retorna: un string conteniendo el popup posicionado (no modifica el objeto!)*/
 pos='value='+u(id);str=obj+'>';
 if (u(obj)=='' || u(id)=='') return obj;
 if (str.search(pos)){
  obj=obj.replace(' selected','');
  obj=obj.replace(pos+'>',pos+' selected'+'>');
 }
 return '\n\t\t'+obj+spc();
}//_________TABLAS_____________________________________________________
function rs2Tbl(scp){/*         pasa un rset a tabla html
                        Requiere: g_fmtList con algun valor (es el array que contiene los formatos para cada tipo
 t='';rStid='';nc=rSt.Fields.Count;fmt='';
 fmt=g_fmtList[s('s_fmtList')];
 fmtArr=new Array();
 argArr=new Array();
 var tot=ifField('t');
 for (rs2Tbli=0;rs2Tbli<nc;rs2Tbli++){
  n=rSt.Fields(rs2Tbli).Name;
  pos=fmt.indexOf(';'+n+',')+1;
  str=fmt.substring(pos+n.length+1,fmt.indexOf(';',pos));
  fmtArr[rs2Tbli]=(pos<=0)?rSt.Fields(rs2Tbli).type:str.substr(0,1);
  argArr[rs2Tbli]=(str.length==1)?'':str.substr(1);
  if (fmtArr[rs2Tbli]!='z'){
   x=(n.substr(0,1)=='.')?'.':((scp=='')?'':n);
```

```
  switch(x){
   case '.':                t+=tl(spc());break;
   default:                 t+=tl(n/*+tag('sub',fmtArr[rs2Tbli]+argArr[rs2Tbli])*/);
  }
 }
}
t=tr(t,'bgcolor=#42B993');
rnro=1;rs2Tbli=0;
while (!rSt.EOF){
 rows='';
 g_totRow=(tot && rSt('t')==1)?1:0;
 color=((++rnro%2)==0 && g_totRow==0)?'bgcolor=#FFFFCC':'bgcoor=#fffff';
 edit=(ifField('edit'))?rSt('edit'):0;
 for(rs2Tbli=0;rs2Tbli<nc;rs2Tbli++){
  if (rSt(rs2Tbli).name=='#') rStid=rSt(rs2Tbli);
  if (fmtArr[rs2Tbli]=='k') argArr[rs2Tbli]=rStid;
  rows+=(edit==1)?cellEdit(rSt(rs2Tbli).name,rSt(rs2Tbli).value,fmtArr[rs2Tbli],rStid):cell(rSt(rs2Tbli),fmtArr
 }
 t+= u(tr(rows,color));rSt.MoveNext();
}
s_('s_tbl',tbl(t,0,0,1,0));
}//_____________________________________________________________________
function cell(val,fm,args,id){/* formatea cada celda
                                 val: contenido de la celda
                                 fm: codigo de formateo
                                 args: argumentos a pasar a cada funcion de formateo*/
 name=val.Name;
 str=(nu(val)!=='')?val+'':spc();
 size=val.DefinedSize;
 args=(args+'');
 if(args.indexOf(' ')!=-1){f=args.substr(0,args.indexOf(' '));p=args.substr(args.indexOf(' ')+1)};
 if(g_totRow==1 && str=='' && fm!='z') return cellB(spc());
 arg1=u((args+'').substr(0,1));
 arg2=u((args+'').substr(1));
 switch (fm){
  case 'z': return '';                                                          //oculta
  case 'b': return cellB(b(str)+spc());                                         //bold
  case 'h': return cellB(lkForm(str,((u(args)=='')?'Load':args),c(id,s('s_User'))));  //hiperlink
  case 'l': return cellB(lk(str,'/'+s('s_cMod')+'/'+rep(id,' ','%20'),u(args),''))    //link comun          str.l
  case 'u': return cellB(lk(strSize(str),'/'+s('s_cMod')+'/990824/'+id,'body'))       //link comun
  case 'w': return cellB(lkNW(strSize(str,size),'doc',f,qs(p+id),str),'','nowrap');   //link en new Window
  case 'm': return cellB(lkMail(str),'','nowrap');                              //link de mail
  case 'k': return cellB(chk('c'+args,str,args));                               //Checkbox
  case 'i': return cellB(inp(name+args,nFmt(val,((u(args)=='')?2:args)),13));   //input
  case 'y': return cellB((nu(val)!='')?inp(name+args,nFmtM(val),11):str+inpB(name,'','hidden',1));//input opcional
  case 'p': return cellB(str2Pop(str,name));                                    //popup
  case 's': return cellB(str2PopOpt(str,name),'','nowrap');                     //popup opcional
  case 'd': return cellB((nu(val)!='')?nFmt(val,((u(args)=='')?2:args)):spc());//Decimales,paso la cant de
  case 'c': return cellB(colorT(((nu(val)!='')?val:spc()),arg1,arg2),'right');  //Coloreo segun umbral, pas
```

```
  case '#': return cellB(font(((nu(val)!='')?val:spc()),args));                    //Coloreo, paso el cod.de co
  case 'f': return cellB(strSize(str+spc(),size),((size<12)?'center':''),' nowrap'); //Forzar a String
  case 5:   return cellB(((nu(val)!='')?nFmt(val,6):spc()),'right');               //Float
  case 6:   return cellB(((nu(val)!='')?nFmtM(val,6):spc()),'right');              //Money
  case 131: return cellB(((nu(val)!='')?nFmtM(val,6):spc()),'right');              //Numeric
  case 11:  return cellB((g_totRow!=1)?(str=='true')?1:0:'');                      //Bit
  case 135: return cellB(date2ISO(str),'center');                                  //Fecha
  case 200: return cellB(strSize(str+spc(),size),((size<12)?'center':''),' nowrap'); //Strings
  default:  return cellB(str);
 }
}//_____________________________________
function cellEdit(name,valu,fmt,id){
 if (fmt=='z') return '';
 switch (name+''){
  case 'sel':      return cellB((nu(rSt('nro'))!='')?chk('sel'):inpB('sel',0,'hidden'));
  case 'Portf':  return cellB(posPopA('a_portf',valu));
  case 'brkSucL':return cellB(posPopS('s_brksuclpop',valu));
  case 'nom':    return cellB(inp('r_nom',valu,10));
  case 'comm':  return cellB((r('r_commexc')=='true')?inp('r_comm',valu,10):nFmt(valu,2)+inpB('r_comm','','hido
  case 'nro':    return cellB(nu(valu),'center');
  case '.':      return cellB(bot(valu,'parent.varios.env',"'mpDetEd',-1+','+"+r('r_id')))
  case '..':     return cellB(bot(valu,'parent.varios.env',"'mpDet',c('"+r('r_id')+"','"+id+"',p(a_portf),p(s_brksuc
  case 'saldo':  return cellB(inp('saldo',valu,10));break;
  default:       return cellB(nFmtM(valu),'right');
 }
}//_____________________________________
function cellB(str,alig,nwp){return td(((g_totRow==1)?b(str):str),((u(alig)=='')?'':'align='+alig)+u(nwp))}
//_____________RECORDSETS_____________________________________________
function rs2Lks(modo){ /* Pasa un rSet a links del frame submenu */
 i=0;
 while (!rSt.EOF){
  g_itemsArbol[i] = (rSt(3).Value!=1)
                        ?qsD((modo=='IBanking')
                                ?lkHtm(rSt(1).Value,'../img/smenu/','smenu')
                                :lkJs(rSt(1).Value,'Find',s('s_cMod')+','+rSt(0).Value+','+s('s_User'),'../img/sme
                        :qsD(rSt(1).Value);
  g_padreArbol[i] = rSt(3).Value;
  i++;
  rSt.MoveNext();
 }
}
//_____________SQL SERVER_____________________________________________
function sql(qry,target,scp,ev,fun,pars){/*     Ejecuta la consulta al sql y formatea los resultados
                                                qry: consulta
                                                target: tipo de formateo esperado 1: tabla,2: variables,3: popup, 4
                                                scp: 'a','s'; scope del objeto variable o popup
                                                ev,fun,pars: argumentos para llamar a evHdl (armar rutinas de java
 s_('s_msg');s_('r_n');
 var dia = new Date();
```

```
 if (!(qry+'').match("web") && s('s_User')==''){return layouts('home',0)}
 if ((qry+'').match("webLogin")){s_('s_Menu');s_('s_Fijos')};
 Session('s_trace')+=qry+'<br>';

 rSt=Server.CreateObject("ADODB.Recordset");
 con=Server.CreateObject("ADODB.Connection");
 con.CommandTimeout = 900;
 con.Open(a('a_str'));
 rSt.Open(qry,con);
 if (rSt.State!=1 || rSt.EOF){s_('s_msg',"Problemas en la Base de Datos.\\n Comunique esto a sistemas: "+rep(qr
 err=(ifField('errd'))?rSt('errd'):'';
 writeLog(qry,err);
 if (err==''){
  switch (u(target)){
   case 1: rs2Tbl(u(scp));break;
   case 2: rs2Var(u(scp));break;
   case 3: rs2PopA();break;
   case 4: rs2Pop(u(scp),ev,fun,pars);break;
   case 5: rs2Lks(u(scp));break;
  }
 }else{
  switch (rSt(0).Value){
   case 0:  return;
   case -1: s_('s_msg',qs(err));break;
   case -2: s_('s_msg','if (confirm('+qs(err)+')){parent.varios.env(\'warn\','+s('s_warning')+')}else parent.body.b
   default: s_('s_msg',rSt(0).Value);
  }
 }
 rSt.Close();
 con.Close();
 if (s('s_msg')!='')layouts('err',0);
}//_____________________________________________________________________
function readFile(file,indiv){
 str='';
 fs=Server.CreateObject("Scripting.FileSystemObject");
  if (!fs.FileExists(a('a_path')+file)){
   s_('s_msg',"'Archivo inexistente: '+"+qs(a('a_path')+file));
   layouts('err');
   return '';
  }
 arch=fs.OpenTextFile(a('a_path')+file,1,false);
 if (u(indiv)!=1)
  str=arch.ReadAll();
 else{
  arch.readline();
  str=arch.ReadAll();
 }
 arch.Close();
 return str;
```

```
}//_____________________________________________________________
function writeLog(accion,error){
 fs=Server.CreateObject("Scripting.FileSystemObject");
 arch=fs.OpenTextFile(a('a_path')+'log.txt',8);
 arch.WriteLine(s('s_User')+'\t'+getDate()+'\t'+accion+'\t'+error);
 arch.Close();
}//_____________________________________________________________
function parser(tarjeta,mes){
 archNew='';cantArchs=0;doc='';dom='';loc='';pcia='';nombre='';cta='';cuenta='';tarj='';ini=getDate();cabecera=
 fs=Server.CreateObject("Scripting.FileSystemObject");
 if (!fs.FileExists('c:\\'+tarjeta+mes+'.txt')){
        msg='Archivo inexistente: '+mes+'.txt\nEl nombre del archivo debe ser '+tarjeta+mes+'.txt y debe estar
        s_('s_msg',qs('No se encuentra el archivo: '+tarjeta+mes+'.txt. Lo estoy buscando en la unidad C: del ser
        layouts('err');
        return '';
 }
 if (!fs.FolderExists('d:\\inetpub\\wwwroot\\dev\\'+tarjeta+'\\'+mes)) fs.CreateFolder('d:\\inetpub\\wwwroo
 if (fs.FileExists('c:\\catRes.txt')) fs.DeleteFile('c:\\catRes.txt',true);
 arch=fs.OpenTextFile('c:\\'+tarjeta+mes+'.txt',1,false);

 while (!arch.AtEndOfStream){
        linea=arch.readline();
        if (linea=='1'){  //Grabo datos de encabezado e inicializo acumulador de archivo nuevo,se que el dato que
                if (doc!='************************' && doc.length>1 && '\t'+doc+'\t'+nombre+'\t'+dom+'\t'+lo
                        cabecera='\t'+doc+'\t'+nombre+'\t'+dom+'\t'+loc+'\t'+pcia+'\t'+mes+'\t'+cuenta+'\t'+t
                        cantArchs++;
                        narch='_'+cantArchs+'.txt';
                        grabarArch(narch,archNew,cabecera,tarjeta,mes);
                } else {                                        //solo agrego al final del archivo
                        fs2=Server.CreateObject("Scripting.FileSystemObject");
                        arch2=fs2.OpenTextFile(a('a_path')+'\\'+tarjeta+'\\'+mes+'\\_'+cantArchs+'.txt',8);
                        arch2.Write('\n'+archNew);
                        arch2.Close();
                }
                if (tarjeta=='visa'){
                        archNew=arch.readline()+'\n'; //sucursal+cuit visa
                        cta=arch.readline()+'\n';               //codigos, cant.paginas
                        archNew+=cta+'\n';
                        cuenta=cta.substring(82,91);
                        tarj=(cta.substring(93,99).match('033'))?'GOLD':'otras';
                        archNew+=arch.readline()+'\n';          //linea en blanco
                        archNew+=arch.readline()+'\n';          //nota
                        doc=arch.readline();                //documento (tipo + nro)
                        archNew+=doc+'\n';
                        doc=doc.substring(89,doc.length);
                        archNew+=arch.readline()+'\n';          //linea en blanco
                        nombre=arch.readline();                 //nombre
                        archNew+=nombre+'\n';
                        nombre=nombre.substring(10,nombre.length)
```

```
                        dom=arch.readline();            //domicilio
                        archNew+=dom+'\n';
                        dom=dom.substring(10,dom.length);
                        loc=arch.readline();            //localidad
                        archNew+=loc+'\n';
                        loc=loc.substring(10,loc.indexOf('CIERRE'));
                        pcia=arch.readline();           //pcia
                        archNew+=pcia+'\n';
                        pcia=pcia.substring(10,pcia.indexOf('SUC'))
                } else {
                        cta=arch.readline()+'\n';       //nro tarjeta y tipo
                        archNew+=cta+'\n';
                        cuenta=cta.substring(101,122);
                        tipo=cta.substring(91,93);
                        tarj=(tipo=='MR')?'REGIONAL':(tipo=='MO')?'ORO':'INTERNACIONAL';
                        archNew+=arch.readline()+'\n';      //linea en blanco
                        archNew+=arch.readline()+'\n';      //otra info
                        archNew+=arch.readline()+'\n';      //otra info
                        archNew+=arch.readline()+'\n';      //otra info
                        nombre=arch.readline();             //nombre
                        archNew+=nombre+'\n';
                        nombre=nombre.substring(22,nombre.length)
                        archNew+=arch.readline()+'\n';      //otra info
                        archNew+=arch.readline()+'\n';      //otra info
                        dom=arch.readline();            //domicilio
                        archNew+=dom+'\n';
                        dom=dom.substring(22,dom.length);
                        archNew+=arch.readline()+'\n';      //otra info
                        loc=arch.readline();            //localidad
                        archNew+=loc+'\n'
                        loc=loc.substring(22,loc.indexOf('CIERRE'));
                        doc='xxxx';
                        pcia='x';
                }
        } else archNew+=linea+'\n';
 }
 cantArchs++;
 grabarArch('_'+cantArchs+'.txt',archNew,'\t'+doc+'\t'+nombre+'\t'+dom+'\t'+loc+'\t'+pcia+'\t'+mes+'\t'+cu
 arch.Close();
 fin=getDate();
 sql('zVuelcoVisa '+mes,2);
 templB(tarjeta,'idx','Inicio del Proceso: '+ini+'<br>Fin del Proceso: '+fin);
}//_____________________________________________________________________________
function grabarArch(narch,contenido,cabecera,tarjeta,mes){
 fs2=Server.CreateObject("Scripting.FileSystemObject");
 arch2=fs2.CreateTextFile(a('a_path')+'\\'+tarjeta+'\\'+mes+'\\'+narch,true);
 arch2.Write(contenido);
 arch2.Close();
```

```
 fs2=Server.CreateObject("Scripting.FileSystemObject");
 if (fs2.FileExists('c:\\catRes.txt'))
  arch2=fs2.OpenTextFile('c:\\catRes.txt',8)
 else
  arch2=fs2.CreateTextFile('c:\\catRes.txt');
 arch2.Writeline(narch+cabecera);
 arch2.Close();
}//_____
function r(name){svar=nu(s(name))+'';return (svar.substr(0)==' ')?svar.substr(1):svar.substr(0)}
function ifField(f){for(ifFieldi=0;ifFieldi<rSt.Fields.Count;ifFieldi++){if(rSt.Fields(ifFieldi).Name==f)return true
//____________ADMIN APLICACION,SESION_____
function clear(){/* limpia las variables para cada pagina */
s_('s_script');
s_('s_dbg');
s_('s_smenu');
s_('s_itemsArbol');
s_('s_padreArbol');
s_('s_newMod');
s_('s_listAct');
s_('s_titDoc');
s_('s_layDoc');
s_('s_tbl');
}//_____
function iniSession(){/* Inicializa la aplicacion y las sesiones */
 s_('s_ss',false);
 s_('s_dbg');
 s_('s_script');
 s_('s_User');
 s_('s_UserD');
 s_('s_Menu');
 s_('s_Fijos');
 s_('s_listAct')
 s_('s_cumples');
 s_('s_msg');
 s_('s_layHlp');
 s_('s_frcapfh');
}//_____
function iniApp(){
 if (!Application('a_ini')){
  var fsObj,archObj,arch;
  a_('str','');
  a_('db','');
  iniSession();
  arch=(Request.ServerVariables("LOCAL_ADDR")=='172.22.25.216')?'d:\mbkloginT':'d:\mbkloginP';
  fsObj=Server.CreateObject("Scripting.FileSystemObject");
  archObj=fsObj.OpenTextFile(arch,1,false);
  a_('str',archObj.ReadLine());
  a_('path',archObj.ReadLine());
  a_('db',archObj.ReadLine());
```

```
  archObj.Close();
  sql('webini',3,'a','');
  Application('a_ini') = 'true';
 } else if(s('s_ss')!='true')iniSession();
}
//_____________App,Session_____________________________________
function nullif(str){return ((u(str)=='')?qs('null'):str)}
function u(ustr){if (ustr=='undefined') return '';return ((ustr+'')!='undefined' && ustr!='null' && ustr!=' ')
function s(name){return Session.Contents(name)}
function s_(name,value){Session.Contents(name)=u(value)}
function a(name){return Application.Contents(name)}
function a_(name,value){Application.Contents('a_'+name)=u(value)}
%>
```

Esta pagina contiene las rutinas JScript suplementarias para cumplir con los requerimientos de la anterior; es decir, la rutina de llamada a la base de datos, rutinas de inicio de aplicación y sesion, de formateo de datos del recordset que devuelve el SQL a objetos HTML, etc. Lo que debemos tener presente es que todas estas rutinas devuelven siempre como resultado un string, conteniendo tags HTML mas el contenido de informacion dentro de dichos tags.

Podemos agrupar las funciones de esta pagina según el siguiente esquema:

- √ Declaracion de variables y arrays globales de la aplicación, para el formateo de las columnas de listados y de los items a enviar en cada boton de formulario.
- √ Tag: generica, cualquier tag se puede generar a partir de esta rutina.
- √ Tags de formateo de tablas (tbl(),tl(), layIdx(), tFh(), tr(), td(), trd(), row(), etc.), donde ademas de la generacion de los tags convencionales de formateo de tablas existen rutinas que convinan estos con su contenido, es decir, que rellenan las celdas con determinados valores.
- √ Tags de armado de objetos del Form: (inp(),chk(), select(), bot(), etc.), que forman el HTML necesario para generar alguno de estos objetos.
- √ Tags para el armado de links, generan el HTML necesario para armar links a otras paginas, ya sean fijos o dinamicos, o dependan de acciones en Javascript (lk, lkHtm, lkForm, lkJs, lkNW, lkMail, lkAbm,etc.).
- √ Tags para la generacion de listas HTML (dl(), dt(), li(), etc.).
- √ Tags para el mostrado de texto (font(), strSize(), color(), b(), br(), img(), etc.)
- √ Tags para formatear el contenido de los datos (nu() para reemplazar campos nulos por blancos, qs() para agregar comillas a un dato, spc() para agregar un espacio HTML, date2ISO() para convertir los datos de tipo fecha al formato ISO, nFmt() para formatear campos numericos, etc.)

√ Rutinas para pasar los datos de un recordset a objetos del lado de las ASP's (en bloque):

- rs2Var: convierte los datos de un recordset en variables de sesion o aplicación.
- Combo: genera un objeto en realidad inexistente en HTML, que depende de la combinacion de los objetos <INPUT> y <SELECT>. Es decir, a nivel de interfaz, un objeto de la pagina es incialmente un <INPUT> que permite el ingreso de texto, que al completarse envia una consulta a la base de datos, esta devuelve los registros cuya informacion coincide con la ingresada y arma un objeto <SELECT>, desde donde el usuario selecciona el item de su interes.
- rs2Pop: convierte los datos del recordset en un objeto <SELECT>.
- rs2Tbl: convierte la grilla de datos recibidos en una tabla HTML, formateando los datos según el modo indicado en la variable global correspondiente.
- rs2Lks: formatea el recordset en un conjunto de links, que pueden organizarse en forma de arbol o ser variables sueltas, según los requerimientos de la aplicación.

√ Rutina para conectarse a la base de datos: sql(). Toma como argumentos el comando que se desea ejecutar (nombre del stored procedure mas sus correspondientes parametros), y el tipo de objeto HTML al que se desean formatear los datos devueltos por la base.

√ Rutinas para el manejo de archivos en el file system: son las que permiten crear, abrir, leer y escribir sobre archivos a nivel de sistema operativo (readfile, writelog, grabarArch, etc.).

√ Rutinas de inicializacion de aplicación y de sesion: (iniApp, iniSession, clear). Por medio de estas rutinas inicializo las variables que se mantendran según su scope. Estas rutinas reemplazan la existencia del archivo global.asa.

Objetos en SQL Server

Como vimos, hasta ahora solo se hablo de contenido de páginas, formateos, templates y demas. Ahora veremos como se graban los datos ingresados por el usuario en la base de datos y como se recuperan dichos datos, ya sea para ser modificados o para ser mostrados.

Para realizar una descripcion mas ordenada, utilizare la organización de los distintos objetos del SQL Server para describir como esta compuesta la aplicación desde este:

Reglas

Aquí defino reglas que luego actuaran sobre determinados tipos de datos. Las definidas hasta el momento son:

♦ Cp: es la regla que aplicare al tipo de dato cp, correspondiente a codigos postales (indicando el rango aceptado de valores para este tipo de dato):
create rule [cp] as @var between 1000 and 99999

- Fecha: aquí defino las caracteristicas del tipo de dato fecha, cuya particularidad sera setear la hora a cero (esto es necesario porque SQL Server toma como fecha el conjunto de datos que identifican la fecha y la hora actuales, y para busquedas seria necesario indicar tambien este segundo dato, cuando para la mayoria de las aplicaciones no es necesario):
  create rule [fecha] as datepart(hh,@fecha)=0 and datepart(mi,@fecha)=0

- Tasa: en esta regla limito el rango de valores posibles para este tipo de dato, que siempre debera estar entre 0 y 100:
  create rule [tasa] as @value between 0.00 and 100.00

- Dni: tambien utilizo esta regla para limitar el rango de valores admitidos para el tipo de dato docn:
  create rule [docn] as @val between 1 and 99999999

Como recomendación, es conveniente crear reglas para todos aquellos datos que contengan valores restringidos a determinados rangos o con caracteristicas particulares, para que el motor de SQL nos asegure la integridad de los mismos antes de grabarlos en las tablas.

Tipos de datos definidos por el usuario

En este conjunto de objetos se encuentran los datos que defino para ser utilizados solo por esta base de datos.

La ventaja de utilizar datos propios radica, en principio, en la posibilidad de asociarles reglas que me permitan validar los datos antes de ser ingresados, asi, cada vez que intente grabar un dato que no cumpla con dicha regla, el motor del SQL me informara del error y no me permitira grabarlo. Pero ademas, al asignar un tipo de dato propio para cada entidad, me aseguro que a futuro cada join que quiera realizar tendra el mismo tipo de datos a cada lado de la igualdad, y no tendre que preocuparme por recordar que tipo de dato le asigne a determinado campo de la tabla.

Algunos ejemplos de tipos de datos definidos:

- EXEC sp_addtype N'usr', N'int', N'not null'
  Defino el tipo de dato 'usr' basado en el tipo int

- EXEC sp_addtype N'ab', N'varchar (10)', N'not null'
  Defino el tipo de dato 'ab' basado en el tipo varchar con 10 caracteres como maximo

- EXEC sp_addtype N'cli', N'int', N'not null'
  Defino el tipo de dato 'cli' basado en el tipo int

- EXEC sp_addtype N'docn', N'int', N'not null'
  EXEC sp_bindrule N'[dbo].[docn]', N'[docn]'
  Defino el tipo de dato 'docn' basado en el tipo int y le asocio la regla 'docn'

- EXEC sp_addtype N'fecha', N'smalldatetime', N'not null'

EXEC sp_bindrule N'[dbo].[fecha]', N'[fecha]'

Defino el tipo de dato 'fecha' basado en el tipo smalldatetime y le asocio la regla 'fecha'

Tablas

Las tablas son los objetos donde definitivamente se almacencan los datos. En cada tabla se almacenara el total de elementos correspondientes a una determinada entidad (por ejemplo la tabla 'SOS' contendra todos los registros de solicitudes de soporte), caracterizados por sus atributos.

Una manera de generar tablas es escribir el siguiente codigo:

```
CREATE TABLE [dbo].[SOS] (
        [id] [sos] IDENTITY (1, 1) NOT NULL ,
        [fh] [smalldatetime] NOT NULL ,
        [usr] [emp] NOT NULL ,
        [problem] [smallint] NOT NULL ,
        [equipo] [smallint] NOT NULL ,
        [eqDescr] [descr] NULL ,
        [Obs] [varchar] (255) NULL ,
        [prioridad] [smallint] NULL ,
        [objeto] [smallint] NULL ,
        [impacto] [smallint] NULL ,
        [sopAsig] [emp] NULL ,
        [sopExt] [descr] NULL ,
        [fSopExt] [fecha] NULL ,
        [seObs] [varchar] (255) NULL ,
        [Deriv] [descr] NULL ,
        [fDeriv] [fecha] NULL ,
        [fFinal] [smalldatetime] NULL ,
        [St] [st] NOT NULL
) ON [PRIMARY]
GO

EXEC sp_bindrule N'[dbo].[fecha]', N'[SOS].[fDeriv]'
GO

EXEC sp_bindrule N'[dbo].[fecha]', N'[SOS].[fSopExt]'
GO

ALTER TABLE [dbo].[SOS] ADD
        CONSTRAINT [FK_SOS__Emp] FOREIGN KEY
        ([usr]) REFERENCES [dbo].[_Emp] ([id]          ),
        CONSTRAINT [FK_SOS__SOSequipo] FOREIGN KEY
        ([equipo]) REFERENCES [dbo].[_SOSequipo] ([id]),
        CONSTRAINT [FK_SOS__SOSimpacto] FOREIGN KEY
        ([impacto]) REFERENCES [dbo].[_SOSimpacto] ([id]),
        CONSTRAINT [FK_SOS__SOSobjeto] FOREIGN KEY
```

```
        ([objeto]) REFERENCES [dbo].[_SOSobjeto] ([id]),
        CONSTRAINT [FK_SOS__SOSprioridad] FOREIGN KEY
        ([prioridad]) REFERENCES [dbo].[_SOSprioridad] ([id]),
        CONSTRAINT [FK_SOS__SOSproblem] FOREIGN KEY
        ([problem]) REFERENCES [dbo].[_SOSproblem] ([id]),
        CONSTRAINT [FK_SOS__St] FOREIGN KEY
        ([St]) REFERENCES [dbo].[_St] ([id])
GO
```

Aquí, ademas de indicar nombre, tipo y longitud de los campos, estamos indicando que ciertas columnas deben contener valores asociados a otras tablas dentro de la base de datos. Con esto nos aseguramos que los datos contenidos en dichas tablas sean correctos, ya que si no existen en las tablas definidas, el motor del SQL generara un mensaje de error y no permitira que se almacenen dichos datos.

Respecto a los nombres de cada tabla, es recomendable que las mismas se identifiquen con palabras que definan la entidad que van a almacenar, y en lo posible adoptar alguna norma de prefijos (según importancia de la entidad, grupos de entidades asociadas, etc.) que permita un ordenamiento util al utilizar el SQL Enterprise Manager (por ejemplo, en nuestra aplicación, las tablas base del sistema comienzan con un carácter '_', y las propias de cada modulo con un conjunto de iniciales que identifiquen a cada uno de ellos.

Vistas

Son consultas sobre las tablas, que se almacenan precompiladas, y se utilizan, generalmente, para tener prearmadas consultas que implican establecer relaciones entre mas de una tabla, o para dar un nuevo formato a los datos, y son frecuentemente realizadas. De este modo se evita tener que repensar el query cada vez.

Un ejemplo de vista podria ser:

```
CREATE view vUsrApp as
Select e.id usr,ap_nom,a.id,a.ab,a.descr,tipo,isnull(href,'') href,target
 from _emp e,_depto d,_Oficina o,_app a,_permisos r
 where e.depto = d.id and e.oficina = o.id and r.depto = e.depto and r.oficina = e.oficina and
r.app=a.id
```

En esta vista estamos relacionando 5 tablas, obteniendo como resultado las aplicaciones a las que puede acceder cada usuario en base al departamento y sector al que pertenezca.

Stored Procedures

Los Stored Procedures (SP) son conjuntos de sentencias Transact-SQL almacenadas en bloques de codigo dentro del SQL Server. En este sistema, todos los datos son recibidos o enviados a través de SP por varias razones:

- Por seguridad: para asegurarnos de que solo se envien a la base de datos llamadas a estos, asi ataques del tipo 'enviar un select a la base dentro de un campo de texto' son detectados desde las ASP's, y nos aseguramos que esos datos no lleguen nunca a la base de datos.
- Para validar los datos a ingresar: ya que si bien SQL nos brinda muchas herramientas para asegurar la integridad y validez de los datos, los mensajes de error ante un dato incorrecto no se presentan al usuario en un modo entendible, para lo cual, es recomendable detectar a priori estos errores. Esto no significa que no valga la pena implementar los medios que el SQL nos brinda para dicha tarea, ya que estos pueden funcionar, por ejemplo, en casos que no hayamos contemplado al momento de escribir el SP, pero que nos fue simple recordar cuando establecimos el modelo de la base y definimos los objetos que la conformarian.
- Por robustez: la versión 3.0 del IIS contaba con muchas fallas en los objetos ADO, y en muchos casos errores imprevistos hacian que los vuelcos de datos a la base quedaran incompletos. Para asegurarnos que los datos hayan sido almacenados correctamente, por medio de los SP, nos aseguramos de no informar el OK al usuario hasta no haber recibido el mensaje de que todo llego y se grabo en forma correcta.
- Para confirmar transacciones de un modo simple y seguro: cuando por medio de una operación de carga del usuario, internamente debemos grabar datos en mas de una tabla, los SP's nos permiten asegurarnos de que todo esta bien antes de confirmar el grabado de los mismos (utilizando las sentencias begin tran – commit – rollback). Microsoft ofrece un producto llamado Microsoft Transaction Server para realizar esta validacion, pero es una herramienta que consume demasiados recursos y lentifica demasiado el proceso de grabacion, con lo cual decidimos hacer esta verificacion por nuestra cuenta y de un modo mucho mas simple, que cubre nuestras necesidades de validacion.

Para cada modulo dentro de la intranet tendremos basicamente los siguientes SP (donde Mod seria la sigla que identifica el modulo):

- ModLoad: permite la edicion de un registro de datos, ya sea nuevo o existente. En ambos casos necesito que me devuelva un recordset con una fila y tantas columnas como datos que conformen el formulario a mostrar al usuario (y a veces otros datos adicionales), ya que los vamos a utlizar para asignarles valores a los campos del formulario antes de armar el template donde el usuario los modificara.
- ModSave: recibira los datos que el usuario haya ingresado, realizara las validaciones correspondientes para cada uno de los datos y los grabara en su/s tabla/s correspondiente/s.
- ModFind: es el SP a través del cual se realizaran todas las consultas del modulo.
- ModExt: existe para los modulos que requieran procesamientos adicionales.

Como aprovechar este codigo para generar otros sistemas

La idea de este esquema de desarrollo siempre fue mejorar la generacion de nuevos sistemas a futuro (teniendo en vista ofertas de trabajo, en principio, y como desafio personal finalmente!). Asi, utilizando este esquema, podemos modificar los contenidos de logica de negocios e interfaz adaptandolo a nuevos requerimientos, sin necesidad de empezar desde cero cada aplicación. Inclusive, desde un modulo a modo de ejemplo se pueden desarrollar los demas.

A continuacion paso a enumerar los pasos a seguir para adaptar este sistema a uno modelo:

En SQL Server:

- En la tabla de aplicaciones (_App) ingresar los nombres y atributos de cada modulo que conforme la aplicación (generalmente estos se obtienen a partir de la diversidad de operaciones que maneje la empresa que va a implementar el sistema.

- En la tabla de usuarios (_Usr) ingresar a los futuros operadores del sistema, incluyendo sus datos personales (para futuros usos, ya que esta tabla bien podria imaginarse como la base de datos del personal de la empresa), sector de la empresa al que pertenezca (para la asignacion de permisos a los distintos modulos del sistema cuando se conecta), usuario y una password.

- En la tabla de sectores (_sector) ingresar los sectores de la empresa, para resolver los permisos de acceso a cada usuario a los modulos que conformen el sistema.

- Crear el SP de login al sistema (WebLogin), que valida al usuario, e inicializa un conjunto de variables que administraran la sesion de trabajo.

- Crear el SP de carga de datos inciales de la aplicación (WebIni, que genera popups de contenido fijo o poco variable).

En las páginas:

- Definir la funcion root(), con los casos de entrada al pedido de nuevas páginas (que se corresponden la mayoria de las veces con el nombre de los SP pero sin el nombre del modulo, por ejemplo 'Load', 'Save', 'Find', etc.).

- En la funcion layouts() definir cada template de las páginas a mostrar (basicamente: index del modulo, edicion del formulario principal, y editores de tablas base del sistema).

- Adaptar los templates de los frames al diseño definido por la empresa.

En principio, siguiendo estos pasos, es posible adaptar muy facilmente este sistema para futuros usos, por ejemplo las bases de datos no necesariamente se deben crear de cero, sino que se pueden realizar copias (vaciando las tablas de datos, por supuesto!), aprovechando SP's genericos (como por ejemplo Webini, WebLogin, un caso modelo de Load, Save y Find, etc.) y reglas y tipos de datos ya definidos. En el caso de las páginas, Base.inc no deberia sufrir modificaciones (considerando que lo que contiene es el conjunto de rutinas basicas de las ASP's), App.inc se deberia adaptar a los nuevos modulos (modificando los templates de cada pagina que se genere), y las demas solo sufririan modificaciones surgidas a partir de los cambios de diseño del nuevo site.

VII Conclusión

El punto mas importante que quiero destacar como conclusion es el hecho de que, de alguna manera, pude comprobar con esta experiencia que el paradigma de codigo reutilizable no es solo un sueño para un desarrollador. Quiero decir, que, teniendo como objetivo generar codigo parametrizable y facil de adaptar a nuevos requerimientos, logre generar al menos dos aplicaciones reales basadas en este esquema, y si bien siempre hay rutinas para optimizar, nuevos requerimientos que implementar, etc., cada nuevo toque en el codigo requiere menos esfuerzo.

Tambien me parecio importante poder tener un modo rapido de armado de prototipos, esto en gran parte se debe a la facilidad que brindan estas nuevas tecnologias (HTML, JavaScript y ASP's), que permiten el armado rapido de interfaces con un minimo de logica, elementos suficientes para dejar de mostrar ideas en papel al usuario, acercandolo mas al producto final.

Mas alla de la experiencia en el desarrollo, cabe destacar el esfuerzo extra que requiere el uso de herramientas tan nuevas:

- ✓ El primer problema es no contar con referentes de las mismas, puesto que poca gente se arriesga a utilizar herramientas nuevas, considerando que el mercado puede provocar tanto su furor como su desaparicion, y la mayoria de las veces cambiar el ambiente de desarrollo da la sensacion de ser una vuelta a empezar.

- ✓ Ademas, dichas herramientas no son simplemente nuevas versiones o similares a otras existentes, sino la directa implementacion de nuevas tecnologias, esto trae aparejado el problema de la escasa prueba de las mismas (debido a la competencia de mercado por los tiempos de salida de los productos), y de muchos aspectos que en teoria deberian funcionar de una determinada manera, pero en la practica no lo hacen, y entre otras cosas, la documentacion de las mismas no siempre se condice con su manera de funcionar.

- ✓ Otro factor a considerar es la seguridad y robustez que una aplicación generada por estas herramientas debe tener, por esto es que en este desarrollo muchas cuestiones relacionadas con estos temas fueron analizados e implementados de manera diferente a la recomendada por la documentacion de las herramientas, para asegurarnos de brindar estas caracteristicas mas alla de las fallas de las herramientas.

Finalmente, los lenguajes aquí utilizados son muy simples, contienen los elementos basicos necesarios para la implementacion de la mayoria de los requerimientos de un sistema convencional de un modo muy sencillo y es destacable la facilidad que brindan para interactuar con otras herramientas que cubran las acciones que estos no tienen contempladas.

En conclusion: considero que el conjunto de herramientas utilizado en este trabajo es muy aceptable, liviano en cuanto a requerimientos y costos, cubre todas las necesidades surgidas hasta el momento y permiten un desarrollo simple y eficiente. Y como aporte adicional en mi experiencia el esquema de aplicación logrado permite simplificar aun mas las etapas de desarrollo convencionales de sistemas.

VIII Bibliografía

- Eco Humberto, *Como hacer una Tesis*, 19° edición, junio 1996, Barcelona, Editorial Gedisa S.A.

- Stephen Walther, *Active Server Pages – the comprehensive solutions package Unlseashed*, 1° edición, 1998, EE.UU., Sams.net Publishing.

- Internet Information Server 4.0 help.

- Microsoft Corporation, *Implementing a Database on Microsoft SQL Server 7.0*, 1° edición, 1998, EE.UU., Microsoft.

- Willcam, *Willcam's Comprehensive HTML Cross Reference*, Internet.

IX Apendice A: HTML

Tags para definir la página

<!DOCTYPE>

Contiene la versión de HTML con que fue conformada la página.

Ej: <!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 3.2 Final//EN'>

<HTML></HTML>

Identifica un archivo como documento HTML. Este tag no contiene atributos.

Encabezado de la pagina:

<HEAD></HEAD>

Define el encabezado del documento HTML. Contiene informacion relacionada con el documento pero que no conforma la pagina que se mostrara a través del browser. No tiene atributos y los unicos tags que puede contener son los que se describen a continuacion:

<BASE>

Provee la URL completa para la página en cuestión.

Atributos:

- HREF: la URL completa del documento.
- TARGET: indica el nombre de la ventana donde quiero que se abra la página indicada. Los valores posibles son: _top, _Bottom.

Ej: <BASE HREF="http://www.viajes.com" TARGET="_top">

<LINK>

Indica links a otros archivos que contienen datos referidos a la pagina que los contiene, como por ejemplo archivos de estilos.

Atributos:

- REL: indica el tipo de archivo a ser linkeado, como por ejemplo: 'stylesheet', 'fontdef', etc.
- TYPE: contiene informacion extra sobre el tipo particular de archivo referido en el atributo anterior, por ejemplo, para 'stylesheet' el tipo es 'text/css'.
- SRC: es la ubicación fisica de la pagina en cuestion.

Ej:

```
<LINK
 REL="stylesheet"
 TYPE="text/css"
 SRC="styles/style1.htm">... </LINK>
```

<META>

Contiene informacion del documento que no conforma el contenido de la pagina HTML, como por ejemplo texto a ser leido por buscadores, parametros de otras aplicaciones que acceden a la pagina, etc.

Atributos:

- NAME: nombre de la documentacion contenida en este tag, los valores que admite son:
  - ✓ Description: utilizado por los buscadores para obtener una descripcion de la pagina,
  - ✓ Keywords: palabras claves que identifiquen la pagina (tambien para ser usadas por los buscadores),
  - ✓ Author: autor de la pagina.
- HTTP-EQUIV: contiene el nombre datos del encabezado de la pagina. Los valores aceptados son:
  - ✓ Expires: dia y hora en que el documento se considera desactualizado,
  - ✓ Refresh: permite indicar un tiempo para que la pagina sea recargada desde el browser sin intervencion del usuario.
  - ✓ Set-Cookie: configura una cookie para Netscape en modo permanente.
- CONTENT: contiene el valor relacionado con el HTTP-EQUIV o con el NAME

EJ: <META NAME="KEYWORDS" CONTENT="HTML, TAGS, REFERENCE, ATTRIBUTES">

<TITLE></TITLE>

Indica el titulo de la pagina, que se mostrara en la barra superior del browser. Cuando agregamos una pagina a un buscador de internet, este tomara el string contenido entre estos tags para agregarlo en las consultas que lo soliciten.

EJ: <TITLE>ESTA ES UNA PAGINA DE EJEMPLO!</TITLE>

<STYLE></STYLE>

Es otra manera de incorporar un stylesheet a la pagina.

Atributos:

- TYPE: indica alguno de los siguientes estilos:
  - ✓ Javascript: 'text/javascript'
  - ✓ Cascading: 'text/css'

```
Ej: <STYLE TYPE="text/css">
      P{font-size:18pt; margin-left:20pt;}
      H1 {color:blue;}
    </STYLE>
```

Ejemplo del encabezado de una pagina:

```
<HEAD>

<TITLE>ESTA ES UNA PAGINA DE EJEMPLO!</TITLE>
```

```
        <BASE HREF="http://www.viajes.com" TARGET="_top">
        <META NAME="keywords" CONTENT="HTML, tags, reference, attributes">
        <STYLE TYPE="text/css">
            P{font-size:18pt; margin-left:20pt;}
            H1 {color:blue;}
        </STYLE>
        <LINK REL="stylesheet" TYPE="text/css" SRC="styles/style1.htm">... </LINK>
</HEAD>
```

Contenido de la pagina:

<BODY></BODY>

Demarca el contenido de la página a ser mostrado por el browser.

Atributos:

- ALINK: indica el color para todos los hiperlinks activos.
- BACKGROUND: indica la URL de la imágen a ser utilizada como background de la página (esta imágen se repite hasta cubrir dicha página).
- BGCOLOR: indica el color default del fondo de la página (esto es tanto para el caso en que no se indique una imágen para el fondo o para cuando la URL indicada para dicha imágen no pudo ser resuelta por el browser).
- BGPROPERTIES: determina si la imágen del background acompaña el scroll de la página o permanece fija. Este atributo solo es reconocido por el Internet Explorer 3.0 en adelante.
- LEFTMARGIN: especifica el margen izquierdo del documento. Solo es reconocido por el Internet Explorer 3.0 en adelante.
- LINK: indica el color de todos los hiperlinks no seleccionados por el usuario.
- TEXT: indica el color del texto del documento.
- TOPMARGIN: especifica el margen superior del documento. Solo es reconocido por el Internet Explorer 3.0 en adelante.
- VLINK: indica el color de los hiperlinks ya visitados del documento.

Frames:

<FRAMESET>

Define un conjunto de frames que conforman una ventana del browser. Este conjunto puede contener mas de un frame y cada uno es definido a través del tag <FRAME>.

Este tag solo se puede usar en páginas de definicion de frames, cuyo unico contenido es la definicion de las caracteristicas de los frames (suele ser la pagina de inicio del site). Este documento no debera contener el tag <BODY>, ya que ningun contenido de esta pagina sera mostrado en el browser, y la ventana contendra las páginas definidas para cada frame.

Los frames se definen en relacion a filas y columnas dentro de la pagina, y para lograr la combinacion entre ambas se debe recurrir a framesets anidados.

Atributos:

- COLS: define las columnas que componen el frame set, y el valor es un string indicando los pixels o porcentaje que ocupara cada columna de ancho. Se recomienda dejar una columna con

el valor '*', para que cuando se quiera agrandar la ventana este frame sea el que se agranda, de lo contrario se extendera en forma proporcional.

- ROWS: al igual que el anterior, permite definir filas contenidas en un frameset. Los valores respetan el mismo formato que el atributo anterior.
- BORDER: indica el grosor en pixels de las lineas divisorias de frames. Si se indica 0 se logra no mostrar lineas divisorias de frames.
- BORDERCOLOR: color indicado para mostrar los bordes de los frames.
- FRAMEBORDER: con los valores 'YES' o 'NO' indica si se quiere mostrar o no un efecto 3D en los bordes de los frames.
- ONBLUR: contiene codigo javascript a ser ejecutado cuando el usuario sale de un frame especifico.
- ONFOCUS: codigo javascript que se ejecutarar cuando el usuario se posicione en el frame.
- ONLOAD: javascript a ejecutarse cuando se cargue la pagina en el browser.
- ONUNLOAD: scripting a ejecutar cuando el usuario salga de la pagina en cuestion.

<FRAME>

Define un frame particular dentro de un conjunto de frames.

Atributos:

- ALIGN: configura la alineación del frame. Los valores posibles son: top, bottom, left, center, right.
- BORDERCOLOR: indica el color del borde del frame.
- FRAMEBORDER: indica si el frame va a mostrar el borde o no. Los valores posibles son 0 y 1.
- MARGINHEIGHT: indica la altura del frame en pixels.
- SRC: URL del documento que conformará el frame.
- WIDTH: indica el ancho del frame en pixels.

<NOFRAME></NOFRAME>

Debe estar contenido dentro del tag <FRAMESET> y se utiliza para definir que se mostrara en el browser cuando este no soporte el uso de frames (versiones 2.0 y anteriores).

Ejemplo de pagina de definicion de frames:

```
<HTML><HEAD><TITLE>Ejemplo de Frames</TITLE></HEAD>
<FRAMESET COLS="20%,*">
    <NOFRAME>Si su browser no soporta frames Ud. estara viendo este mensaje!.</NOFRAME>
  <FRAME SRC="frametoc.htm" NAME="noname">
  <FRAMESET ROWS="30%,*">
   <FRAME SRC="frtoc1.htm" NAME="toptoc">
   <FRAME SRC="frstart.htm" NAME="outer">
  </FRAMESET>
</FRAMESET>
</HTML>
```

Tags que conforman el contenido de la página

Identificadores de Bloques de Texto:

<ADDRESS>…</ADDRESS>

Se utiliza para resaltar direcciones de mail, generalmente formatea la misma en estilo italico.

Ej: <ADDRESS>khernand@anice.net.ar</ADDRESS>

<BLOCKQUOTE>…</BLOCKQUOTE>

Agrega un indentado al bloque de texto que contiene. No tiene atributos.

Ej: <BLOCKQUOTE>este parrafo se deberia mostrar indentado. </BLOCKQUOTE>

<DIV>…</DIV>

Encierra un bloque de contenido de una pagina, y permite aplicar ciertas caracteristicas a dicho contenido.

Atributos:

- ALIGN: indica la alineacion horizontal del contenido del bloque. Admite los siguientes valores: LEFT, CENTER y RIGHT.
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ej:

```
<DIV ALIGN=RIGHT STYLE=REDSTYLE>
<H1>Alineando un bloque de texto a la izquierda</H1>
<P>Se puede utilizar este tag para alinear un bloque de texto hacia la izquierda de la pagina.</P>
<P>El contenido puede estar formado por cualquier elemento que componga una pagina, incluidas tablas, imágenes, listas, etc. Pero puede suceder que algunos elementos internos tengan definido su propio formato, y en tal caso se ignorara este tag.</P>
</DIV>
```

<H1>…</H1>
<H2>…</H2>
<H3>…</H3>
<H4>…</H4>
<H5>…</H5>
<H6>…</H6>

Muestran encabezados o titulos, dentro del texto que conforma la pagina. La numeracion 1 a 6 distinguen el tamaño de la letra, y todos los contenidos de estos tags se mostraran en negrita.

Atributos:

- ALIGN: indica la alineacion horizontal del contenido del bloque. Admite los siguientes valores: LEFT, CENTER y RIGHT.

- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ej:
<H1>Encabezado en nivel 1</H1>
<H2>Encabezado en nivel 2</H2>
<H3>Encabezado en nivel 3</H3>
<H4>Encabezado en nivel 4</H4>
<H5>Encabezado en nivel 5</H5>
<H6>Encabezado en nivel 6</H6>

<P>…</P>

Identifica un parrafo. Esto es, deja un espacio desde el elemento anterior y muestra el texto que le sigue en una nueva linea. No es obligatorio cerrar este tag, pero al hacerlo se asegura que tambien deje un espacio abajo del parrafo.

Atributos:
- ALIGN: indica la alineacion horizontal del contenido del bloque. Admite los siguientes valores: LEFT, CENTER y RIGHT.
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ej:
<P>Este texto esta encerrado en tags de parrafo, esto significa que deberia mostrarse como un bloque separado de texto, con espacios antes y despues de esta frase.</P>

<PRE>…</PRE>

Muestra el texto encerrado entre este tag respetando el formato asignado en el documento original, esto es, respeta los margenes, espacios y demas contenidos en el texto. Puede resultar util para incluir en el texto formateado en un editor de texto, de modo que respete los formateos asignados.

El uso de este tag no impide incluir otros en el texto que contenga, es decir, que se pueden asignar otras caracteristicas de formato al texto que se muestre.

Atributos:
- COLS: indica el maximo de caracteres a mostrar por linea, y se recomienda acompañarlo del atributo WRAP para permitir que las lineas mas extensas que la maxima definida se puedan mostrar en dos lineas.
- WRAP: permite mostrar una linea demasiado larga en varias lineas, respetando la cantidad maxima de caracteres que se pueden mostrar según el tamaño actual de la ventana del browser.
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.

- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ej:
<PRE>
To: Juan Perez
From: Jose Lopez
Subject: Reunion del equipo de planificacion
Date: jueves, 14 Agosto de 1999 22:00:05
9/20/99 8:00 a.m. salon 218
9/21/99 9:00 a.m. salon 218
9/22/99 2:00 p.m. salon 111
Los temas a desarrollar seran comunicados en los proximos dias..
</PRE>

<XMP>…</XMP>

Muestra el texto contenido entre ambos tags respetando el ancho fijo de la pagina, espacios, lineas, etc.; incluyendo los tags HTML incluidos en el texto.

Atributos:
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ej:
<P>La estructura basica de un documento HTML es:
<XMP>
<HTML>
<HEAD>aquí ubicamos la informacion del encabezado de la pagina. </HEAD>
<BODY>aquí incluimos el contenido de la pagina. </BODY>
</HTML>
</XMP>

Listas:

<DIR>…</DIR>

Muestra una lista de items simples o cortos, sin enumerar. Dentro de este tag, cada item se define por el tag <LI>.

<MENU>…</MENU>

Muestra una lista de items simples, no enumerados. Dentro de este tag, cada item se define por el tag <LI>.

<UL>…</UL>

Muestra una lista de items no enumerados. Cada item se indica con el tag <LI>.

Atributos:

- TYPE: define el estilo de numeracion a utiliar. Los valores posibles son: "CIRCLE", "DISC", "SQUARE".
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

<OL>…</OL>

Muestra una lista de items enumerados. Cada item se indica con el tag <LI>.

Atributos:

- START: indica a partir de que valor se comenzara a enumarar la lista.
- TYPE: define el estilo de numeracion a utiliar. Los valores posibles son: "A", "a", "I", "i", "1".
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

<LI>

Define un item o elemento de una lista, ya sea numerada o sin numerar.

Atributos:

- TYPE: indica el estilo usado para los números que ordenan la lista. Los valores posibles son: DISC, CIRCLE, SQUARE, A, a, I, i, 1).
- VALUE: indica el valor del número de orden a asignarle al item definido, en el caso de pertenecer a una lista ordenada.
- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ejemplo de lista sin numerar:

```
<P>Una estructura de directorios:</P>
<DIR>
 <LI>páginas
  <DIR>
  <LI>index.htm
  <LI>info.htm
  <LI>form.htm
  </DIR>
 <LI>imgs
  <DIR>
  <LI>mapa.gif
  <LI>zonas.gif
  </DIR>
```

```
<DIR>
```

Ejemplo de lista numerada:

```
<P>Meses del año:</P>
<OL START=1 TYPE='I'>
 <LI>Enero
 <LI>Febrero
 <LI>Marzo
 <LI>Abril
 <LI>Mayo
 <LI>Junio
 <LI>Julio
 <LI>Agosto
 <LI>Septiembre
 <LI>Octubre
 <LI>Noviembre
 <LI>Diciembre
</OL>
```

<DL>…</DL>

Define una lista de definiciones. Esta puede contener terminos (identificados con el tag <DT>) y definiciones (marcadas con el tag <DD>). Ambos se muestran en lineas diferentes, a menos que se utilice el atributo COMPACT para indicar que se quiere mostrar termino y definicion en una misma linea.

Atributos:

- COMPACT: muestra el termino y la definicion en una misma linea.

<DT>

Define un termino dentro de una lista.

<DD>

Define la descripcion de un termino.

Ejemplo de lista de terminos y definiciones:

```
<DL>
 <DT>Noroeste
  <DD>Incluye las provincias de Jujuy, Salta, Catamarca, La Rioja y Santiago del Estero.
 <DT>Mesopotamia
  <DD>Conformada por formosa, Chaco, Santa Fe, Misiones, Corrientes y Entre Rios.
 <DT>Centro y Cuyo
  <DD>Con todas las bellezas de las provincias de San Juan, Mendoza, San Luis y Cordoba.
 <DT>Region Bonaerense
  <DD>Aqui solo contamos con la provincia de Buenos Aires.
 <DT>Patagonia
  <DD>Formada por las provincias de Neuquen, La Pampa, Rio Negro, Chubut, Santa Cruz y
Usuahia.
```

</DL>

Formateado de Textos:

<B>…</B>

Muestra el texto contenido entre ambos tags en negrita.

Ej: <B>Este es un texto que se debería mostrar en negrita</B>

<BASEFONT>

Especifica las características básicas del formateo del texto.

Atributos:

- COLOR: indica el color default para el texto incluído en el documento. Solo funciona con Internet Explorer 3.0 en adelante.
- FACE: indica el estilo de texto default. Solo funciona con Internet Explorer 3.0 en adelante.
- SIZE: indica el tamaño default de la letra. Este valor puede estar comprendido entre 1 y 7 y de no ser especificado se asume que es 3.

Ej: <BASEFONT SIZE=2 COLOR=RED FACE=ARIAL>

<BIG>…</BIG>

Asigna al texto contenido el tamaño Large definido en el browser.

Ej: <BIG>Este texto debe verse en letra más grande!</BIG>

<BLINK>…</BLINK>

Hace que el texto contenido se muestre en la página en modo intermitente.

Ej: <BLINK>Texto en modo blink</BLINK>

**
**

Genera una nueva línea en el documento que sera interpretada por el browser.

Ej:

Este texto se muestra en una línea,
Y este se mostrara en la misma línea.

Este texto se muestra en una línea,
Y este en una nueva línea.

<CENTER>…</CENTER>

Centra el texto contenido.

Ej: <CENTER>Este texto se debe mostrar centrado.</CENTER>

<CITE>…</CITE>

Formatea el texto contenido como una citación o referencia, típicamente en itálico (similar al tag <ADDRESS>).

Ej: <CITE>Esto es una referencia</CITE>

<EM>…</EM>

Enfatiza el texto contenido formateándolo con estilo itálico (al igual que los tags <ADDRESS> y <CITE>).

<FONT>…</FONT>

Formatea un texto con un tamaño, estilo y color en particular.

Atributos:

- COLOR: indica el color del texto, este puede ser expresado como una constante o con su correspondiente valor RGB (ver Tabla de Colores).
- FACE: estilo de letra a aplicar.
- SIZE: tamaño de la letra. Puede ser un número entre 1 y 7 o aplicar el modo relativo (esto es, un signo '+' o '-' y un número a continuación indicando el número de tamaños a aumentar o disminuir.

<HR>

Dibuja una linea horizontal a traves de la pagina. Suele utilizarse para separar informacion de texto contenida en una pagina, correspondiente, por ejemplo, a distintas secciones.

Atributos:

- ALIGN: indica la alineacion horizontal de la linea. Admite los siguientes valores: LEFT, CENTER y RIGHT.
- NOSHADE: dibuja la linea sin efecto 3D.
- SIZE: indica el ancho de la linea en pixels, el minimo es 2.
- WIDTH: define el largo de la linea, que se puede expresar en pixels o en porcentaje.

Ej:

```
<P>Este texto aparecera por encima de la linea
<HR NOSHADE ALIGN="CENTER" WIDTH="50%" SIZE="8">
<P>Este texto quedara por debajo de la linea.
```

<I>…</I>

Muestra el texto encerrado en formato italico.

Ej: <I>Este texto se mostrara en Italico</I> y este en formato normal.

<KBD>…</KBD>

Muestra el texto en un font especial para indicar, por ejemplo, texto que debe ingresarse tal cual en algun campo. El browser utiliza para esto el font y tamaño indicados en el tipo de texto 'fixed-width font'

Ej:

```
<P>En el primer campo ingresar el nombre <KBD>demo</KBD> y en el siguiente la password
<KBD>demo</KBD>.
```

<MULTICOL>

Establece una zona de texto encolumnado en n columnas de igual tamaño.

Atributos:

- COLS: indica el numero de columnas en las que se mostrara el texto.
- GUTTER: indica, en pixels, el espacio a dejar libre entre las columnas.
- WIDTH: indica el ancho de las columnas en pixels. Si no se indica, se repartira el espacio disponible proporcionalmente entre las columnas.

Ej:

```
<MULTICOL COLS="3" WIDTH="520" >
<P>
Los tentáculos de la recesión no alcanzan a los que se van de vacaciones en invierno. La Secretaría de Turismo de la Nación calcula que la temporada que comienza convocará a un 20 por ciento más de viajeros que la anterior. O sea, a unos 6.250.0000 argentinos y extranjeros desde ahora y hasta fines de septiembre próximo.
<P>
¿Las razones? Una de ellas, y la que tienta a los turistas de mayor poder adquisitivo, se refiere a las abundantes nevadas en los principales centros de esquí del país, donde las reservas superan el 85 por ciento y en no pocos casos están agotadas para la primera semana de vacaciones.
<P>
Ushuaia, en Tierra del Fuego, es un claro ejemplo de ello. Años atrás, la de verano era la mejor temporada en la ciudad más austral del mundo. La ocupación alcanza hoy el 70 por ciento de su hotelería. Algo similar ocurre en Las Leñas, en Mendoza, donde se estima que este año facturarán unos 50 millones de pesos.
<P>
La promoción en el Brasil es otra de ellas. "Unas 490 empresas brasileñas compraron paquetes para la Argentina este año. Empezaron a cambiar un poco los Estados Unidos o Europa por la nieve en nuestro país", explicó el secretario de Turismo de la Nación, Francisco Mayorga.</P>
</MULTICOL>
```

<NOBR>…</NOBR>

Fuerza a que una linea extensa de texto se muestre en una sola linea de la pagina.

Ej:

```
<NOBR>
Este tag muestra todo el texto incluido en el en una sola linea. Hay que usarlo cuando sea realmente necesario, pues puede provocar lineas demasiado largas en una pagina.
</NOBR>
```

<PLAINTEXT>

Muestra el resto del documento sin interpretar los tags de HTML contenidos en el texto.

Ej:

```
<PLAINTEXT>
El tag <B>PLAINTEXT</B> impide la interpretacion de los tags HTML del texto que le sigue, hasta el final del documento.
```

<SMALL>…</SMALL>

Reduce el tamaño de fuente utilizado. Es similar al tag <FONT> con el atributo SIZE configurado en –1.

Ej:
<FONT SIZE="+1">
Este es un ejemplo donde vamos

<SMALL>REDUCIENDO Y <SMALL>REDUCIENDO EL FONT </SMALL></SMALL></SMALL>
</FONT> desde su tamaño normal.

<SPACER>

Permite incorporar espacios en el texto de una pagina.

Atributos:

- TYPE: indica si el espacio se debe insertar entre caracteres (HORIZONTAL), entre lineas (VERTICAL) o como si fuera una imagen invisible (BLOCK).
- SIZE: cuando no se utiliza el tipo BLOCK, indica la cantidad de espacios a dejar.
- WIDTH: indica el largo del bloque de espacios en pixels.
- HEIGHT: indica el alto del bloque de espacios en pixels.
- ALIGN: solo funciona cuando se inserta un bloque y se lo considera en relacion al texto que lo rodea.

Ej:
<P>Esta pala<SPACER TYPE=HORIZONTAL SIZE=20>bra tiene un pequeño espacio en medio!.
<SPACER TYPE=VERTICAL SIZE=40>
Esta linea se encuentra a 40 pixels de la anterior!.
<P>Y este texto tiene incorporado
<SPACER TYPE=BLOCK WIDTH=100 HEIGHT=80 ALIGN-MIDDLE>un bloque de espacios semejante a una imagen invisible.
</P>

<SPAN>…</SPAN>

Delimita con caracteristicas diferentes de formato un trozo de texto encerrado por los tags.

Atributos:

- CLASS: indica la clase de estilo a aplicar al bloque, y que ha sido definida en un tag <STYLE> en el encabezado del documento.
- ID: identifica un estilo elegido por su nombre.
- LANG: indica el lenguaje del texto que conforma la pagina.
- STYLE: identifica un estilo a aplicar en el bloque.

Ej:
<P>Este es un texto normal, pero si se quisiera destacar algo en particular, podriamos usar el tag <SPAN STYLE="color:blue; font-weight:bold; font-style:italic"> SPAN</SPAN> para destacarlo.

<STRIKE>…</STRIKE>

Dibuja una linea sobre el texto contenido. En general se utiliza esto para destacar que el texto ha sido modificado.

Ej: <P><STRIKE>En algun lugar deberia escribirse esto mismo pero correctamente!.</STRIKE>

<STRONG>…</STRONG>

Se utiliza para destacar una porcion de texto que el visitante no deberia dejar de leer antes de tomar cualquier decision. El efecto es similar al del tag <B>, ya que simplemente convierte a negrita la porcion de texto que encierra.

Ej: <P><STRONG>Cuidado!,</STRONG> este es un aviso importante!!!.

_…

Permite mostrar parte del texto en modo subindice, es decir, debajo de la linea normal de texto.

Ej:
<P>La notacion quimica del agua es <CODE> H ₂ 0 </CODE>.

[…]

Permite mostrar parte del texto en modo superindice, por encima de la linea normal de texto.

Ej: <P>La famosa formula de Einstein es <CODE> e=mc ₂</CODE>.

<TT>…</TT>

Muestra el texto en formato de font fijo, similar al tag <KBD>

<U>…</U>

Permite subrayar el texto.

Ej: <P><U>Todo este texto se muestra subrayado</U>. Este no!

<VAR>…</VAR>

Destaca una palabra o string en un texto mostrandolo en italico. Similar al tag <I>

<WBR>

Indica que en ese lugar se podria producir un salto de pagina, aun cuando se estuviera dentro de un tag <NOBR>.

Ej:
<NOBR>
Este tag solo se utiliza en bloques de texto encerrados entre tags NOBR para permitir el salto de linea aun dentro de estos.
<WBR>
No siempre se produce el salto de linea, pero la inclusion del tag indica que podria producirse.
</NOBR>

Marcadores:

<A>…</A> Hiperlink

El texto contenido entre estos tags es un hiperlink o anchor (marca dentro de una página).

Atributos:

- HREF: la dirección de la página que el browser deberá llamar si se clickea el hiperlink.
- NAME: nombre del link o anchor. Solo es necesario usarlo para identificar un anchor dentro de una página.
- TARGET: indica el nombre de la ventana donde quiero que se abra la página indicada. Los valores posibles son: _top, _Bottom.

Ej: <A HREF='html://www.server.edu/home.html'>Este texto es interpretado como un hiperlink</A>

<A HREF='html://www.server.edu/home.html#item3'>Esto es un anchor</A>

<LINK>

Define una relación entre el documento actual y otro indicado en sus atributos.

Atributos:

- HREF: indica la dirección http del documento al que apunta.
- TITLE: provee un título relacionado con la página que señala.

Imágenes:

<AREA>

Especifica un área para una imágen mapeada que puede ser ejecutada del lado del cliente (esto es, que la ejecute directamente el browser, sin enviar información al Web Server). Este tag esta contenido en el tag <MAP>, que es el que se utiliza habitualmente.

Atributos:

- ALT: texto alternativo a mostrar cuando el cliente configuró su browser para no mostrar imágenes.
- COORDS: define una región sobre la cual el usuario puede clikear en la imágen.
- HREF: la URL de la página a la que remite el mapeo de la imágen.
- NOHREF: indica una zona de la imagen sin mapeo asociado.
- SHAPE: indica el formato del área clickeable. Sus valores pueden ser: rect, circle o poly.
- TARGET: indica el nombre de la ventana donde quiero que se abra la página indicada.

<IMG>

Incorpora una imágen a un documento.

Atributos:

- ALIGN: Alínea la imágen en la página. Los valores posibles son: top, middile, bottom, left, center, right. Netscape tambien soporta: texttop, baseline y absmiddle.
- ALT: texto alternativo a mostrar si el browser no permite la bajada de las imágenes junto con la página.
- BORDER: tamaño del borde de la imágen.

- CONTROLS: determinan como aparecerán los controles VCR para administrar medios dinámicos, por ejemplo, video clips.
- DYNSRC: URL del código dinámico, como video clips o VRML.
- HEIGHT: alto de la imágen en pixels.
- HSPACE: espacios en blanco a cada lado de la imágen, en pixels.
- ISMAP: indica si la imágen es un mapa de hiperlinks manejado por el Web Server.
- LOOP: indica el número de veces que puede volver a ejecutarse un video clip. Los valores posibles son un número o el string INFINITE.
- LOWSRC: URL de una imágen de menor resolución a mostrar mientras termina de bajar la definitiva.
- SRC: URL de la imágen a insertar.
- START: determina cuando debe comenzar la aplicación indicada en DYNSRC. Los valores posibles son: FILEOPEN, MOUSEOVER.
- USEMAP: URL del nombre del mapa para un mapeo controlado por el browser.
- VSPACE: indica los espacios en blanco arriba y abajo de la imágen, en pixels.
- WIDTH: indica el ancho de la imágen en pixels.

<MAP>…</MAP>

Indica la inserción de una imágen .

Atributos:

- ALIGN: Alínea el encabezado por encima de la tabla. Los valores posibles son: top, bottom, left, right.

Tablas:

<TABLE>…</TABLE>

Define una tabla. Dentro de estos tags, se utilizaran el tag <TR> para indicar filas y el tag <TD> para indicar celdas dentro de las filas. Hay tambien otros tags que incorporan formateos adicionales a los elementos de la tabla, que se veran a continuacion en esta misma seccion.

Atributos:

- ALIGN: Alínea el encabezado por encima de la tabla. Los valores posibles son: LEFT, CENTER, RIGHT.
- WIDTH: define el ancho de la tabla. Si no se indca, ocupara el ancho de la pagina, o el necesario para mostrar el contenido de la misma.
- HEIGHT: define el alto de la tabla. Si no se indica, este se adaptara al contenido de la misma.
- BORDER: indica la cantidad de pixels necesarios para dibujar los bordes de la tabla. Si este valor es cero, no se mostraran los bordes de la tabla.
- CELLPADDING: indica el espacio entre el borde de la celda y el contenido de la misma, en pixels.
- CELLSPACING: indica el espacio entre las celdas de la tabla (en sentidos vertical y horizontal), en pixles. El valor default es 2.
- BGCOLOR: permite indicar un color de fondo para la tabla.
- HSPACE: indica el margen horizontal para la tabla, respecto de los demas contenidos de la pagina.

- VSPACE: delimita el margen vertical de la tabla.
- COLS: indica la cantidad de columnas que contiene la tabla, solo se utiliza cuando se desea que todas las columnas tengan el mismo ancho (es mas economico que indicar el ancho a cada una de las celdas).

<CAPTION>…</CAPTION>

Denota el encabezado de una tabla.

Atributos:

- ALIGN: Alínea el encabezado por encima de la tabla. Los valores posibles son: top, bottom, left, right.

<COLGROUP>

Indica propiedades para una o más columnas que conforman una tabla. Solo es reconocido por el Internet Explorer 3.0 en adelante.

Atributos:

- ALIGN: especifica la alineación horizontal del texto dentro de las columnas. Los valores posibles son: left, center, right.
- SPAN: indica el número de columnas afectadas con esta propiedad.
- VALIGN: especifica la alineación vertical del texto dentro de las columnas. Los valores posibles son: TOP, MIDDLE y BOTTOM.
- WIDTH: indica el ancho de las columnas en el grupo indicado.

<TR>…</TR>

Agrega una fila a una tabla.

Atributos:

- ALIGN: indica el alineamiento horizontal del contenido de la tabla. Los valores posibles son: LEFT, CENTER, RIGHT.
- VALIGN: indica el alineamiento horizontal del contenido de la tabla. Los valores posibles son: BASELINE, BOTTOM, MIDDLE y TOP.
- BGCOLOR: permite indicar un color de fondo de celdas diferente para cada fila de la tabla.

<TD>…</TD>

Incorpora una celda a una fila de la tabla.

Atributos:

- COLSPAN: indica el numero de columnas que puede ocupar la celda.
- ROWSPAN: indica el numero de filas que puede ocupar la celda.
- NOWRAP: impide que el contenido de la celda se muestre en mas de una linea.
- ALIGN: indica la alineacion horizontal del contenido de la celda. Los valores posibles son: LEFT, CENTER, RIGHT.
- VALIGN: especifica la alineación vertical del texto dentro de las columnas. Los valores posibles son: BASELINE, TOP, MIDDLE y BOTTOM.
- HEIGHT: setea el alto de la celda.
- WIDTH: indica el ancho de la celda.
- BGCOLOR: setea el color de fondo de la celda.

<TH>…</TH>

Tambien agrega una celda a una fila de una tabla, con la diferencia de que su contenido se muestra centrado y en negrita. Suele utilizarse para mostrar encabezados.

Atributos:

- COLSPAN: indica el numero de columnas que puede ocupar la celda.
- ROWSPAN: indica el numero de filas que puede ocupar la celda.
- NOWRAP: impide que el contenido de la celda se muestre en mas de una linea.
- ALIGN: indica la alineacion horizontal del contenido de la celda. Los valores posibles son: LEFT, CENTER, RIGHT.
- VALIGN: especifica la alineación vertical del texto dentro de las columnas. Los valores posibles son: BASELINE, TOP, MIDDLE y BOTTOM.
- HEIGHT: setea el alto de la celda.
- WIDTH: indica el ancho de la celda.
- BGCOLOR: setea el color de fondo de la celda.

Ejemplo de tabla:

```
<TABLE BORDER CELLPADDING="8" CELLSPACING="4" BGCOLOR=yellow>
 <TR>
  <TH> English </TH>
  <TH> Spanish </TH>
  <TH> German </TH>
 </TR>
 <TR>
  <TD> one </TD>
  <TD> uno </TD>
  <TD> ein </TD>
 </TR>
 <TR>

  <TD> TWO </TD>
  <TD> dos </TD>
  <TD> zwei </TD>
 </TR>
 <TR>
  <TD> three </TD>
  <TD> tres </TD>
  <TD> drei </TD>
 </TR>
 <CAPTION ALIGN="BOTTOM">
  <B>Table 1</B>: Tables are as easy as one, two, three
 </CAPTION>
</TABLE>
```

Formularios:

<FORM>…</FORM>

Indican el inicio y fin de un formulario dentro de una página. Entre ambos tags se deben definir los distintos objetos que conformen el formulario:

- Elementos de ingreso de datos por el usuario (como INPUT, SELECT, CHECKBOX o TEXTAREA),
- Elementos que accionan el formulario, es decir, que envian los datos al Web Server (Botones o INPUT de tipo SUBMIT, o con acciones definidas en los metodos ONCLICK o ONSUBMIT de JavaScript).

  Cuando los datos son enviados al Web Server, este los recibe en el siguiente formato: nombre1=valor1&nombre2=valor2, etc.

Atributos:

- ACTION: URL del programa o pagina que recibirá los datos para ser procesados.
- ENCTYPE: indica el modo en que serán codificados los datos contenidos en el formulario. Para que un botón de Upload de un archivo funcione correctamente, deben indicarse como valor de este atributo: multipart/form-data.
- METHOD: el método utilizado para enviar los contenidos del formulario. Los valores posibles son: post y get. El valor mejor soportado por la mayoría de los browsers es post, y su uso nos asegura que los datos siempre llegarán al server!.
- TARGET: indica el nombre de la ventana o frame donde espero que aparezcan los resultados del envío de datos realizado. Esto es util cuando quiero que la respuesta se muestre en una pagina o ventana diferente de la actual.
- NAME: asocia un nombre a un formulario. Esto permite detectar uno de una pagina que contenga varios, desde JavaScript.
- ONRESET: permite asociar codigo JavaScript que sera ejecutado cuando el usuario presione un boton de RESET en la pagina que contenga el formulario.
- ONSUBMIT: asocia codigo JavaScript a ejecutar cuando se presione un boton de SUBMIT incorporado dentro del formulario.

<INPUT>

Crea un campo o celda para el ingreso de datos en un formulario.

Atributos:

- ACCEPT: indica el tipo MIME que puede aceptar un botón de upload.
- ALIGN: alínea la celda respecto del texto que tenga alrededor. Los valores posibles son: top, middle, bottom, left, right.
- CHECKED: indica si un checkbox esta encendido o no cuando se pide inicialmente la página.
- MAXLENGTH: indica la cantidad máxima de caracteres cuando la celda es de tipo TEXT o PASSWORD.
- NAME: nombre de la celda.
- SIZE: indica el ancho de la celda.
- SRC: URL de una imágen. Este atributo se puede utilizar cuando la celda es de tipo SUBMIT o RESET, lo que indica que se trata de un botón, y la imágen reemplaza el dibujo tradicional de este objeto.
- TABINDEX: indica el número de orden de la celda dentro del formulario. Solo funciona con Internet Explorer 3.0 en adelante.
- TYPE: indica el tipo de objeto. Los valores posibles son:
  - ✓ TEXT: campo de texto (si no se indica este atributo se asume que es de tipo texto).

- ✓ PASSWORD: indica campo de texto, pero los caracteres ingresados se muestran reemplazados por '*'.
- ✓ CHECKBOX: casilleo clickeable con solo 2 estados posibles: on y off.
- ✓ RADIO: forma parte de un conjunto de checkbox relacionados.
- ✓ SUBMIT: botón que envía los datos del formulario al Web Server.
- ✓ RESET: botón que limpia los contenidos de todos los campos del formulario, dejando los valores default de los mismos.
- ✓ FILE: botón de upload de documentos.
- ✓ TEXTAREA: campo de texto que permite la edición de líneas múltiples.
- ✓ HIDDEN: campo que forma parte del formulario pero no es mostrado por el browser.
- ✓ IMAGE: imágen que reemplaza el botón de submit.

- VALUE: indica un valor inicial o default para los campos de tipo TEXT, HIDDEN o PASSWORD. También se pueden especificar valores para los campos de tipo CHECKBOX o RADIO.

<SELECT>…</SELECT>

Define una lista de opciones desde la cual el usuario puede seleccionar uno o mas items.

Atributos:

- NAME: indica el nombre del objeto, con el cual se identificara del lado de la aplicación que reciba los datos del formulario.
- MULTIPLE: permite al usuario seleccionar mas de un elemento.
- ONBLUR: contiene codigo JavaScript que se ejecutara cuando el usuario seleccione otro objeto del formulario.
- ONCHANGE: contiene codigo JavaScript que se ejecutara cuando el usuario modifique la seleccion.
- ONCLICK: contiene codigo JavaScript que se ejecutara cuando el usuario haga click sobre este objeto.
- ONFOCUS: contiene codigo JavaScript que se ejecutara cuando el usuario se enfoque sobre este objeto.
- SIZE: indica la cantidad de items visibles.

<OPTION>…</OPTION>

Define un item de un objeto SELECT.

Atributos:

- VALUE: contiene el valor asociado al item. Si no se asigna un valor, HTML le asignara la descripcion del item.
- SELECTED: marca el item como seleccionado por el usuario.

<TEXTAREA>…</TEXTAREA>

Define un campo de ingreso de texto de n lineas y n columnas. En caso de ser necesario, el browser puede dibujar barras de scrolling.

Atributos:

- COLS: indica el numero de caracteres a mostrar por fila.
- NAME: nombre del objeto.

- ONBLUR: codigo JavaScript que se ejecutara cuando el usuario seleccione otro objeto del formulario.
- ONCHANGE: contiene codigo JavaScript que se ejecutara cuando el usuario modifique la selección.
- ONFOCUS: contiene codigo JavaScript que se ejecutara cuando el usuario se enfoque sobre este objeto.
- ONSELECT: contiene codigo JavaScript que se ejecutara cuando el usuario seleccione parte del texto contenido en el objeto.
- ROWS: indica el numero de lineas a mostrar.
- WRAP: indica si permite o no mostrar el texto ingresado en mas de una linea cuando el usuario no ingreso caracteres de nueva linea.

Ejemplo de formulario:

```
<form name=body action=body.asp method=POST>
Inputs:

<INPUT NAME=INPUT TYPE=TEXT SIZE=6 VALUE='INPUT'><BR>
Popups:
<select name=popup>
    <option value=1 selected>item 1
    <option value=2>item 2
    <option value=3>item 3
</select><br>
Checkboxs:
<input name=checkbox type=checkbox><br>
Botones:
<input name='g' type=button value='submit'>
<input name='c' type=reset value='reset'>
</form>
```

Scripts, Applets & Plug-ins:

<APPLET>…</APPLET>

Incorpora un applet Java a la página.

Atributos:

- ALIGN: alinea el applet alrededor del texto de la página. Los valores posibles son: left, right,top,texttop,middle, absmiddle, baseline, bottom, alsbottom.
- ALT: texto a mostrar en browsers que no soportan el uso de applets Java.
- ARCHIVE: indica la URL del archivo (zipeado pero descomprimido) que contiene el applet Java.
- CODE: nombre del applet Java.
- CODEBASE: especifica el directorio donde esta ubicado el applet Java.
- HEIGHT: indica la altura de la ventana donde se abrirá el applet en pixels.
- HSPACE: especifica el espacio a izquierda y derecha del applet en pixels.
- MAYSCRIPT: determina cuando un applet Java puede utilizar Javascript (solo para Netscape).
- NAME: un nombre que identifica un applet Java.

- VSPACE: especifica el espacio arriba y abajo del applet en pixels.
- WIDTH: indica el ancho de la ventana donde se abrirá el applet en pixels.

<PARAM>

Permite enviar un parametro a un applet. Debe estar encerrado por los tags que definen el applet.

Atributos:

- NAME: indica el nombre de la variable o parametro definido en el applet.
- VALUE: contiene el valor a enviar.

Ejemplo de applet:

```
<P>Aquí estamos definiendo un applet con dos parametros: velocidad y mensaje.
<APPLET CODE="jumping.class" CODEBASE=jclasses
   WIDTH=240 HEIGHT=400
   ALIGN=ABSMIDDLE HSPACE=10 VSPACE=20>
  <PARAM NAME=msg
     VALUE="Estoy enviando un mensaje como parametro!. ">
  <PARAM NAME=vel VALUE="4">
</APPLET>
```

<BGSOUND>

Permite agregar un sonido como background de la página. Solo funciona con Internet Explorer 3.0 en adelante.

Atributos:

- LOOP: número de veces que se debe ejecutar el archivo de sonido. Los valores posibles son: un número o la palabra INFINITE, la cual indica que el archivo se ejecutara continuamente mientras la página sea mostrada.
- SRC: la URL del archivo de sonido.

Ej: <BGSOUND SRC=www.server.edu/sound.ram LOOP=5>

<EMBED>

Muestra el resultado de un archivo que es ejecutado por un plug-in en un rectangulo de la pagina a mostrar.

Atributos:

- ALIGN: alínea el objeto embebido. Los valores posibles son: left, right, top, bottom.
- BORDER: tamaño del borde del objeto en pixels.
- FRAMEBORDER: indica si el marco tiene borde o no. Si se le quiere asignar borde, basta con indicar el valor para BORDER, sino, a este atributo se le asigna el valor 'no'.
- HIDDEN: determina si el plug-in permanecerá oculto o no. Los valores posibles son true, false.
- HSPACE: indica los espacios vacios a la derecha e izquierda del objeto.
- HEIGHT: indica la altura del objeto dentro de la página, en pixels.
- NAME: nombre del objeto.
- PALETTE: configura la paleta de colores a utilizar por el objeto.
- PLUGINSPAGE: URL indicando la página que provee la información correspondiente al plug-in.
- SRC: URL del objeto.

- TYPE: indica el MIME TYPE del objeto.
- UNITS: especifica la unidad de medida a usar en los argumentos width y height.
- VSPACE: espacios en blanco superiores e inferiores del objeto
- WIDTH: indica el ancho ocupado por el objeto dentro de la página, en pixels.

<NOEMBED>…</NOEMBED>

Inserta un texto a mostrar cuando el browser no puede mostrar contenidos desde plug-ins.

Ejemplo de plug-in:

```
NOEMBED>En este espacio se mostraria el resultado de un plug-in si el browser lo soportara.</NOEMBED>
<EMBED SRC="MyMovie.mov" WIDTH="150" HEIGHT="250" CONTROLS="TRUE">
<EMBED SRC="Game.ids" WIDTH="400" HEIGHT="300">
```

<OBJECT>

Este tag puede ser utilizado para embeber cualquier tipo de objetos en las paginas, como por ejemplo: plug-ins, componentes Java, controles ActiveX, applets e imágenes. Siempre se deben indicar el tipo y ubicación (en caso de ser necesaria) del objeto.

Atributos:

- CLASSID: se debe indicar la URL de la implementacion del objeto. Similar al atributo CODE del tag APPLET.
- DATA: apunta la URL de los datos que conforman el objeto.
- CODEBASE: indica la URL del directorio que contiene el archivo de clases y recursos relacionados con el objeto.
- TYPE: indica el tipo MIME de objeto.
- ALIGN: indica la alineacion horizontal del objeto en relacion con el contenido de la pagina.
- HEIGHT: indica el alto maximo a ocupar por el objeto dentro de la pagina.
- WIDTH: indica el ancho maximo del objeto dentro de la pagina.
- ID: indica el nombre del objeto.

Ej:

```
<OBJECT CODEBASE="classes" CLASSID="ticker.obj" DATA="data" WIDTH="150" HEIGHT="250">
```

<SCRIPT>…</SCRIPT>

Define scripts JavaScript a ser ejecutado en el browser, según las acciones realizadas por el usuario que accede a la pagina. Es aconsejable ubicar este codigo en el encabezado de la pagina, para tener una vision mas prolija de su contenido.

Atributos:

- LANGUAGE: permite indicar la version de JavaScript a utilizar.
- SRC: permite cargar el codigo de scripting desde otra pagina.

Ej:

```
<script language=JavaScript>
innerWidth=750;
innerHeight=500;
</script>
```

<NOSCRIPT>…</NOSCRIPT>

Permite ingresar un texto para ser mostrado en en lugar del texto afectado por el scripting para los browser que no permiten ejecutar JavaScript.

<SERVER>…</SERVER>

Indica que el scripting contenido en este tag debe ejecutarse en el server. Esto funciona si se utiliza el Web Server de Netscape, en el caso del IIS los tags utilizados son: **<%>…</%>**.

Otros

<!-- --> Comentario

Todo el texto contenido entre estos tags no es interpretado por el browser.

Ej: <!-— Esto es un comentario -->

<LAYER>…</LAYER>

Permite posicionar bloques de contenidos, denominados layers. Estos pueden superponerse, ser transparentes u opacos, visibles o invisibles, asi como tambien anidarse. Este tag permite especificar bloques de direccion absoluta.

Atributos:

- ID: indica el nombre del layer.
- LEFT: indica posiciones horizontal y vertical de los layers.
- TOP="pixelPosition"
- PAGEX: indica la posicion horizontal del layer en relacion a la ventana del documento.
- PAGE: indica la posicion vertical del layer en relacion a la ventana del documento.
- SRC: path de la pagina que contiene el contenido del layer.
- Z-INDEX: indica el order en la pila del layer, en valores enteros y positivos.
- ABOVE: indica que el layer actual es creado a continuacion del indicado en este valor en el orden establecido.
- BELOW: indica que el layer actual es creado debajo del indicado, en el orden de layers.
- WIDTH: indica el ancho del layer.
- HEIGHT: indica el alto asignado al layer.
- CLIP: indica el area visible del layer (descripta por 4 valores que representan los pixels que ocupa), que puede ser menor a la definida.
- VISIBILITY: define cuando un layer debe volverse visible. Los valores posibles son: SHOW, HIDDEN, e INHERIT.
- BGCOLOR: señala el color de fondo del layer.
- BACKGROUND: indica una imagen como fondo del layer.
- OnMouseOver: JavaScript que se ejecutara cuando el usuario pase por encima del layer.
- OnMouseOut: JavaScript que se ejecutara cuando el usuario salga de encima del layer.
- OnFocus: JavaScript que se ejecutara cuando el usuario se posicione en el layer.
- OnBlur: JavaScript que se ejecutara cuando el usuario se posicione en otro objeto de la pagina.
- OnLoad: JavaScript que se ejecutara cuando el layer este siendo cargado por el browser.

Ej:

```
<LAYER ID=layer1 TOP=250 LEFT=50 WIDTH=200 HEIGHT=200 BGCOLOR=RED>
  <P>Layer 1</P>
</LAYER>
<LAYER ID=layer2 TOP=350 LEFT=150 WIDTH=200 HEIGHT=200 BGCOLOR=BLUE>
  <P>Layer 2</P>
</LAYER>
<LAYER ID=layer3 TOP=450 LEFT=250 WIDTH=200 HEIGHT=200 BGCOLOR=GREEN>
  <P>Layer 3</P>
</LAYER>
```

<ILAYER>…</ILAYER>

Brinda la misma funcionalidad que el tag <LAYER>, con la diferencia de que las direcciones que se le asignan son relativas.

<NOLAYER>…</NOLAYER>

Es un tag para ingresar texto alternativo para los browsers que no soporten el uso de tags de layer.

Tabla de Colores

Color (constante)	Valor RGB
Aqua	#00FFFF
Black	#000000
Blue	#0000FF
Fuschia	#FF00FF
Gray	#808080
Green	#008000
Lime	#00FF00
Maroon	#800000
Navy	#000080
Olive	#808000
Purple	#800080
Red	#FF0000
Silver	#C0C0C0
Teal	#008080
White	#FFFFFF
Yellow	#FFFF00

Conversion de Caracteres

Para poder ver algunos caracteres (como acentos y simbolos con caracteristicas particulares para el lenguaje HTML) es necesario convertirlos a simbolos o entidades especiales. A continuacion se muestra una tabla con dicha conversion para cada uno de los caracteres conocidos:

Caracter	Decimal	Entidad						
¡	¡	¡	¢	¢	¢

£	£	£
¤	¤	¤
¥	¥	¥
¦	¦	¦
§	§	§
¨	¨	¨
©	©	©
ª	ª	ª
«	«	«
¬	¬	¬
\x7f		
®	®	®
¯	¯	¯
°	°	°
±	±	±
²	²	²
³	³	³
´	´	´
µ	µ	µ
¶	¶	¶
·	·	·
¸	¸	¸
¹	¹	¹
º	º	º
»	»	»
¼	¼	¼
½	½	½
¾	¾	¾
¿	¿	¿
À	À	À
Á	Á	Á
Â	Â	ˆ
Ã	Ã	Ã
Ä	Ä	&Auml
Å	Å	˚
Æ	Æ	Æ
Ç	Ç	Ç
È	È	È
É	É	É
Ê	Ê	Ê
Ë	Ë	Ë
Ì	Ì	Ì
Í	Í	Í
Î	Î	Î
Ï	Ï	Ï
Ð	Ð	Ð
Ñ	Ñ	Ñ
Ò	Ò	Ò
Ó	Ó	Ó
Ô	Ô	Ô
Õ	Õ	Õ
Ö	Ö	Ö
×	×	×
Ø	Ø	Ø
Ù	Ù	Ù

Ú	Ú	Ú
Û	Û	Û
Ü	Ü	Ü
Ý	Ý	Ý
Þ	Þ	Þ
ß	ß	ß
à	à	à
á	á	á
â	&#226	â
ã	ã	ã
ä	ä	ä
å	å	å
æ	æ	æ

ç	ç	ç
è	è	è
é	é	é
ê	ê	ê
ë	ë	ë
ì	ì	ì
í	í	í
î	î	î
ï	ï	ï
ð	ð	&ieth;
ñ	ñ	ñ
ò	ò	ò
ó	ó	ó

ô	ô	ô
õ	õ	õ
ö	ö	ö
÷	÷	÷
ø	ø	ø
ù	ù	ù
ú	ú	ú
û	û	û
ü	ü	ü
Ý	ý	ý
Þ	þ	þ
Ÿ	ÿ	ÿ

X Apendice B: JAVASCRIPT - JSCRIPT

Javascript y JScript son dos lenguajes muy similares. El primero fue desarrollado por Netscape y en una primera instancia solo se utilizaba dentro de las paginas HTML, aportando algo de logica al codigo HTML que solo cubria necesidades de formato de texto. Actualmente Netscape utiliza este mismo lenguaje para desarrollar desde su Web Server aplicaciones dinamicas. JScript es un lenguaje desarrollado por Microsoft, practicamente igual al Javascript, pero solo se utiliza del lado del server, no es interpretado por el browser como el primero.

Si bien el uso de ambos lenguajes tiene distintos ambitos, las sentencias, operadores, objetos, etc. son los mismos. Esto facilita el desarrollo en terminos generales, pero siempre hay que tener en claro donde se ejecuta cada linea de codigo: es decir, si sera ejecutada por el Web Server, al momento de requerirse una pagina, o si viajara junto con el codigo HTML para ser ejecutada luego, en el browser, frente a determinados eventos del usuario.

Operadores:

Categoria	Operador	Descripcion
Aritmetico	+	Suma dos numeros
	++	Incrementa en uno una variable numerica
	-	Niega el valor de un argumento o resta dos numeros
	--	Decrementa en uno una variable numerica
	*	Multiplica dos numeros
	/	Divide dos numeros
	%	Devuelve el resto del cociente entre dos numeros
String	+	Concatena dos strings
	+=	Concatena dos strings asignando el resultado al primero
Logico	&&	AND logico
	||	OR logico
	!	Negacion logica
Asignacion	=	Asigna al primer operando el contenido del segundo
	+=	Suma dos numeros y asigna el resultado al primero
	-=	Resta dos numeros y asigna el resultado al primero
	*=	Multiplica dos numeros y asigna el resultado al primero
	/=	Divide dos numeros y asigna el resultado al primero
	%=	Asigna el modulo de los dos operandos al primero
Comparacion	==	Retorna verdadero si los operandos son iguales
	!=	Retorna verdadero si los operandos son distintos
	>	Retorna verdadero si el operando de la izquierda es mayor que el de la derecha
	>=	Retorna verdadero si el operando de la izquierda es mayor o igual que el de la derecha
	<	Retorna verdadero si el operando de la izquierda es menor que el de la derecha
	<=	Retorna verdadero si el operando de la izquierda es menor o igual que el de la derecha
Especiales	?:	permite escribir una sentencia if - else simplificada

	,	Evalua dos expresiones y retorna el resultado de la segunda
	delete	borra una propiedad de un objeto o un elemento de un array
	new	crea una nueva instancia de un objeto
	this	palabra clave que permite la referencia al objeto actual
	typeof	indica el tipo de dato de un determinado operando
	void	indica una expresion que sera evaluada pero no retornara datos

Sentencias

- **Break:** termina un lazo **while** o **for**, transfiriendo el control del programa a la sentencia siguiente del lazo. Tambien se puede pasar como argumento el nombre de un bloque de sentencias, siempre que se hayan identificado previamente.

  Sintaxis:

  ```
  break
  break label
  ```

- **Comment:** permite ingresar un texto de comentario en referencia al codigo escrito.

  Sintaxis:

  ```
  // texto de comentario
  /* comentario que puede
        contener mas de una linea */
  ```

- **Continue:** termina la ejecucion de un bloque de sentencias **while** o **for**, continuando con la ejecucion del lazo para la siguiente interaccion.

  Sintaxis:

  ```
  continue
  continue label
  ```

- **Delete:** borra una propiedad de un objeto o un elemento de un array.

  Sintaxis:

  ```
  delete objectName.property
  delete objectName[index]
  delete property
  ```

- **Do ... While:** ejecuta las sentencias contenidas hasta que la evaluacion de la condicion resulte falsa. Las sentencias que lo componen se ejecutaran al menos la primera vez, antes de evaluar la condicion.

  Sintaxis:

  ```
  do
        Sentencias
  ```

while (condicion)

- **For:** crea un loop basado en tres expresiones principales opcionales, separadas por el simbolo ';', y cada una puede contener una o mas sentencias.

  Sintaxis:

  for (expresion inicial ; condicion ; incremento de la expresion inicial){
  sentencias
  }

  donde:

  - Expresion inicial es el conjunto de variables que deben inicializarse antes de ejecutar el lazo,
  - Condicion es la sentencia que se debe cumplir para que se vuelva a ejecutar el lazo,
  - Incremento permite actualizar los valores que permiten controlar la condicion del lazo, y
  - Sentencias es el conjunto de pasos a realizar dado el cumplimiento de la condicion.

- **For ... in:** itera una variable especifica para todas las propiedades de un objeto, ejecutando para cada una de ellas las sentencias especificadas.

  Sintaxis:

  for (variable in objeto){
  sentencias
  }

- **Function:** declara una funcion JavaScript que pueda aceptar parametros, ya sean strings, numeros y objetos. Pueden contener hasta 255 parametros, se pasan a la funcion por valor y no por referencia, y no son obligatorios.

  Sintaxis:

  function nombre (param1, param2, ..., param n){
  sentencias
  }

- **If ... else:** ejecuta el primer conjunto de sentencias si la condicion se cumple, caso contrario ejecutara las sentencias declaradas dentro del bloque else.

  Sintaxis:

  if (condicion){
  sentenciasIF
  }
  else{
  sentenciasELSE
  }

- **Labeled:** es una etiqueta asociada a un bloque de sentencias, que utilizan las sentencias **break** o **continue** para continuar el flujo de ejecucion.

Sintaxis:

label:

sentencias

- **Return:** indica el valor que debe devolver una funcion.

Sintaxis:

return expresion

- **Switch:** permite evaluar una expresion y ejecutar un conjunto definido de sentencias para cada ocurrencia de la misma.

Sintaxis:

```
switch (expresion){
        case expresion1:
                sentencias;
                break;
        case expresion2:
                sentencias;
                break;
        ...
        default: sentencias;
}
```

- **Var:** declara y eventualmente inicializa una variable.

Sintaxis:

```
var nombre;
var nombre=valor;
```

- **While:** encierra un lazo de sentencias, que se ejecutaran solo si la condicion de entrada a este es verdadera, y se repetira la ejecucion mientras se mantenga este resultado para la condicion.

Sintaxis:

```
while (condicion){
        sentencias
}
```

- **With:** establece un objeto default para un conjunto de sentencias. Dentro de dicho conjunto, cualquier propiedad que no tenga un objeto asociado se asume propia del objeto default.

Sintaxis:

```
with (objeto){
        sentencias
}
```

Tipos de Datos

- **Array:** representa una colección de elementos, a los cuales se puede acceder indicando su posicion dentro del array, para lo cual se debe considerar que los elementos se enumeran a partir de la posicion cero.

  Sintaxis:

  new Array(cantidad de elementos);

  new Array(lista de elementos);

  Propiedades:

  - Index: indice del array.
  - Input: representa el string original con que fue creado el array.
  - Length: cantidad de elementos del array.
  - Prototipe: permite agregar propiedades a un objeto array.

  Metodos:

  - Concat: junta dos arrays devolviendo uno nuevo.
  - Join: junta todos los elementos del array en un string.
  - Pop: remueve el ultimo elemento de un array devolviendo dicho elemento.
  - Push: agrega uno o mas elementos a un array, retornando el ultimo elemento sumado.
  - Reverse: devuelve un array con sus elementos ordenados en sentido contrario, es decir, el primer elemento en ultimo lugar y asi con los demas.
  - Shift: extrae el primer elemento del array.
  - Slice: devuelve un nuevo array extrayendo una seccion de uno existente.
  - Splice: agrega o quita elementos de un array.
  - Sort: ordena los elementos de un array.
  - ToString: retorna un string representando el array.
  - Unshift: suma elementos al comienzo del array y devuelve el nuevo tamaño

- **Boolean:** es un objeto creado para almacenar un valor booleano.

  Sintaxis:

  new Boolean(valor);

  Propiedades:

  - Prototipe: permite agregar propiedades a ser utilizadas por cualquier objeto booleano.

  Metodos:

  - ToString: retorna un string representando el objeto booleano.

- **Date:** objeto que permite trabajar con datos de tipo fecha y hora.

Sintaxis:

new Date()

new Date("mes dia, año horas:minutos:segundos")

new Date(num_año, num_mes, num_dia)

new Date(num_año, num_mes, num_dia, num_hora, num_minutos, num_segundos)

Propiedades:

- Prototipe: permite agregar propiedades a ser utilizadas por cualquier objeto Date.

Metodos:

- getDate: retorna el dia del mes de una fecha determinada.
- getDay: retorna el dia de la semana de una fecha especificada.
- getHours: retorna la hora de un dato de tipo Date.
- getMinutes: retorna los minutos de una fecha especificada.
- getMonth: retorna el mes de una fecha determinada.
- getSeconds: retorna los segundos de una fecha.
- getTime: retorna la hora de una fecha especificada.
- getTimesoneoffset: expresa en minutos la diferencia entre la hora local y el meridiano de Greewnwich.
- getYear: retorna el año de una fecha.
- parse: retorna el numero de milisegundos del intervalo entre una fecha dada y el 1/1/1970, en fecha local.
- setDate: setea el dia del mes de una fecha determinada.
- setHours: setea la hora de un dato de tipo Date.
- setMinutes: setea los minutos de una fecha especificada.
- setMonth: setea el mes de una fecha determinada.
- setSeconds: setea los segundos de una fecha.
- setTime: setea la hora de una fecha especificada.
- setYear: setea el año de una fecha.
- toGMTString: convierte una fecha a string, usando las convenciones GMT.
- toLocaleString: convierte una fecha a string, usando las convenciones locales.
- UTC: retorna el numero de milisegundos del intervalo entre una fecha dada y el 1/1/1970, respetando los codigos GMT.

- **Function:** indica un string de JavaScript que debera ser compilado como una funcion.

Sintaxis:

new Function (arg1, arg2, ... argN, functionBody);

Propiedades:

- arguments: un array conteniendo los argumentos pasados a la funcion.
- arity: indica el numero de argumentos esperados en la funcion.
- caller: contiene la funcion que llamo a la actual
- prototipe: permite agregar propiedades a un objeto Function.

Metodos:

- ToString: retorna un string representando el objeto Function.

- **Math:** es un objeto propio del lenguaje que contiene propiedades y metodos para constantes y funciones matematicas.

Ej:

```
with (Math) {
  a = PI * r*r
  y = r*sin(theta)
  x = r*cos(theta)
}
```

Propiedades:

- E: constante de Euler, base de los logaritmos naturales, aproximadamente 2.718.
- LN10: logaritmo natural de base 10, aproximadamente 2.302.
- LN2: logaritmo natural de base 2, aproximadamente 0.693.
- LOG10E: logaritmo natural de base 10 de E, aproximadamente 0.434.
- LOG2E: logaritmo natural de base 2 de E, aproximadamente 1.442.
- PI: radio de la circunferencia de un circulo hacia su diametro, aproximadamente 3.14159.
- SQRT1_2: raiz cuadrada de $\frac{1}{2}$, o el equivalente de 1 sobre la raiz cuadrada de 2, aproximadamente 0.707.
- SQRT2: raiz cuadrada de 2, aproximadamente 1.1414.

Metodos:

- abs: valor absoluto de un numero.
- acos: arco coseno en radianes de un numero.
- asin: arco seno en radianes de un numero.
- atan: arco tangente en radianes de un numero.
- atan2: arco tangente del cociente de sus argumentos.
- ceil: retorna el menor entero mayor o igual que un numero.
- cos: coseno de un numero.
- exp: retorna el exponencial de un numero, o e elevado a un numero, donde e es la constante de Euler.
- floor: retorna el mayor entero menor o igual que un numero.
- log: logaritmo natural en base E de un numero.
- max: retorna el maximo de dos numeros.
- min: retorna el minimo de dos numeros.
- pow: retorna la base de una potencia exponencial, es decir, la base elevada al exponente.
- random: retorna un valor aleatorio comprendido entre cero y uno.
- round: retorna el valor de un numero redondeado al entero mas cercano.
- sin: seno de un numero.
- sqrt: raiz cuadrada de un numero.
- tan: tangente de un numero.

- **Number:** es un objeto que permite trabajar con valores numericos.

Sintaxis:

new Number (value);

Propiedades:

- MAX_VALUE: el maximo numero representable.
- MIN_VALUE: el minimo numero representable.
- NaN: valor especial representado 'Not a Number'.
- NEGATIVE_INFINITY: valor infinito especial negativo, retornado en overflow.
- POSITIVE_INFINITY: valor infinito especial positivo, retornado en overflow.
- prototipe: permite agregar propiedades a un objeto Number.

Metodos:

- toString: retorna un string representando el objeto Number.

- **Object:** es el tipo de datos primitivo de JavaScript, es decir, que todos los objetos de JavaScript son generados a partir de este.

Sintaxis:

new Object ();

Propiedades:

- Constructor: indica la funcion que crea el nuevo objeto prototipo.
- prototipe: permite agregar propiedades a cualquier objeto.

Metodos:

- eval: evalua un string de codigo JavaScript en el contexto del objeto especificado.
- toString: retorna un string representando el objeto especificado.
- unwatch: remueve un puntero a una propiedad de un objeto.
- valueOf: retorna el valor primitivo del objeto especificado.
- watch: agrega un puntero a una propiedad de un objeto.

- **String:** un objeto conteniendo una serie de caracteres conforma un string.

Sintaxis:

new String(string);

Propiedades:

- length: contiene el tamaño del string.
- prototipe: permite agregar propiedades a un objeto String.

Metodos:

- anchor: crea un objeto 'anchor' en HTML.
- Big: muestra el texto en un tamaño mas grande de letra.
- Blink: muestra el texto con el efecto de parpadeo semejante al tag <BLINK>.
- Bold: muestra el texto en negrita.
- CharAt: retorna el carácter correspondiente a un indice especificado dentro del string.
- CharCodeAt: retorna la posicion de un carácter en un string.
- Concat: concatena dos strings.

- Fixed: muestra un texto en un ancho prefijado, semejante al efecto del tag <TT>.
- Fontcolor: muestra un string en un color especificado.
- Fontsize: muestra un string en el tamaño especificado.
- FromCharCode: devuelve un string formado por los caracteres correspondientes a los valores ISO indicados.
- IndexOf: retorna la posicion en el indice de la primer ocurrencia de un carácter indicado.
- Italics: muestra un string en tipo de letra italico.
- Link: crea un link HTML para el string especificado. Recibe como parametro la URL correspondiente.
- Match: busca una cadena de caracteres determinada dentro de un string.
- Replace: busca y reemplaza una cadena dada de caracteres por otra dentro de un string.
- Search: busca un string dentro de otro string.
- Slice: extrae una seccion de un string retornando uno nuevo.
- Small: muestra el string en una letra menor a la actual.
- Split: convierte un string conteniendo una colección de strings (separados por un carácter determinado) en un array.
- Strike: muestra el string con una linea arriba (como denotando texto tachado).
- Sub: muestra el texto en formato subindice.
- Substr: retorna un substring del original, desde y hacia los indices especificados.
- Sup: muestra el texto en formato superindice.
- ToLowerCase: retorna el string con todos sus caracteres convertidos a minusculas.
- ToUpperCase: retorna el string con todos sus caracteres convertidos a mayusculas.

Objetos del Documento

Estos objetos se identifican con objetos propios de HTML, y el sentido de que exitan a su vez en este lenguaje se debe a que desde un script se pueden modificar los atributos de los mismos, aun cuando inicialmente hayan sido definidos en modo estatico desde el HTML.

- **Anchor:** implementa un anchor de HTML.

Sintaxis:

TheString.anchor(nameAttribute)

Donde

- ✓ theString es un objeto string
- ✓ nameAttribute es un string

- **Applet:** incluye un applet Java en una pagina. Soporta todas las propiedades y metodos propias del tag <APPLET> de HTML.

- **Area:** define un area de una imagen como mapa de links.

- **Document:** contiene informacion sobre el documento actual, conteniendo metodos para modificar el mismo.

  Manejadores de Eventos:
  - ✓ onClick
  - ✓ onDblClick
  - ✓ onKeyDown
  - ✓ onKeyPress
  - ✓ onKeyUp
  - ✓ onMouseDown
  - ✓ onMouseUp

  Propiedades:
  - alinkColor: especifica el atributo ALiNK.
  - anchors: es un array que contiene la coleccion de anchors contenida en la pagina.
  - applets: array que contiene los datos de todos los applets incluidos en la pagina.
  - bgColor: especifica el atributo BGCOLOR.
  - cookie: agrega una cookie a la pagina.
  - domain: contiene el nombre del dominio del server donde se encuentra almacenada la pagina.
  - embeds: es un array que guarda una entrada para cada plug-in incluido en el documento.
  - fgColor: indica el atributo TEXT.
  - formName: indica el nombre de un form contenido en la pagina.
  - forms: array almacenando los datos de cada form dentro de la pagina.
  - images: array conteniendo cada imagen contenida en el documento.
  - lastModified: fecha de la ultima actualizacion de la pagina.
  - layers: array de layers contenidos en el documento.
  - linkColor: especifica el atributo LINK.
  - links: array de links que conforman la pagina.
  - plugins: array de plug-ins contenidos en el documento.
  - referrer: contiene la URL de la pagina.
  - tittle: contiene el tag TITTLE del documento.
  - URL: indica la URL completa del documento.
  - vlinkColor: indica el atributo VLINK.

  Metodos:
  - captureEvents: configura el documento para capturar todos los eventos de un tipo determinado.
  - close: cierra una sesion de pagina, ventana, etc.
  - getSelection: devuelve un string conteniendo el texto seleccionado.
  - handleEvent: solicita el servicio de captura de eventos.

- open: abre una pagina.
- releaseEvents: setea el documento o ventana para soltar los eventos capturados de un tipo especificado.
- routeEvent: envia los eventos capturados a un punto especifico, salteando la jerarquia de Javascript.
- write: escribe expresiones en HTML en un documento especificado.
- writeln: escrite texto HTML en una pagina, terminandolo con un caracter de nueva linea.

Objetos de la Ventana del Browser

- **Frame:** especifica una ventana que puede mostrar varias paginas independientes.

- **History:** contiene un array de informacion sobre las URL visitadas ultimamente por el usuario del browser.

  Sintaxis:
  History.go(indice del array)

  Propiedades:
  - current: la URL de la pagina actual.
  - length: numero de entradas almacenadas.
  - next: URL siguiente a la actual.
  - previous: URL anterior a la actualmente visitada.

  Metodos:
  - back: trae la URL anterior a la pagina actual.
  - forward: carga la URL siguiente a la actualmente visitada.
  - go: trae la URL de un determinado orden dentro del array.

- **Location:** contiene informacion sobre la URL actual.
  Sintaxis:
  Window.location.href="http://default.htm"

  Propiedades:
  - hash: indica el nombre de un anchor en la URL.
  - host: contiene el nombre de host y de dominio o la IP address de un host de la red
  - hostname: indica la porcion host:port de la URL.
  - href: indica la URL completa
  - pathname: indica la porcion URL-path de la URL.
  - port: contiene el puerto de comunicaciones que utiliza el servidor.
  - protocol: indica el comienzo de la URL, incluidos los dos puntos.
  - search: especifica un query.

  Metodos:
  - reload: fuerza una recarga de la pagina actual.

- Replace: reemplaza la pagina actual por la indicada.

- **Screen:** es un conjunto de propiedades que describen la pantalla y los colores utilizados.

  Propiedades:

  - availHeight: indica el alto de la pantalla, en pixels, sin contar el espacio requerido para los elementos propios del sistema operativo (ej: barra de Windows).
  - availWidth: indica el ancho de la pantalla, en iguales condiciones que la propiedad anterior.
  - ColorDepth: indica cual es la paleta de colores que se esta utilizando.
  - Height: muestra el alto de la pantalla.
  - PixelDepth: indica la cantidad de bits x pixel utilizados en la pantalla (resolucion).
  - Width: muestra el ancho de la pantalla.

- **Window:** representa una ventana del browser o un frame. Es el objeto principal, que contiene los objetos documento, location y history.

  Propiedades:

  - closed: indica que una ventana determinada ha sido cerrada.
  - defaultStatus: contiene el mensaje default a mostrar en la barra de estado del browser.
  - document: contiene informacion acerca del documento actual y provee metodos para enviar informacion a ser mostrada en la pagina.
  - frames: array conteniendo los frames que conforman la ventana.
  - history: contiene informacion acerca de las paginas que visito el usuario.
  - innerHeight: indica la dimension vertical, en pixels, del area de contenido de la ventana.
  - innerWidth: indica la dimension horizontal, en pixels, del area de contenido de la ventana.
  - length: contiene el numero de frames de la ventana.
  - location: contiene informacion acerca de la URL actual.
  - locationbar: representa la barra de direccion del browser.
  - menubar: representa la barra de menu del browser.
  - name: nombre de la ventana.
  - opener: contiene el nombre de la ventana del documento que llamo a la actual, si esta fue abierta mediante el uso del metodo Open.
  - outerHeight: indica la dimension vertical de los limites de la ventana, en pixels.
  - outerWidth: indica la dimension horizontal de los limites de la ventana, en pixels.
  - pageXoffset: indica la posicion actual de la pagina en relacion a la pantalla, en pixels, en el plano de x.
  - pageYoffset: indica la posicion actual de la pagina en relacion a la pantalla, en pixels, en el plano de y.
  - parent: un sinonimo de la ventana o frame que contiene a la actual.
  - personalbar: representa la barra personal del browser.
  - scrollbars: representa la barra de scrolling del browser.

- self: sinonimo de la ventana actual.
- status: indica el mensaje a mostrar en la barra de estado del browser.
- statusbar: representa la barra de estado del browser.
- toolbar: representa la barra de herramientas del browser.
- top: sinonimo de la primer ventana abierta del browser.
- window: sinonimo de la ventana actual.

Metodos:

- alert: muetra un cuadro de dialogo con un mensaje y un boton de OK.
- back: deshace la ultima entrada del history, volviendo a la pagina mostrada previamente.
- blur: remueve el foco del ultimo objeto apuntado.
- captureEvents: configura la ventana para capturar todos los eventos de un tipo determinado.
- clearInterval: cancela un timeout que fue seteado con el metodo setInterval.
- clearTimeout: cancela un timeout que fue seteado con el metodo setTimeout.
- close: cierra la ventana especificada.
- confirm: muestra un cuadro de dialogo con un mensaje y los botones de OK y Cancel.
- disableExternalCapture: deshabilita la captura externa de eventos definida por medio del metodo enableExternalCapture.
- enableExternalCapture: permite a una ventana con frames capturar eventos en paginas cargadas desde diferentes direcciones.
- find: busca un texto especificado en el contenido de una ventana.
- focus: apunta a un objeto especificado.
- forward: carga la siguiente URL contenida en el history del browser.
- handleEvent: invoca un manejador para un evento determinado.
- home: apunta a la pagina que el browser tiene configurada como Home.
- moveBy: mueve la ventana una cantidad especificada de pixels.
- moveTo: mueve la esquina superior izquierda del browser al punto indicado por las coordenadas de la pantalla.
- open: abre una nueva ventana.
- print: imprime los contenidos de la ventana o frame.
- prompt: muestra un cuadro de dialogo con un mensaje y un campo de ingreso de datos.
- releaseEvents: configura la ventana para soltar los eventos capturados de un tipo especificado, enviandolos a quienes deberian haberlos capturado.
- resizeBy: redimensiona la ventana moviendo la esquina inferior derecha tantos pixels.
- resizeTo: redimensiona la ventana segun los nuevos parametros de ancho y alto especificados.
- routeEvent: pasa un evento capturado por encima de la jerarquia definida en Javascript para los objetos involucrados.
- scroll: sube o baja a traves de la pagina hasta la coordinada especificada.
- scrollBy: mueve el area visible de la ventana un numero determinado de pixels.

- scrollTo: mueve el area visible de la ventana hacia el punto de coordenadas de la pantalla especificado.
- setInterval: evalua una expresion o llama a una funcion cada cierta cantidad de milisegundos especificada.
- setTimeout: evalua una expresion o llama a una funcion cada determinado tiempo.
- stop: detiene la bajada actual de una pagina.

Formularios

Estos objetos se relacionan con sus correspondientes en HTML, permitiendo administrarlos dinamicamente.

- **Button:** boton de un formulario.

  Propiedades:
  - form: indica a que formulario corresponde el boton especificado.
  - name: nombre del boton.
  - type: refleja el atributo TYPE
  - value: refleja el atributo VALUE

  Metodos:
  - blur: elimina el foco del boton.
  - click: simula un click de mouse sobre el boton.
  - focus: enfoca la accion a ejecutarse sobre el boton.
  - handleEvent: llama al manejador para el evento determinado.

- **Checkbox:** checkbox de un formulario.

  Propiedades:
  - checked: propiedad binaria que indica el estado del objeto.
  - defaultChecked: refleja el atributo CHECKED.
  - form: indica el form que contiene el objeto checkbox.
  - name: refleja el atributo NAME.
  - type: refleja el atributo TYPE.
  - value: refleja el atributo VALUE.

  Metodos:
  - blur: elimina el foco del checkbox.
  - click: simula un click de mouse sobre el checkbox.
  - focus: enfoca la accion a ejecutarse sobre el checkbox.
  - handleEvent: llama al manejador para el evento determinado.

- **FileUpload:** un elemento de bajada de archivo de formulario.

  Propiedades:

- form: indica el form que contiene el objeto archivo.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE.
- value: refleja el atributo VALUE.

Metodos:

- blur: elimina el foco del objeto.
- click: simula un click de mouse sobre el objeto.
- focus: enfoca la accion a ejecutarse sobre el objeto.
- handleEvent: llama al manejador para el evento determinado.

- **Form:** define un formulario HTML.

Propiedades:

- action: refleja el atributo ACTION.
- elements: array conteniendo los elementos que componen el formulario.
- encoding: refleja el atributo ENCTYPE.
- length: cantidad de elementos que contiene el formulario.
- method: refleja el atributo METHOD.
- name: refleja el atributo NAME.
- target: refleja el atributo TARGET.

Metodos:

- handleEvent: llama al manejador para el evento determinado.
- reset: simula un click de mouse sobre el boton de reset del formulario.
- submit: envia la informacion a la aplicacion que procesara el formulario.

- **Hidden:** un objeto input que no se desea mostrar al usuario.

Propiedades:

- form: indica el form que contiene el objeto archivo.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE.
- value: refleja el atributo VALUE.

- **Option:** item de objeto select.

Propiedades:

- defaultSelected: indica el estado de seleccion inicial de la opcion.
- selected: indica el estado de seleccion actual de la opcion.
- text: contiene el texto a mostrar en la lista.
- value: indica el id de la seleccion.

- **Password:** campo de ingreso de datos donde los mismos se reemplazan por el caracter * al ser mostrados en pantalla.

Propiedades:

- defaultValue: refleja el atributo VALUE.
- form: indica el form que contiene el objeto password.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE.
- value: refleja el valor actual del campo.

Metodos:

- blur: elimina el foco del objeto.
- focus: enfoca la accion a ejecutarse sobre el objeto.
- handleEvent: llama al manejador para el evento determinado.
- Select: selecciona el area de ingreso de un objeto.

- **Radio:** conjunto de botones de seleccion excluyente.

Propiedades:

- checked: propiedad binaria que indica el estado del objeto.
- defaultChecked: refleja el atributo CHECKED.
- form: indica el form que contiene el objeto.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE.
- value: refleja el atributo VALUE.

Metodos:

- blur: elimina el foco del objeto.
- click: simula un click de mouse sobre el objeto.
- focus: enfoca la accion a ejecutarse sobre el objeto.
- handleEvent: llama al manejador para el evento determinado.

- **Reset:** boton de reset de un formulario.

Propiedades:

- form: indica a que formulario corresponde el boton especificado.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE
- value: refleja el atributo VALUE

Metodos:

- blur: elimina el foco del boton.
- click: simula un click de mouse sobre el boton.
- focus: enfoca la accion a ejecutarse sobre el boton.
- handleEvent: llama al manejador para el evento determinado.

- **Select:** lista de seleccion de un formulario.

Propiedades:

- form: indica a que formulario corresponde el objeto.
- length: numero de opciones en la lista.

- name: refleja el atributo NAME.
- options: refleja el tag OPTION.
- selectedIndex: refleja el indice de la opcion seleccionada.
- type: refleja el atributo TYPE.

Metodos:

- blur: elimina el foco del boton.
- focus: enfoca la accion a ejecutarse sobre el boton.
- handleEvent: llama al manejador para el evento determinado.

- **Submit:** boton de envio de los datos del formulario a la aplicacion que deba procesarlos..

Propiedades:

- form: indica a que formulario corresponde el boton especificado.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE
- value: refleja el atributo VALUE

Metodos:

- blur: elimina el foco del boton.
- click: simula un click de mouse sobre el boton.
- focus: enfoca la accion a ejecutarse sobre el boton.
- handleEvent: llama al manejador para el evento determinado.

- **Text:** campo de ingreso de texto.

Propiedades:

- defaultValue: refleja el atributo VALUE.
- form: indica el form que contiene el objeto.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE.
- value: refleja el valor actual del campo.

Metodos:

- blur: elimina el foco del objeto.
- focus: enfoca la accion a ejecutarse sobre el objeto.
- handleEvent: llama al manejador para el evento determinado.
- Select: selecciona el area de ingreso de un objeto.

- **Textarea:** campo multilinea de ingreso de texto.

Propiedades:

- defaultValue: refleja el atributo VALUE.
- form: indica el form que contiene el objeto.
- name: refleja el atributo NAME.
- type: refleja el atributo TYPE.
- value: refleja el valor actual del campo.

Metodos:

- blur: elimina el foco del objeto.
- focus: enfoca la accion a ejecutarse sobre el objeto.
- handleEvent: llama al manejador para el evento determinado.
- Select: selecciona el area de ingreso de un objeto.

Browser

- **navigator:** contiene informacion sobre la version de navegador en uso.

  Propiedades:
  - appCodeName: indica el codigo de nombre del browser.
  - appName: nombre del browser.
  - appVersion: informacion acerca de la version del browser.
  - language: indica el idioma de la version.
  - mimeTypes: array de todos los tipos MIME soportados por el browser.
  - platform: contiene el tipo de maquina para la cual fue compilado el browser.
  - plugins: array conteniendo datos sobre todos los plug-ins instalados en el cliente.
  - userAgent: especifica el encabezado del user-agent.

  Metodos:
  - javaEnabled: contiene el estado de este servicio en el cliente.
  - plugins.refresh: deja disponibles los nuevos plugins instalados y opcionalmente reabre los documentos que contienen plugins.
  - preference: permite setear preferencias del browser mediante un script.
  - taintEnabled: indica cuando esta habilitado el servicio de data tainting.

- **MimeType:** representa el tipo de extension de mail utilizada por el usuario.

  Propiedades:
  - description: descripcion del tipo de MIME.
  - enabledPlugin: referencia al objeto plugin configurado para el tipo de MIME.
  - suffixes: string listado las extensiones de archivo posibles para el tipo de MIME.
  - type: nombre del tipo de MIME.

- **Plugin:** representa un modulo de plug-in instalado en el cliente.

  Propiedades:
  - description: descripcion del plug-in.
  - filename: nombre del archivo de plug-in en disco.
  - length: cantidad de elementos plug-in en el array de objetos mimeType.
  - name: nombre del plug-in.

Eventos

Evento	Manejador del Evento	Cuando Ocurre
abort	onAbort	El usuario aborta la carga de una imagen.
blur	onBlur	Un elemento de formulario o una ventana pierden enfoque.
change	onChange	Un elemento de formulario pierde enfoque y su contenido fue modificado.
click	onClick	Se hace click sobre un objeto de formulario.
dblclick	onDblClick	Se realiza un doble click sobre un objeto de formulario.
dragdrop	onDragDrop	El usuario arrastra un objeto dentro de la ventana.
error	onError	La carga de un documento o imagen provoca un mensaje de error.
focus	onFocus	Una ventana, frame u objeto de formulario es enfocado.
keydown	onKeyDown	El usuario presiona una tecla.
keypress	onKeyPress	El usuario presiona o suelta una tecla.
keyup	onKeyUp	El usuario libera una tecla.
load	onLoad	El browser termina la carga de una ventana o de todos los frames que la misma contiene.
mousedown	onMouseDown	El usuario presiona un boton del mouse.
mousemove	onMouseMove	El usuario mueve el cursor.
mouseout	onMouseOut	El cursor libera un area de link o map de la pagina
mouseover	onMouseOver	El cursor pasa por encima de un area de link o map
mouseup	onMouseUp	El usuario presiona un boton del mouse sobre un link o map
move	onMove	El usuario o algun script apuntan hacia otra ventana o frame
reset	onReset	El usuario presiona un boton de 'Reset' (para limpiar campos en un formulario)
Resize	onResize	La ventana o frame son redimensionados, ya sea por accion del usuario o a traves de un script
Select	onSelect	El usuario selecciona texto desde una campo de formulario
Submit	onSubmit	El usuario presiona un boton de 'Submit' (enviando los datos ingresados al servidor)
Unload	onUnload	El usuario sale del documento actual

Particularidades del Jscript

Como dijimos anteriormente, Jscript se ejecuta en el Web Server, y para tales efectos, cuenta con un conjunto de objetos adicionales al lenguaje propiamente dicho. Los mas importantes son:

- Objeto Application: contiene una coleccion de variables que podran ser visualizadas desde cualquier parte de la aplicacion durante todo el tiempo en que este activo el

servicio de Web. Por ejemplo, los parametros generales del sistema se pueden almacenar en variables de Aplicacion, las cuales seran compartidas por todos los usuarios.

- Objeto Request: este objeto se utiliza para capturar informacion enviada desde el Browser al Web Server. Por ejemplo, al enviar la informacion de un formulario a traves de un boton de 'Submit', en la aplicacion esa informacion sera recibida utilizando este comnado.

- Objeto Response: por medio de este objeto enviamos informacion desde el Web Server a la pagina, generalmente enviaremos un string a mostrar en el lugar desde donde se realizo el envio de informacion al Web Server.

- Objeto Server: contiene un conjunto de propiedades que permite parametrizar determinadas acciones realizadas por el Web Server, como por ejemplo fijar un tiempo maximo de espera para la ejecucion de un script.

- Objeto Session: al igual que el objeto Application, contiene un conjunto de variables, solo que en este caso genera una instancia diferente de cada variable para cada sesion de usuario. Se utilizan para almacenar datos propios de la sesion de cada usuario, como por ejemplo su id de usuario, variables necesarias para procesar la informacion ingresada por el, etc.