Comunicación Asíncrona en aplicaciones Web

Autor: Oswaldo Álvarez Calderón

Tesina presentada para obtener el grado de:
Ingeniero en Sistemas Computacionales

Nombre del asesor:
César Augusto Mendoza Morales

UVAQ

UNIVERSIDAD VASCO DE QUIROGA

FACULTAD DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

"Comunicación Asíncrona en aplicaciones Web"

TESINA

PARA OBTENER EL TÍTULO DE

INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTA

Oswaldo Alvarez Calderón

ASESOR

César Augusto Mendoza Moreno

CLAVE: 16PSU0049F

ACUERDO: LIC000808

MORELIA, MICHOACÁN DICIEMBRE 2008

UNIVERSIDAD VASCO DE QUIROGA

Comunicación Asíncrona en aplicaciones Web

Tesina sometida a la Escuela
de Ingeniería en Sistemas computacionales
de la Universidad Vasco de Quiroga

Para obtener el grado de
Ingeniero en Sistemas Computacionales

Presenta

Oswaldo Alvarez Calderón

Morelia, Michoacán, México
Diciembre de 2008

ÍNDICE GENERAL

PLANTEAMIENTO DEL PROBLEMA

Entre las aplicaciones modernas de cómputo, se encuentran las orientadas a Web, las cuales siguen un esquema de conexión cliente-servidor basado en peticiones explícitas y respuestas concretas de información; sin embargo, esta forma de comunicación se lo conoce como sincrónica, es decir, al hacer una petición se obtiene una respuesta y hasta no concluirla no se logrará hacer una nueva petición.

Está comunicación limita a trabajar con recargas de contenidos de páginas Web continuas, por lo que hay tiempos de espera que podrían aprovecharse mientras el resultado es devuelto o mientras se gestiona la información ya obtenida.

Otra observación, es la manera de crear interfaces porque las acciones de su contenido en la mayoría de las ocasiones requieren llamar a otra página que muestre una nueva serie de controles o elementos, cayendo nuevamente en la recarga y por consiguiente en mayor número de tiempos de espera.

Sería recomendable que en lugar de hacer múltiples recargas y llamados a otras páginas, se pudieran crear los controles con sus respectivas acciones de manera dinámica con la finalidad de eliminar el contenido que no sea necesario y ubicar dentro de la página el contenido nuevo.

De ésta manera se ahorrarían muchas recargas y mucho tiempo de espera, tal como lo propone AJAX.

JUSTIFICACIÓN

El tema de este proyecto es de interés para programadores web, ya que muestra una alternativa de desarrollo de aplicaciones Web que se diferencía de las demás por la forma de comunicación asíncrona que utiliza y permite un mejor manejo de contenidos.

Por otro lado, está considerado por la W3C como una forma de programación que permite la creación de aplicaciones Web más amigables y rápidas. Cabe destacar que desde el año 2005, ha marcado una nueva era en los desarrollos Web, permitiendo la evolución de Internet que hace años no se veía. [49]

Por todo lo anterior, es razón suficiente para estudiar este fenómeno que marca una nueva tendencia de programación y ofrece enormes beneficios para los usuarios de la red.

OBJETIVOS

Objetivo General

El objetivo de este proyecto es aprender un nuevo paradigma de desarrollo Web, capaz de establecer comunicación de tipo cliente-servidor de manera asíncrona para ofrecer a los usuarios una mejor experiencia con las aplicaciones Web.

Objetivos Específicos

Describir la evolución de las aplicaciones en internet.
Describir que es y cómo funciona el paradigma de programación AJAX.
Conocer las tecnologías que integran a AJAX.
Determinar las tendencias de la nueva era de internet y AJAX.
Ejemplificar el funcionamiento básico de una aplicación AJAX.

Capítulo 1 INTRODUCCIÓN

Este proyecto tiene como objetivo estudiar una nueva forma de desarrollo Web que posiblemente marque una nueva era para internet, debido a que tiene capacidades parecidas a un programa de computadora convencional, es decir, a una aplicación que se instala en la computadora para trabajar con él de forma independiente, a este tipo de aplicación se le conoce como *stand-alone*.

AJAX acrónimo de JavaScript Asíncrono + XML, es el paradigma a estudiar. Integra varias tecnologías, que unidas, hacen que una aplicación tenga un mayor desempeño y sea más funcional, por lo tanto, crea entornos Web más amigables para el usuario logrando con esto destacar entre las tecnologías de programación en su ramo.

Dentro del capítulo 1 se hace una recopilación de los antecedentes más sobresalientes en la historia de las aplicaciones Web, es importante retomar la base del fenómeno que se va a estudiar de forma en que abra un panorama más amplio para el lector, tomando como punto de partida el inicio de internet y posteriormente pasando a los CGI que más adelante se escribirán.

El capítulo 2, muestra una descripción más clara y profunda, el concepto de aplicación Web, revisando aspectos básicos como los lenguajes de programación usados para su desarrollo, servidores Web como dispensadores del servicio y algunos de los elementos más importantes que se deben de tener en cuenta cuando se está desarrollando la aplicación usando conceptos como interfaz, interactividad y usabilidad.

La esencia principal del proyecto está contenida dentro del capítulo 3, en la cual, se hace una revisión de AJAX como una forma de hacer aplicaciones Web.

En este capítulo se muestra ¿Qué es AJAX? ¿De qué forma trabaja? Algunas de sus ventajas y desventajas frente a las “aplicaciones clásicas” refiriéndose estas últimas,

a las aplicaciones que necesitan de recargas para poder funcionar, la compatibilidad que tiene con los navegadores de internet.

Por otro lado, se hace mención de algunas posibles alternativas a AJAX ya que de alguna forma sirvieron como base para que AJAX existiera y por último, los frameworks que ayudan a los programadores a utilizarlo de una manera más sencilla.

En el capítulo 4 se hace una revisión de todas y cada una de las tecnologías que integran AJAX, de forma que sirva como introducción para el desarrollo de aplicaciones con esta metodología, añadiendo algunos ejemplos para su mejor comprensión y los más importante de todo, el objeto XMLHTTPRequest que hace posible la existencia de AJAX.

El capítulo 5, está destinado a un caso práctico que muestra de una forma muy breve el uso de una aplicación desarrollada con AJAX, enfocándose a la manera en que se hace la conexión.

Y finalmente en el capítulo 6, se establecen algunas conclusiones sobre el tema, comentarios sobre las mejoras de las aplicaciones Web con el uso de AJAX, que se pueden esperar para hacer que internet sea más funcional y visto de otra manera.

AJAX simplemente marca un nuevo rumbo en la historia de internet y por tal motivo, fenómenos como este requieren de atención suficiente para proponer nuevos usos de la red y conocer su desempeño.

Capítulo 2 ANTECEDENTES

Internet inició como un proyecto de defensa de los Estados Unidos a finales de los años 60 por la Agencia de Proyectos de Investigación Avanzados (ARPA) del Departamento de Defensa mejor conocida como ARPAnet en ese entonces.

Este proyecto, es la unión de varias redes de computadoras interconectadas mundialmente, la cual tenía la finalidad de mantener la comunicación entre sus bases aliadas, suponiendo que se aproximaba un ataque nuclear, y de esta manera no perderían conexión alguna de ellas.

En 1975, ARPAnet comenzó a funcionar como red, sirviendo como punto de desarrollo tecnológico de investigaciones militares y universidades. Más tarde se desarrollarían formas de conexión más avanzadas para diferentes tipos de ordenadores y cuestiones más específicas. En 1983 se adoptó el protocolo TCP/IP como estándar principal para todas las comunicaciones. Posteriormente en 1990, ARPAnet y redes funcionalmente similares fueron evolucionando.

Internet está constituido por varios servicios, de los cuales, el servicio WWW (World Wide Web), es uno de los más usados, junto con el correo electrónico. El servicio WWW fue desarrollado por el CERN de Ginebra Suiza a principios de los 90's, creado en base a un sistema de hipermedios distribuidos, es decir, un sistema que distribuye toda su información en módulos, unida mediante vínculos o ligas (links). Esta información puede presentarse de distintas maneras, como archivos ejecutables, de texto, gráficos, audio, vídeo, animación o imagen.

Inicialmente Internet era solamente una colección de páginas estáticas y documentos para consultar y/o descargar. Posteriormente en su evolución, se incluyó un método para elaborar páginas dinámicas que permitía que el contenido fuera generado a partir de los datos de una petición. Este método fue conocido como CGI ("Common

Gateway Interface") y definía un mecanismo mediante el que se podía pasar información entre el servidor y programas externos como C, Perl, Visual Basic, etc.

Los CGI en realidad no son un lenguaje o un protocolo en realidad solo son un conjunto de variables y convenciones, de denominación común, que sirven para pasar información de ida y vuelta al servidor. Los CGI's son la interfaz entre el servidor HTTP del sitio Web y algunos de sus recursos.

Durante algún tiempo fueron bien vistos, hasta que se encontraron limitados; cada vez que el servidor recibía una solicitud que accedía a un programa CGI, debía crear un nuevo proceso para la ejecución de dicho programa, pasando a través de variables de ambiente y entrada de datos estándar, información que pudiera ser necesaria para generar la respuesta, requiriendo de tiempo y significativos recursos del servidor, limitando la cantidad de solicitudes que podían ser procesadas.

Otro problema con el que se topó, radica en el hecho de que un programa CGI no puede interactuar con el servidor web o beneficiarse de las habilidades del servidor una vez comenzada su ejecución, puesto que ésta era realizada en un proceso aparte.

Posteriormente se comenzó a utilizar el *FastCGI* que era una alternativa al CGI estándar, cuya diferencia radicaba en el hecho de que el servidor creaba un único proceso persistente por cada programa *FastCGI* en lugar de por cada solicitud del cliente.

Aunque fue un gran avance dentro de las aplicaciones Web, tuvo problemas en el desarrollo de sus procesos, ya que en todo momento existía un proceso activo por cada programa.

Hoy en día los CGI's han sido casi remplazados por tecnologías completamente dedicadas a la comunicación entre el cliente y el servidor, pero como en la evolución

natural de las aplicaciones Web se encuentra otra problemática con las que se tiene que luchar.

Internet a lo de largo de su historia, generó y seguirá generando necesidades que aportarán beneficios como ya se han presentado con anterioridad, para determinar su mejor funcionamiento, pero no cabe duda que uno de los logros mas acertados fue el de crear tecnologías y métodos para la generación dinámica de contenidos y procesamiento de información y lograr con esto el desarrollo de aplicaciones Web.

Capítulo 3 Comunicación Clásica Cliente-Servidor

2.0 Aplicaciones Web "Clásicas"

Para iniciar con este capítulo es necesario realizar una definición sobre lo que es una *aplicación Web*.

Una *aplicación Web* es un programa informático que permite al usuario utilizar una computadora que obtiene recursos como datos o porciones de código que no necesariamente son instalados localmente, sino que provienen de su conectividad al Internet. Estas aplicaciones se integran al software de una computadora, y se ejecutan generalmente en el servidor donde residen. [3]

2.1 Características

1. Por lo general una aplicación Web siempre se encuentra instalada dentro de un *servidor,* esto le da la ventaja que pueda tener muchos usuarios (clientes) sin importar su ubicación, además de facilitar su actualización ya que todos sus usuarios no tendrán necesidad de molestarse por hacer algún cambio local, todos son distribuidos por la aplicación servidor.

El servidor siempre está en espera de que el navegador (cliente) le haga una petición. Esta petición se realiza utilizando el protocolo HTTP (Hypertext Transfer Protocol), diseñado para transferencia de datos.

Una vez que el servidor recibe la petición, contesta con el envío de datos al navegador, una aplicación capaz de formatear y visualizar dichas respuestas. La figura 2.1 muestra la estructura general de una aplicación Web.

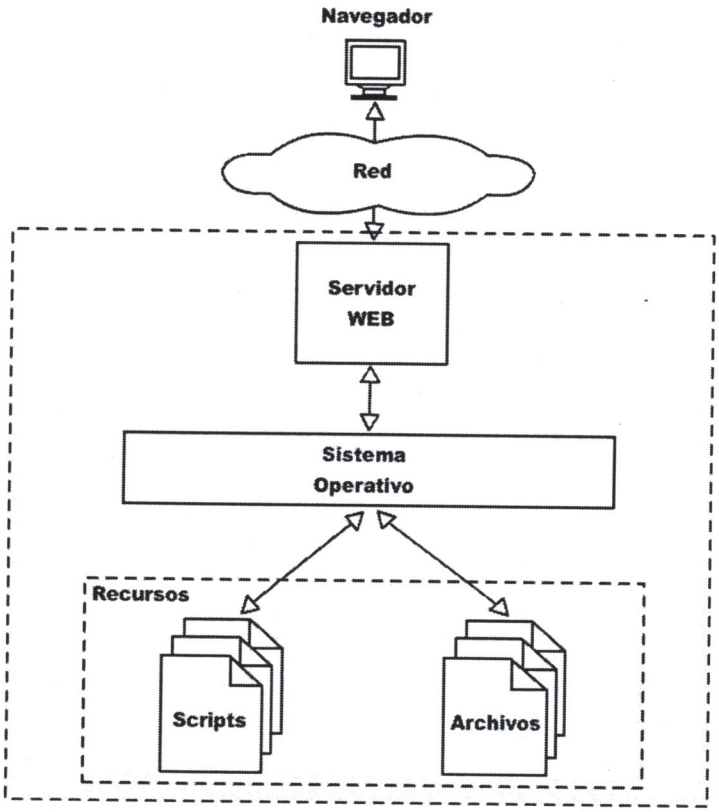

Figura 3.1 Estructura general del funcionamiento de aplicación web.

Servidores Web más utilizados:

- Apache.
- IIS (Internet Information Server).
- Sun.

2. Las aplicaciones Web están constituidas por lenguajes y/o tecnologías de programación que las hacen ser mucho más ágiles ya que están integradas por módulos y presentan la información dinámicamente.

Para que sea posible el funcionamiento de la aplicación, es importante mencionar que las aplicaciones deben de contar con el *lenguaje intérprete,* como lo es el HTML,

ya que lee y ejecuta una serie de instrucciones previamente definidas o creadas dinámicamente, línea por línea, es decir, las interpreta. Los *lenguajes intérprete* se procesan dentro del *navegador* de internet.

Por otro lado se encuentran los lenguajes de programación Web, que se encargan de realizar el manejo de la información, es decir, es una tecnología que realiza la comunicación entre el usuario, que cuenta con un navegador, y el servidor que contiene los recursos necesarios que le solicita el navegador, en este caso, información que es desplegada por HTML; de esta manera funciona una aplicación Web, a través de peticiones y presentaciones de datos.

Para la creación de aplicaciones Web dinámicas se cuenta con varias tecnologías de desarrollo de las cuales se describen las mas comunes.

- **PHP (Hypertext Pre-processor)** Es un lenguaje de programación open source de *scripts*, que se ejecuta en un servidor y es utilizado para la creación de páginas Web dinámicas. Tiene una sintaxis similar a *Perl* o *C* que se incluye entre etiquetas especiales, lo que permite al programador incrustar el código *PHP* en el *HTML* Esta tecnología puede realizar cualquier tarea que un programa *CGI*, pero su fortaleza está en la compatibilidad con los diversos tipos de manejadores de bases de datos.

- **ASP (Active Server Pages)** Son páginas dinámicas que utilizan la extensión ASP y contienen *scripts* de *ActiveX* (Tecnología Microsoft). Son similares a los *CGI* pero permiten a los programadores de Visual Basic trabajar con herramientas familiares. Cabe señalar que ASP sólo puede correr en un servidor Web IIS (Internet Information Server). La excepción es Apache a través del módulo ASP que puede manejar páginas ASP que estén codificadas en perl, emulando a un servidor IIS.

- **JSP (Java Server Pages)** Es una tecnología basada en *Java* que permite a los programadores y realizar páginas Web dinámicas sin la necesidad de conocer este lenguaje de programación a fondo. La fortaleza de esta tecnología es la facilidad de uso y la incorporación de ciertos componentes como los *javabeans.*

En la figura 2.2 se muestra el funcionamiento de una aplicación Web clásica basada en un servidor.

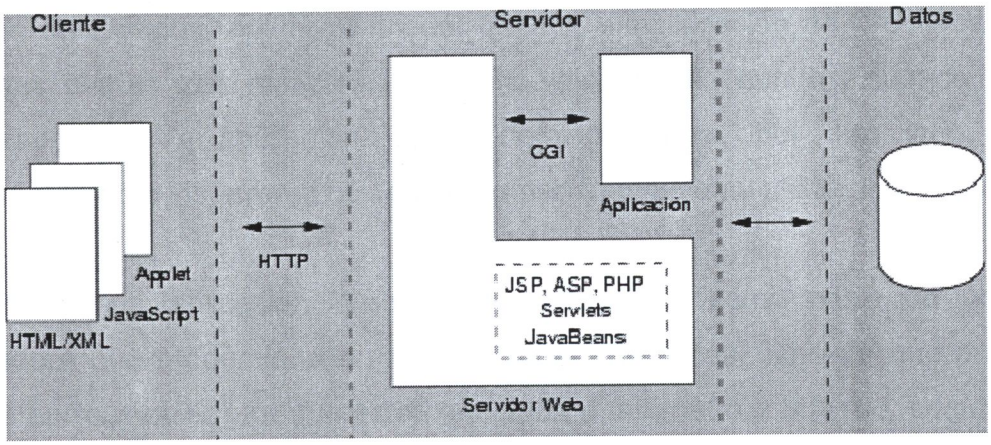

Figura 3.2 Funcionamiento de una aplicación web clásica.

3. Es necesario que la aplicación Web tenga una *Interfaz* amigable con el usuario para su desempeño óptimo y funcional.

Una buena *interfaz Web* se basa en el uso de herramientas contextuales o gráficas que son desplegadas en la pantalla para permitir al usuario identificar los contenidos, con la finalidad de controlar el acceso a la información y dar continuidad al proceso que se sigue dentro de la aplicación.

El diseño de la *interfaz*, facilita el despliegue de los contenidos en el momento que se desee, poniendo a disposición del usuario la información que solicite, con sencillez y claridad.

Dentro de la *interfaz* se debe tomar en cuenta dos aspectos muy importantes que son la *interactividad* y la *usabilidad*, contribuyendo a que no sea tediosa la experiencia que se tiene con la aplicación en desarrollo.

La *interactividad* y *usabilidad* brindan al usuario sencillez y claridad con la que puede ser logrado el exitoso funcionamiento de la aplicación Web enfocándose directamente a las necesidades del usuario.

A demás, aporta grandes beneficios como lo son la facilidad de aprendizaje, reducción de tiempos, optimización del trabajo, simplificación de procesos, en las actividades diarias que realizará el usuario, alentándolo aún así cuando no esté familiarizado con la aplicación, brindándole confianza, por otro lado, elimina considerablemente los errores que se comenten mediante su utilización. [4]

En esencia, una aplicación Web puede concebirse como parte de una gran computadora que proporciona un recurso de software casi ilimitado que puede ser accedido por cualquier otra que este conectada a internet o a una red, según sea su función.

Cada aplicación debe cumplir con ciertas características como lo son interactividad, usabilidad, funcionalidad, que sean predictivas para que su funcionamiento sea bueno, pero cabe aclarar que puede haber algunas con un grado de funcionalidad aceptable, pero no muy fáciles de usar para la persona que va a trabajar con ellas, es por esto que se debe contemplar una buena interfaz, que ayude al usuario a hacer correctamente los procedimientos para lograr sus objetivos y reflejar los resultados de la aplicación.

Capítulo 4 AJAX – Comunicación Asíncrona

3.0 Introducción

En la actualidad, existen otras técnicas para desarrollar aplicaciones Web que rompen el esquema del funcionamiento de una aplicación clásica, brindando al usuario una experiencia mucho mas amigable, sencilla y ágil para trabajar, una de ellas es AJAX acrónimo de Asynchronous JavaScript + XML, que es el tema principal de esta tesina.

Es un paradigma de programación para páginas Web compuesta de varias tecnologías, XHTML, CSS, Document Object Model (DOM), el objeto *XMLHttpRequest* y XML, creado en el 2005 por Jesse James Garrett.

3.1 Historia

AJAX realmente no tiene una historia como tal; desde 1998 se ha venido explorando formas para hacer que las aplicaciones Web requirieran menos demora en la carga de información o en la respuesta de sus procesos.

El desarrollo de las tecnologías que componen esta técnica de programación de páginas Web, se remonta hace más de una década con la iniciativa del desarrollo del Scripting Remoto que Microsoft inició.

Microsoft incluyó desde el Internet Explorer 4, el *Microsoft's Remote Scripting* que tenía la función de enviar datos a través de un *applet JAVA* el cual se podía comunicar con el cliente usando *JavaScript*. Posteriormente este elemento fue implementado en el Netscape 4, donde tuvo una funcionalidad bastante aceptada.

En el 2002, aproximadamente, Jesse James Garrett realizó una modificación al *Microsoft's Remote Scripting* para reemplazar el applet Java por *XMLHttpRequest*

manejado por *JavaScript,* además de darle el nombre de AJAX y darlo a conocer ante la industria de desarrolladores Web.

Desde entonces el objeto *XMLHttpRequest* ha sido implementado en la mayoría de los navegadores y ha sido mas frecuente su uso dentro de las aplicaciones Web. Sin embargo, todavía se utiliza donde se requiere una mayor compatibilidad, una reducida implementación, o acceso cruzado entre sitios Web.

3.2 Definición de AJAX

Cuando se habla del modelo clásico de aplicaciones Web, se entiende que se realiza una petición HTTP por medio de una *interfaz*. El servidor efectúa varios procesos (recopilación de información, proceso de operaciones, comunicación con el servidor) y devuelve una página HTML al cliente para posteriormente ser visualizada por el navegador.

Este modelo es funcional para el manejo de hipertexto, pero no es necesariamente bueno para las aplicaciones Web ya que, mientras el servidor está haciendo lo suyo, el usuario lo único que hace es esperar por la respuesta y en cada petición que solicita el usuario tiene que seguir esperando.

Una forma ideal de poder trabajar con aplicaciones Web podría ser que al momento que se carga la *interfaz*, la interacción del usuario no debería detenerse al momento que la aplicación necesite algo del servidor, sino que pudiera continuar con su interacción sin necesidad de esperar.

En la figura 3.1 se muestra una comparativa entre una aplicación clásica y una hecha con AJAX.

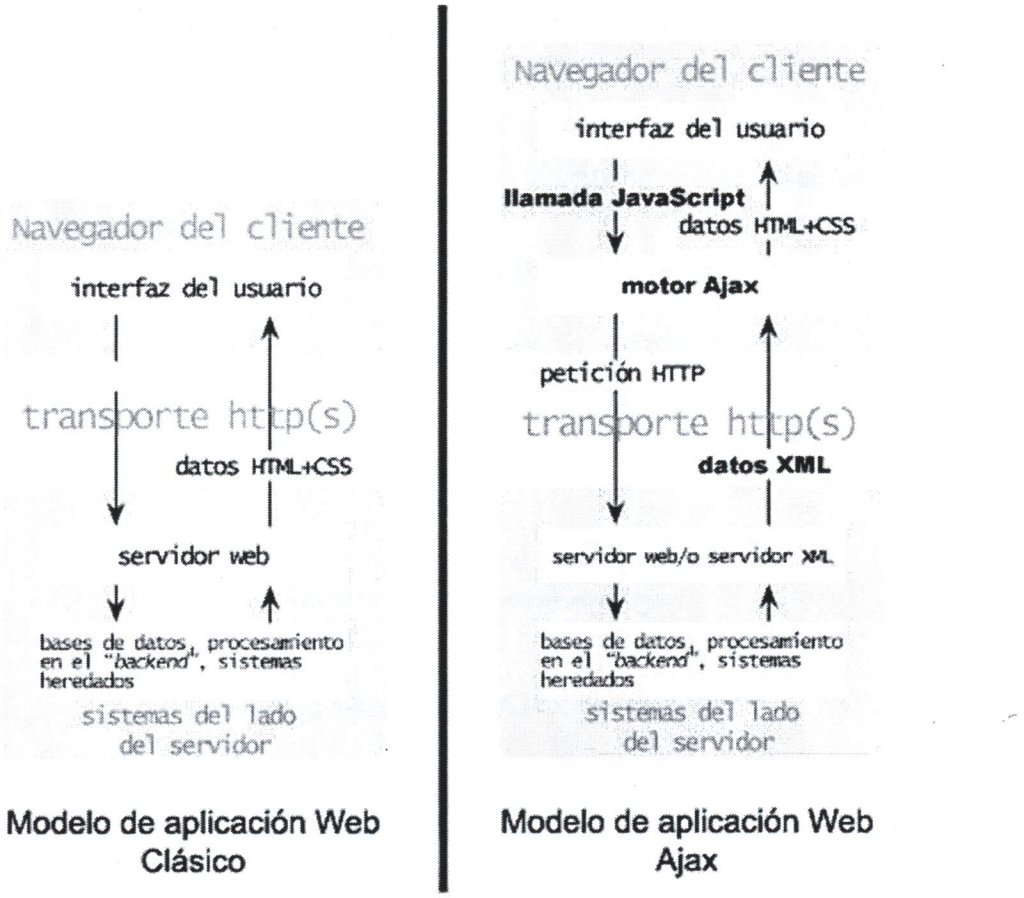

Figura 4.1 Funcionamiento del modelo clásico y modelos AJAX.

Una aplicación AJAX elimina la naturaleza de la recarga completa de la interacción en la Web, introduciendo un intermediario, un motor AJAX, entre el usuario y el servidor. Se podría pensar que al añadir este motor la aplicación produciría más retrasos y menos respuestas; sin embargo, sucede todo lo contrario.

En vez de cargar un página Web, al inicio de una sesión, el navegador carga el motor AJAX (escrito en JavaScript, en un *frame* oculto). Este motor es el responsable de recargar la *interfaz* que el usuario ve y de comunicarse con el servidor.

El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el

usuario rara vez estará mirando una ventana en blanco del navegador o un icono de reloj de arena esperando a que el servidor haga algo.

En la figura 3.2 se muestra como se comunican las aplicaciones clásicas y AJAX.

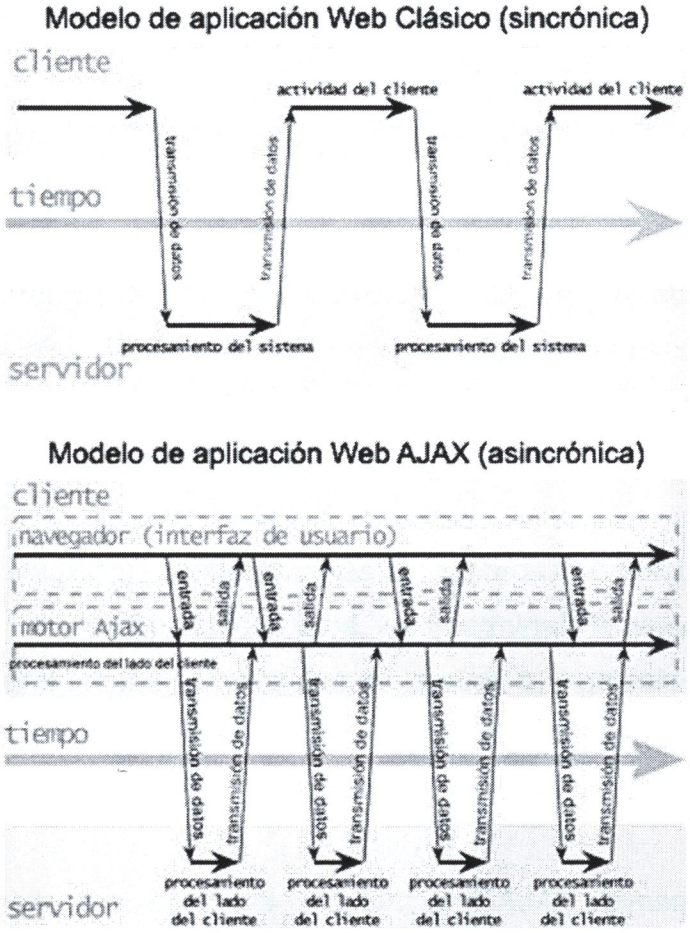

Figura 4.2 Comunicación sincrónica y asincrónica.

Cada acción de un usuario, que normalmente generaría un requerimiento HTTP, toma la forma de un llamado JavaScript al motor AJAX, en vez de ese requerimiento. Cualquier respuesta a una acción del usuario que no requiera un viaje de vuelta al servidor (como una simple validación de datos, edición de datos en memoria, incluso algo de navegación) es manejado por su cuenta.

Si el motor necesita algo del servidor para responder (sea enviando datos para procesar, cargar código adicional, o recuperando nuevos datos) hace esos pedidos asincrónicamente, usualmente usando XML, sin frenar la interacción del usuario con la aplicación.

3.3 Situación Actual

En la actualidad AJAX es, sin lugar a dudas, una de las novedades más atractivas y prometedoras de la denominada Web 2.0 Este paradigma redefine el término "interactividad", pasando a un nivel superior al que comúnmente se conoce.

La Web 2.0 es la representación de la evolución de las aplicaciones tradicionales hacia aplicaciones Web enfocadas al usuario final, donde el usuario tiene un papel muy importante, ya que tiene control sobre la aplicación haciéndose mayormente independiente del desarrollador.

Con este nuevo paradigma, quedarán atrás las recargas completas de las páginas, que permiten el envío y recepción de datos, los interminables listados de información que el navegador presentaba, información de la cual, tal vez sólo se necesitaba una pequeña parte. AJAX hace que la información sea cargada de manera precisa, fácil y ágil.

Para dar una referencia a cómo es que funciona AJAX, tomamos como ejemplo *Gmail,* el servicio de correo electrónico de Google, en http://mail.google.com. Este servicio es una interfaz para consulta y administración de correo electrónico desarrollado enteramente usando AJAX. Enviar un nuevo mensaje o borrarlo en este servicio no implica recargar todos los mensajes de la bandeja de entrada una vez terminada la acción. O abrir un antiguo mensaje para recuperar un dato que requerimos, no significará tener que volver a solicitar y descargar toda la interfase del sistema otra vez (imágenes, html, scripts).

Para hacer una analogía rápida, AJAX hace más parecidas las aplicaciones Web a aplicaciones *Stand-alone* o instaladas localmente sin requerimiento de conectividad alguna, pero con todas las ventajas de lo que implica una aplicación Web.

Lo que hace a AJAX tan interesante es que las nuevas aplicaciones y servicios son lanzados con mayor frecuencia, haciendo uso de esta tecnología. Servicios con una agilidad y velocidad notable y mejora en la simplicidad de uso.

3.4 Ventajas y Desventajas de AJAX

AJAX, al igual que muchas otras técnicas de programación de desarrollo de aplicaciones Web aparentemente nuevas, presenta ciertas ventajas y desventajas cuando se implementa, algunas de ellas son descritas a continuación.

Ventajas

- Evita recargar una página Web entera: cuando un usuario realiza algún cambio en la página que tiene en uso, la página realiza pequeñas actualizaciones en ella misma, evitando con esto la recarga de la página completa.

- Las aplicaciones Web pueden ser más rápidas, más interactivas, y más fáciles de usar. Debido a que las peticiones de la página se hacen en "modo *background*", el usuario puede tener interactividad con la página mientras el servidor regresa los datos solicitados.

- Utiliza el objeto *XMLHTTPRequest* para enviar datos a un servidor Web: de esta manera la página o aplicación Web se comunica con el servidor de manera asíncrona o síncrona en "modo *background*".

- Se utiliza el formato XML para recibir los datos del servidor. De esta manera la aplicación manda llamar los datos del servidor y los extrae en una página XML, en la cual los almacena temporalmente y los muestra cada vez

que el navegador lo solicite, sin necesidad de hacer una nueva petición al servidor, haciendo que trabaje de una forma más rápida y eficiente.

Desventajas

- Los links no dan señales inmediatas de que están cargando datos. Generalmente los usuarios, están acostumbrados a que al momento de hacer *click* en alguna de las ligas, se visualice una respuesta de carga de datos. Al emplear AJAX en las aplicaciones, las ligas no responden de esta manera, ya que recargan pequeñas partes de información. Esto se mitiga mediante la presentación temporal de etiquetas o popups indicando al usuario que se está realizando una petición.

- Se rompe el flujo de navegación tradicional (botón "volver" y "actualizar"). Al hacer una transferencia asíncrona después de haber cargado una página, la dirección principal sigue apuntando a la carga original, y por lo tanto los botones atrás y actualizar nos llevarán a la página completa anterior, no a la modificación modular previa.

- Demasiado código AJAX hace lento al navegador. JavaScript trabaja de parte del navegador y para hacer una aplicación completa se requiere mucho código, el uso excesivo de este, hace que ocupe muchos recursos de la computadora del usuario, haciéndola más lenta y poco funcional.

- Las aplicaciones se vuelven completamente dependientes de una conexión. Es probable que en un futuro no se tenga que instalar una aplicación en la computadora, sino que se podrán utilizar vía Internet o Intranet.

3.5 Navegadores que permiten AJAX

Esta es una lista general, y el soporte de las aplicaciones AJAX depende de las características que el navegador permita.

- Navegadores basados en *Gecko*, como *Mozilla*, *Mozilla Firefox*, *SeaMonkey, Camino, Flock, Epiphany, Galeon* y *Netscape* versión 7.1 y superiores.
- *Microsoft Internet Explorer* para Windows versión 5.0 y superiores, y los navegadores basados en él.
- *Opera* versión 8.0 y superiores, incluyendo *Opera Mobile Browser* versión 8.0 y superiores.
- Navegadores basados en Webkit, como Safari y Konqueror.

3.6 Navegadores que no permiten AJAX

- *Opera 7* y anteriores.
- *Microsoft Internet Explorer* para Windows versión 4.0 y anteriores.
- *Microsoft Internet Explorer* para Macintosh, todas las versiones.
- *Dillo*.
- Navegadores basados en texto, como *Lynx* y *Links*.
- Navegadores para incapacitados visuales (*braille*).

3.7 Alternativas a AJAX.

Como ya se mencionó anteriormente, AJAX permite la comunicación asíncrona entre el cliente y el servidor, pero no es la única en hacerlo. Existen varias alternativas que tienen una funcionalidad muy parecida y se mencionan a continuación.

3.7.1 JavaScript Remote Scripting (JSRS)

Esta técnica realiza un intercambio de datos entre la aplicación cliente (la parte visible para el usuario) corriendo en el navegador y la aplicación del servidor sin necesidad de hacer una recarga integral de la pagina para obtener los resultados de importancia. [5]

El envío de la solicitud y respuesta de datos dinámicos se produce cuando un evento DHTML (HTML dinámico) es accionado y se ejecuta una acción sobre algún objeto en la interface cliente. La acción asociada al objeto envía una solicitud HTTP hacia una aplicación de lado del servidor, la cual será capaz de realizar cálculos o consultas a un motor de bases de datos. La aplicación del servidor obtiene los resultados necesarios y envía una respuesta HTTP al cliente, actualizando única y directamente a los objetos que necesitemos modificar, este puede ser un campo texto, una imagen, etc.

Una de las maneras mas fáciles de llevar a cabo este proceso, es utilizando un IFRAME oculto en la interface cliente, el cual trabajará junto a un script en el servidor que se ocupara de las acciones requeridas o solicitadas. Un IFRAME es un elemento HTML que permite insertar o incrustar un documento HTML dentro de un documento HTML principal.

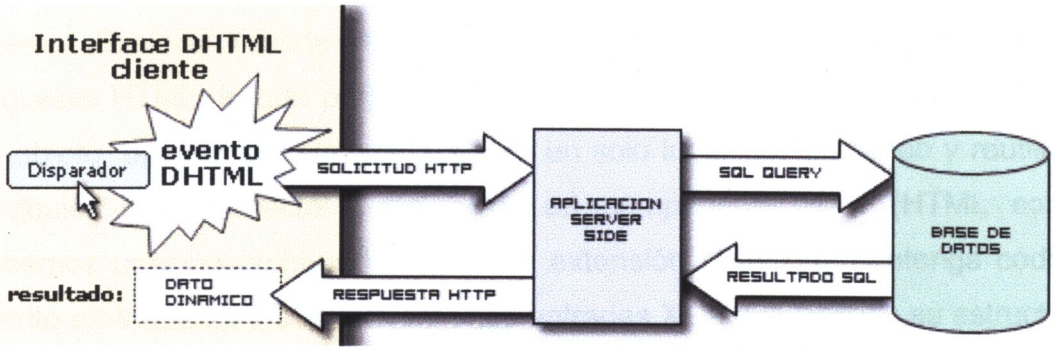

Figura 4.3 JavaScript Remote Scripting

3.7.2 Microsoft's Remote Scripting (MSRS)

Microsoft's Remote Scripting ofrece la opción de realizar llamadas al servidor de manera asíncrona, esto quiere decir que, el *script* del cliente sigue corriendo aun cuando el método del servidor está en ejecución. Las llamadas asíncronas evitan que la interface del usuario de la aplicación sea lenta debido a que el usuario puede seguir trabajando mientras que el *script* del servidor que se abrió inicialmente se esté ejecutando. [6] [7]

Las llamadas remotas asíncronas se realizan de manera muy parecida a las síncronas, para realizar la llamada se necesita:

- Una función de respuesta de JavaScript en el código del cliente es llamada cuando el método del servidor se termine de procesar. Por ejemplo, si el script remoto realiza una búsqueda en una base de datos, la función de respuesta puede regresar el valor de la búsqueda desde la llamada remota y lo despliega en el control de la página
- Puede ser opcional una función JavaScript de error en la respuesta, si al realizar la llamada asíncrona se produce un error.
- Un parámetro de contexto opcional. Estos son los datos que se envían al método y serán pasados posteriormente. Normalmente ayudan a determinar si en la función de respuesta del método se ha hecho la llamada.

3.7.3 Internet Explorer: *Downlad Behavior*

Los *Behaviors* son simples componentes que se utilizan para potenciar y extender el Internet Explorer (desde la versión 5.0) agregando funcionalidad a los distintas etiquetas HTML, lo que proporciona cierto "comportamiento". Al ser definidos como un objeto, pueden ser almacenados en un solo lugar del sitio Web y reutilizados en cualquier otro proyecto. Para crear un componente HTC (HTML component) debemos usar un archivo de texto con extensión ".htc" que contenga código script escrito en VBScript o JScript y algunas entradas XML que definan su estructura.

Los Componentes HTML son útiles en varias situaciones. Una de ellas es cuando se le asocia con un elemento en particular de la página HTML y le permite, de esta forma, proveer manejadores de eventos y disparar eventos personalizados y otras funcionalidades propias de los objetos (como son los métodos y las propiedades). Otro posible uso es acceder al código *script* desde distintos dominios, ya que las propiedades, métodos y eventos expuestos por un *behavior* pueden ser utilizados desde fuera del dominio. Este componente HTML puede ser referido a uno o más objetos HTML y cualquier objeto HTML puede requerir funcionalidad de uno o más *behaviors*.

3.7.4 Webservice Behaviour

Un *WebService Behaviour* permite que los *scripts* que están ejecutándose en un cliente puedan invocar métodos remotos expuestos por servicios Web, u otros servidores Web, que soporten SOAP y WSDL 1.1. Este *behaviour* permite que los desarrolladores de páginas Web utilicen SOAP sin que se tenga un conocimiento profundo de dicho protocolo. [8]

El *behaviour* soporta el uso de una amplia variedad de tipos de datos, incluyendo tipos de datos SOAP intrínsecos, vectores, objetos, y datos XML. El *behaviour* se implementa a través de un fichero HTC (*HTML component*), y es soportado por Microsoft Internet Explorer 5 y versiones posteriores.

Aun cuando los *behaviours* utilizan el protocolo SOAP para comunicarse con los servicios Web, su propósito es proporcionar una manera simple de aprovechar este protocolo sin requerir un vasto conocimiento del mismo.

Una de las ventajas de los *behaviours* radica en que no es necesario modificar el código de los servicios Web existentes, sólo se deben agregar unas líneas de código JavaScript a la página Web desde donde se pretende acceder a dicho servicio. El servicio Web debe ser accesible desde el servidor que almacena la página Web. Para iniciar una llamada remota a través del *behaviour*, sólo es necesario conocer los métodos y los parámetros requeridos del servicio que se desea invocar, los que se obtienen del fichero WSDL de dicho servicio.

3.7.5 XML-RPC

Es un protocolo de llamada a procedimiento remoto que usa *XML* para codificar las llamadas y *HTTP* como mecanismo de transporte.

Es un protocolo muy simple ya que sólo define unos cuantos tipos de datos y comandos útiles, además de una descripción completa de corta extensión. La simplicidad del XML-RPC está en contraste con la mayoría de protocolos RPC que tiene una documentación extensa y requiere considerable soporte de software para su uso.

Fue creado por Dave Winer de la empresa UserLand Software en asociación con Microsoft en el año 1995. Al considerar Microsoft que era muy simple y adicionar funcionalidades y después de varias etapas de desarrollo el estándar dejó de ser sencillo y se convirtió en lo que actualmente se conoce como SOAP.

3.7.6 RSLite

Es una puesta en práctica extremadamente ligera de Scripting remoto que utiliza cookies. Es compatible con navegadores Opera, pero limitado para escoger llamadas y cantidades pequeñas de datos.

3.7.7 Flash

Flash aunque no propiamente es una alternativa a AJAX, también cumple con una característica similar, gracias a una función llamada LoadVars. LoadVars permite enviar datos al servidor a través de una conexión tal como lo hace un formulario HTML además de poder ser procesado por cualquier *Script* del lado del servidor (ASP, PHP, JSP, Perl, CGI, etc).

3.8 FRAMEWORKS Y API's

Para trabajar de un forma más sencilla con AJAX, se puede hacer a través de archivos de código y/o programas previamente desarrollados que se utilizan para facilitar la programación de los *scripts* conocidos como frameworks y API's, básicamente estos archivos se usan como bloques de código que se reutilizan para ahorrar tiempo y trabajo cada que sea necesario ocuparlos.

Los frameworks son archivos o programas utilizados por desarrolladores de software que contienen bibliotecas, funciones y/o rutinas de código que facilitan y agilizan el proceso de desarrollo de aplicaciones.

Estas "bibliotecas" funcionan con una sintaxis establecida por sus desarrolladores, las sintaxis son fáciles de comprender y a su vez, fáciles de manejar, logrando con esto que sea más sencillo manejar la programación de cualquier tecnología.

Las principales ventajas de la utilización de un framework son:

1. El desarrollo rápido de aplicaciones. Los componentes incluidos en un framework ayudan al programador en la escritura de código.

2. La reutilización de componentes que se utilizan con mayor frecuencia. Los frameworks son los paradigmas de programación que el desarrollador usa con mucha frecuencia y le ahorra líneas de código.

3. El uso y la programación de componentes que siguen una estructura de diseño uniforme. Un framework orientado a objetos logra que los componentes sean clases que pertenezcan a una gran jerarquía de clases, por lo que resultan bibliotecas más fáciles de aprender a usar.

Las desventajas de los frameworks son:

1. La dependencia del código fuente de una aplicación con respecto al framework. Si se desea cambiar de framework, la mayor parte del código debe reescribirse.

2. La demanda de grandes cantidades de recursos computacionales debido a que la característica de reutilización de los frameworks tiende a generalizar la funcionalidad de los componentes. El resultado es que se incluyen características que están "de más", provocando una sobrecarga de recursos que se hace más grande en cuanto más amplio es el campo de reutilización.

AJAX también cuenta con este tipo de utilerías para simplificar su uso y demostrar sus capacidades dentro de su entorno.

Existe una gran cantidad de frameworks de AJAX, cada uno de ellos fue desarrollado con un fin y para cumplir una tarea especifica, por ejemplo, algunas de las tareas más utilizadas en el desarrollo de páginas Web, son la introducción de efectos para dar una apariencia más dinámica, agradable, amistosa y funcional, otros tienen la finalidad de disminuir las tareas complejas que se utilizan como herramientas para sistemas de información Web, los cuales por medio de funciones y clases contenidas dentro de estos, ahorran tiempo de programación.

Por otro lado, la diferente variedad de frameworks lleva al desarrollador a la búsqueda de la librería para cumplir su objetivo, como pueden ser frameworks para

desarrollo de aplicaciones RIA, herramientas para ahorro de código JavaScript, desarrollo de código extensible, etc.

A través de una conocida página de Internet [17] de AJAX, se obtuvieron los resultados de una encuesta realizada en el 2006 para dicha comunidad, en la que dan a conocer cuáles son los Frameworks más utilizados y qué plataformas de programación son las más utilizadas.

Tabla 4.1 Plataformas de programación más utilizadas.

Lenguaje de Programación	Repuesta (%)	Conteo de Votos
ColdFusion	5.2%	45
Java (J2EE, JSP, JSF, Tomcat, etc.)	36.8%	318
Microsoft.NET (C#, ASP.NET, VB.NET)	15.7%	136
PHP	**49.5%**	428
Perl	5.0%	43
Python	5.7%	49
Ruby on Rails	13.8%	119

Tabla 4.2 Ajax frameworks, toolkits, o librerias JavaScript más utilizados.

Frameworks más utilizados	Repuesta (%)	Conteo de Votos
Uso del objeto XMLHTTPRequest sin otro tipo de herramientas.	25.2%	218
Atlas, Microsoft	4.4%	38
Dojo	18.7%	162
DWR	11.6%	100
jQuery	7.2%	62
JSON	11.0%	95
Moo.fx	11.0%	95

Prototype	**43.1%**	373
Rico	4.7%	41
Ruby on Rails	7.9%	68
Script.aculo.us	33.0%	285

Este es el punto de partida para hablar sobre algunos de los frameworks mas populares o mas usados, aclarando que estan hechos con distintos fines.

3.8.1 Prototype

Prototype es un framework de AJAX que ha ido ganado mucha popularidad entre los desarrolladores, debido a que es una compleja herramienta diseñada para facilitar el desarrollo de aplicaciones en JavaScript complejas.

Su principal función es permitir al desarrollador lograr una conexión a través de AJAX de forma sincrónica y asincrónica.

Ventajas

- Comunicación síncrona y asíncrona.
- Ofrece una interfaz limpia gracias a la actualización de los elementos HTML.
- Respuesta a todas las peticiones con facilidad.
- Facilidad y practicidad en su manejo.

Desventajas

- Al hacer el llamado a Prototype, carga todas las funciones y objetos, haciéndolo muy pesado en su carga.

3.8.2 Script.aculo.us

Script.aculo.us es una librería de AJAX orientada al área de efectos visuales, como lo es el manejo del famoso "arrastrar y soltar" (drag and drop), herramientas de autocompletado, y otros efectos avanzados.

Ventajas

- Código completamente reutilizable.
- Soporta completamente CSS.
- API con interfaz de reproducción de sonidos MP3.

Desventajas

- Debido a que es una simple librería, Script.aculo.us, necesita de algún otro framework, para ser funcional ya que como se dijo anteriormente, simplemente ayuda a crear efectos visuales y no aplicaciones.
- Escases de información ya que la mayoría de sus aplicaciones son descargables.

3.8.3 DOJO

Dojo está enfocado a asuntos de usabilidad comunes como pueden ser la navegación y detección del navegador, soportar cambios de URL en la barra de navegación para luego regresar a ellas (bookmarking), y la habilidad de degradar cuando AJAX/JavaScript no es completamente soportado en el cliente.

Ventajas

- Ahorra tiempo de desarrollo y diseño de interfaces complejos.
- Buen manejo de eventos complejos.
- Promueve la elaboración de pruebas unitarias y depuración.

Desventajas

- Depende de otros elementos avanzados para el desarrollo del contenido Web.
- Incompatibilidad con algunos navegadores.

3.8.4 JQuery

Son bibliotecas de Javascript que permite simplificar la manera de interactuar con documentos HTML, permitiendo manejar eventos, desarrollar animaciones, y agregar interacción con la tecnología AJAX a las páginas Web. jQuery está diseñado para cambiar la forma de escribir código JavaScript.

Ventajas

- Soporta gran cantidad de selectores de CSS, logrando con ellos la manipulación de estilos.
- Añade y remueve con gran facilidad atributos en las etiquetas de HTML.
- Facilita mucho mas el uso de AJAX como herramienta de desarrollo.
- Compatibilidad con otras librerías.

Desventajas

- La mayoría de las ocasiones donde se usa esta biblioteca tiene que ir acompañada de alguna otra para que cumpla su función.

3.8.5 Ruby On Rails

Es un framework para el desarrollo de aplicaciones Web basadas en un lenguaje de scripts, multiplataforma, netamente orientado a objetos llamado *Ruby*, ideal para manejo de bases de datos con el patrón arquitectónico Model-View-Control.

Ventajas

- Fácil de aprender y utilizar.
- Es un framework muy poderoso ya que está basado en un lenguaje orientado a objetos.

Desventaja

- Falta de documentación por ser un framework relativamente nuevo.
- Problemas al momento de implementar una aplicación con varios usuarios.

Los Frameworks son herramientas que utilizan los programadores de aplicaciones, que les permiten explotar un poco más los recursos de los lenguajes de programación, estas herramientas ayudan entre otras cosas, a reutilizar líneas de código que se usan con mayor frecuencia, rutinas capaces de manejar información logrando con esto la reducción de tiempos.

Es importante mencionar que cada uno de los frameworks que existen tienen un punto fuerte, es decir, están destinados a desempeñarse en un área específica, animación, manejo de bases de datos, manejos de estilos y efectos, entre otros, por ello, se debe tener cuidado y conocimiento de sus funciones para saber cual y cuando se puede utilizar dependiendo la función que vaya a realizar la aplicación que se quiera hacer. A demás, tienen la ventaja de poderse utilizar varios a la vez y lograr mayores resultados funcionales y agradables a la vista.

Capítulo 5 TECNOLOGÍAS

4.0 Tecnologías usadas por AJAX

AJAX está formado por diversas tecnologías que contribuyen a su funcionamiento aclarando que no es un lenguaje de programación, sino un paradigma que combina varias tecnologías Web logrando resultados sorprendentes, de gran utilidad, que dejan de lado los métodos tradicionales en la forma de desarrollar aplicaciones Web.

Es por ello que se da una breve introducción a cada uno de sus elementos, de manera fácil y práctica para conocerlos de manera superficial.

4.1 JavaScript

JavaScript se inicia en al año de 1995 cuando Netscape introduce la versión 2.0 de su navegador e incluye *JavaScript* bajo el nombre de *"Mocha"*, cuando aparece esta versión de navegador se le llamaba *"LiveScript"*. Finalmente se le bautiza con el nombre de *JavaScript* para llamar la atención de los medios y la industria de la informática, ya que por otro lado Java iniciaba el lanzamiento de su lenguaje de programación basado en *applets*.

Netscape quería que *JavaScript* fuera un lenguaje de guiones, fácil de usar y que cualquier persona pudiera utilizarlo. Después de 2 años *JavaScript* se convirtió en una de las herramientas más utilizadas por los desarrolladores Web, incluso se utiliza más que el propio Java y ActiveX.

En el año de 1996 Microsoft empieza a tener gran interés por lograr competir con *JavaScript* por lo que lanza su lenguaje llamado *Jscript*, pero no logra tener gran éxito por lo que no ha podido competir directamente con este lenguaje.

A mediados de 1997, Netscape promocionó *JavaScript* y lanzó la versión 1.2 de este lenguaje. Esta nueva versión incluye nuevos componentes que dan gran potencial al

lenguaje, pero lamentablemente esta versión sólo funcionaba en la ultima versión del *Navigator* en ese momento. En la actualización, esta versión de *JavaScript* es soportada en la mayoría de los navegadores y es utilizado en casi todo los sitios de Internet existentes.

La característica principal de *JavaScript*, es la de ser un lenguaje de *scripting*, pero principalmente y sobre todo, la de ser el lenguaje *scripting* más usado por los programadores Web, utilizándose casi al 90% de los desarrollos Web.

JavaScript es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez y con gran ligereza.

JavaScript es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias *JavaScript* contenidas en una página HTML y ejecutarlas adecuadamente.

JavaScript es un lenguaje orientado a eventos, es decir, cuando un usuario hace *click* sobre un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante *JavaScript* se pueden desarrollar *scripts* que ejecuten acciones en respuesta a estos eventos.

Está orientado a objetos. El modelo de objetos de *JavaScript* está reducido y simplificado, pero incluye los elementos necesarios para que los *scripts* puedan acceder a la información de una página y puedan actuar sobre la *interfaz* del navegador.

JavaScript contiene una serie de limitaciones, cuyo objetivo principal es prevenir que se altere el sistema archivos del cliente, por lo que no puede leer, escribir, crear, borrar o listar ficheros con excepción de las cookies. Carece además de "primitivas de red", de manera que no puede establecer conexión directa con otras máquinas.

Cabe aclarar que *JavaScript* no tiene nada que ver con JAVA, son lenguajes totalmente diferentes.

Tabla 5.1 Diferencias entre JAVA y JavaScript.

JAVASCRIPT	JAVA
Interpretado por el cliente	Compilado (bytecodes). Se descarga del servidor y se ejecuta en el cliente
Basado en objetos. Usan objetos, pero no clases ni herencia.	Programación orientada a objetos. Los applets constan de clases objeto con herencia.
El código se integra e incrusta en documentos HTML	Se utilizan APPLETS. Se accede a ellos desde documentos.
Los tipos de datos de las variables no se declaran	Es necesario definir los tipos de datos de las variables
No se puede escribir automáticamente en el disco duro	No puede escribir automáticamente en el disco duro

Con *JavaScript* se pueden crear efectos especiales en las páginas y definir interactividades con el usuario.

Entre las acciones en las que mayormente se emplea *JavaScript* se encuentra:
Por un lado los efectos especiales sobre páginas Web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, *JavaScript* permite comprobar la validez de la entrada de formularios, abrir y cerrar ventanas, ejecutar instrucciones como respuesta a las acciones del usuario, además se pueden crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo.

Existen dos formas de introducir un script de *JavaScript* en una página HTML:

1. Incrustado en el código HTML, entre las etiquetas o *tags* `<script>` y `</script>`. El código JavaScript se coloca entre las marcas de comentario HTML `<!--` y `-->`, para que los navegadores antiguos (que no soportan las etiquetas *script*) no muestren el código fuente en la página.

Ejemplo de código *JavaScript* incrustado en el código HTML de una página

```
<HTML>
<HEAD>
<TITLE>Introducción a JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
function saludo() {
window.alert('¡Bienvenido a JavaScript!')
}
//-->
</SCRIPT>
</HEAD>
<BODY onLoad="saludo()">
</BODY>
</HTML>
```

2. Como archivo ".js" que se carga con la página HTML. Para ello, debe indicarse en las etiquetas o *tags* el nombre y ubicación del archivo ".js" que contiene el *script JavaScript*, como en este ejemplo:

```
<HTML>
<HEAD>
<TITLE>Tutorial de JavaScript</TITLE>
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript"
SRC="codigo.js"></SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

4.2 DOM (Document Object Model)

El DOM (Document Object Model) fue introducido por primera vez en 1995 por Netscape, implementándose en la versión 2 de su navegador. El primer modelo de DOM era, sustancialmente, una jerarquía de instancias de objetos *JavaScript*, mediante la cual los elementos de la página estaban en relación descendente entre sí.

En la versión 3.0, Netscape extendió el DOM a través de la introducción de otros objetos para representar: *layer*, *marcos*, *plug-in*, *applet*, imágenes, *enlaces* y *anclajes*, sin modificar mucho su estructura original.

Posteriormente, *Microsoft* introdujo su propio lenguaje de scripting (*Visual Basic Scripting Edition*, *VBScript*) en la versión 3 de Internet Explorer, y el DOM adoptado era similar al del adversario Netscape.

Con la cuarta versión del Explorer, Microsoft extendió el DOM a todos los contenidos y marcas de página, y se aproximó a las recomendaciones actuales del W3C.

El DOM implementado en MSIE 4 era más completo que el propuesto por Netscape. Ese nuevo DOM de Microsoft, era accesible con *JavaScript* y *VBScript* el cual incluía el objeto "all", que representaba el conjunto de todos los elementos de una página. Netscape modificó su navegador hasta dejarlo a punto, pero aún así, los eventos quedaban limitados a objetos específicos, mientras que Microsoft abría los elementos de una página a todos los eventos, lo que permitió el acceso a todos los atributos de un elemento, incluidos color, fondo y fuente.

El Modelo de Objetos del Documento (DOM) es una interfaz de programación de aplicaciones (API) para documentos HTML y XML. Define la estructura lógica de los documentos y el modo en que se accede y manipula un documento. En las especificaciones del DOM, el término "documento" se utiliza en un sentido amplio.

Siendo una especificación del W3C, uno de los objetivos importantes del Modelo de Objetos del Documento es proporcionar un interfaz estándar de programación que pueda utilizarse en una amplia variedad de entornos y aplicaciones. El DOM se ha diseñado para ser utilizado en cualquier lenguaje de programación.

En el DOM, los documentos tienen una estructura lógica que es muy parecida a un árbol, como se muestra en la figura 4.1.

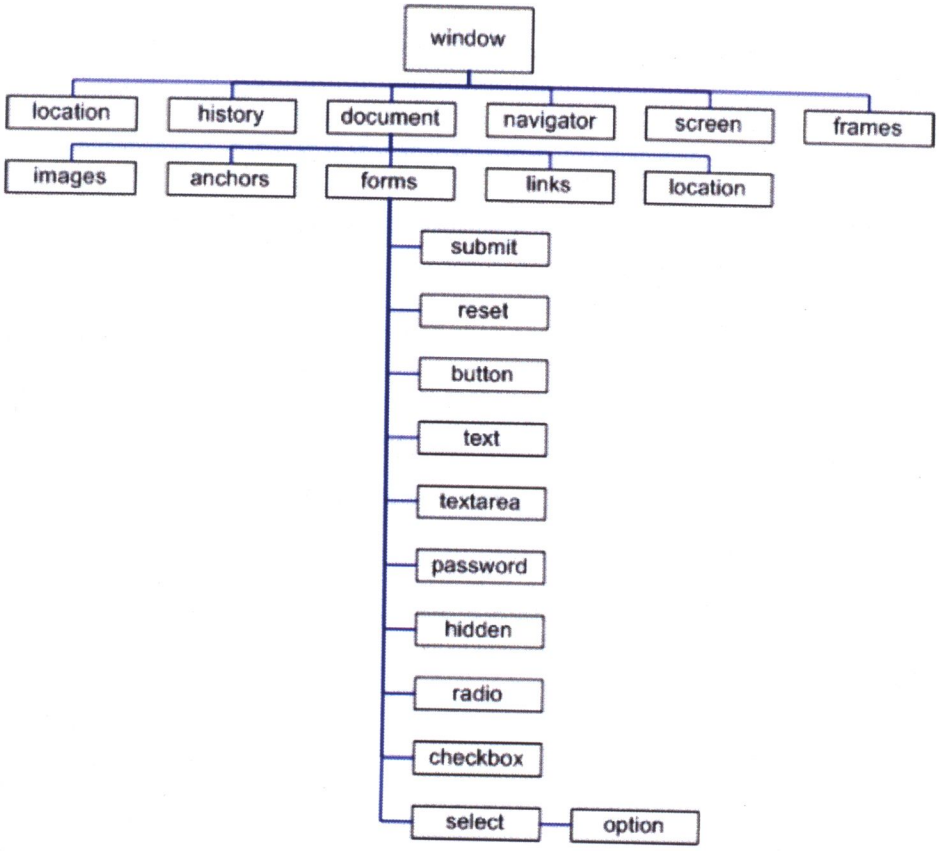

Figura 5.1 Estructura lógica del DOM.

Sin embargo, el DOM no especifica que los documentos deban ser implementados como un árbol, ni tampoco especifica cómo deben implementarse las relaciones entre objetos. El DOM es un modelo lógico que puede implementarse de cualquier manera que sea conveniente. Una propiedad importante de los modelos de estructura o árboles del DOM es su *isomorfismo estructural*: si dos implementaciones cualesquiera del Modelo de Objetos del Documento se usan para crear una representación del mismo documento, ambas crearán el mismo modelo de estructura, con exactamente los mismos objetos y relaciones.

El nombre "Modelo de Objetos del Documento" se debe a que es un "modelo de objetos" en el sentido tradicional del diseño orientado a objetos: los documentos se

modelizan usando objetos, y el modelo comprende no solamente la estructura de un documento, sino también el comportamiento de un documento y de los objetos de los cuales se compone. Como modelo de objetos, el DOM identifica:

- las interfaces y objetos usados para representar y manipular un documento
- la semántica de estas interfaces y objetos, incluyendo comportamiento y atributos
- las relaciones y colaboraciones entre estas interfaces y objetos

El Modelo de Objetos del Documento consiste actualmente de dos partes, el Núcleo del DOM y el DOM HTML. El Núcleo del DOM representa la funcionalidad usada para los documentos XML, y también sirve de base para el DOM HTML. Una implementación conforme del DOM debe implementar todas las interfaces fundamentales sobre el Núcleo con la semántica definida. Además, debe implementar o bien el DOM HTML o bien las interfaces extendidas (XML), o ambas, con la semántica definida.

Con el Modelo de Objetos del Documento se pueden construir documentos, navegar por su estructura, y añadir, modificar o eliminar elementos y contenido. Se puede acceder a cualquier cosa que se encuentre en un documento HTML o XML, y se puede modificar, eliminar o añadir usando el Modelo de Objetos del Documento, salvo algunas excepciones. En particular, aún no se han especificado las interfaces DOM para los subconjuntos internos y externos de XML.

4.2.1 La Raíz (Root) Document

El objeto `document` sirve de raíz de un árbol de nodos. Implementa también la interfaz `Node`. Este árbol va a tener nodos hijos pero no tendrá nodo padre ni nodos de su mismo nivel, dado que él es el nodo inicial. Además de ser un `Node`, también implementa la interfaz `Document`.

Esta interfaz proporciona métodos para acceder y crear otros nodos en el árbol del documento. Algunos métodos son:

```
getElementById()
getElementsByTagName()
createElement()
createAttribute()
createTextNode()
```

Cabe destacar, que de modo distinto a otros nodos, sólo hay un objeto `document` en una página. Todos los métodos anteriores (excepto `getElementsByTagName()`) pueden utilizarse sólo con el objeto `document`, usando la sintaxis `document.methodName()`.

El objeto `document` también puede tener otras propiedades relacionadas con el soporte de antiguos niveles del DOM.

Estas propiedades se mantienen con la intención de proporcionar cierta compatibilidad hacia atrás para que páginas diseñadas para navegadores antiguos sigan funcionando apropiadamente con las nuevas versiones. Se pueden utilizar todavía en los scripts pero probablemente no estén soportadas en versiones futuras.
Recorriendo el árbol del Documento.

El árbol del documento refleja la estructura del código de una página. Cada etiqueta o par de etiquetas está representada por un nodo o elemento con otros nodos representando atributos.

El objeto `document` tiene sólo un elemento hijo, dado por `document.documentElement`. Para páginas Web, éste representa la etiqueta exterior HTML, y ésta actúa como el elemento raíz del árbol del documento, por lo tanto, los elementos hijos HEAD y BODY, que tendrán a su vez otros elementos hijos.

Los métodos de la interfaz `Node`, pueden recorrer el árbol del documento para acceder a los nodos individuales contenidos en dicho árbol. Tomando en cuenta lo siguiente:

```
<html>
<head>
<title></title>
</head>
<body><p>Ejemplo de un párrafo.</p></body>
</html>
```

Y éste código:

```
alert(document.documentElement.lastChild.firstChild.tagName);
```

que debería mostrar la etiqueta <p>, el nombre de la etiqueta representada por dicho nodo. El código se puede descomponer como sigue:

- `document.documentElement` - devuelve la etiqueta HTML de la página.
- `.lastChild` - devuelve la etiqueta BODY.
- `.firstChild` - devuelve el primer elemento en BODY.
- `.tagName` - devuelve el nombre de la etiqueta de dicho elemento, "P" en este caso.

Existen varios problemas obvios para acceder a los nodos de esta forma. Por ejemplo, un simple cambio del código de la página, como agregar texto o elementos con formato o imágenes, cambiará la estructura de árbol. La ruta utilizada anteriormente no apuntará, en ese caso, al nodo intencionado.

Además de otras que no son muy obvias, como ciertas incompatibilidades entre los navegadores. Nótese que en el ejemplo HTML anterior, no hay espacio entre las etiquetas BODY y P. Si se hubiese añadido simplemente un salto de línea,

```
<html>
<head>
<title></title>
</head>
<body>
<p>Ejemplo de un párrafo.</p>
</body>
</html>
```

Netscape añadiría un nodo para este dato, mientras que IE no lo haría. Por esto, en Netscape, el código JavaScript mostrado anteriormente debería devolver "undefined" como su nuevo puntero para el nodo de texto para este espacio en blanco. Como no es un nodo elemento, no tiene la propiedad tagName. IE, por lo tanto, no agrega nodos para espacios en blanco cómo estos, por lo que todavía apuntaría a la etiqueta P.

4.2.2 Accediendo a los Elementos Directamente

Aquí es donde el método `document.getElementById()` adquiere utilidad. Mediante la adición de un atributo ID a la etiqueta del párrafo (o cualquier otra etiqueta, para el caso), se puede hacer referencia a ella directamente.

```
<p id="miParrafo">Ejemplo de un párrafo.</p>...
alert(document.getElementById("miParrafo").tagName);
```

De este modo, se pueden evitar problemas de compatibilidad y actualizar los contenidos de la página sin tener que preocuparse de dónde está el nodo del párrafo en el árbol del documento. Debemos recordar que cada ID tiene que ser único en una página.

Un método algo menos directo de acceder a los nodos de tipo elemento es proporcionado por `document.getElementsByTagName()`. Éste devuelve un array de nodos representando todos los elementos en una página que tengan la etiqueta HTML especificada. Por ejemplo, cambiar el color de todos los links en una página.

```
var nodeList = document.getElementsByTagName("A");
for (var i = 0; i < nodeList.length; i++)
  nodeList[i].style.color = "#ff0000";
```

4.3 CSS (Cascading Style Sheets)

En el año 1998, Netscape e Internet Explorer participaban en la comúnmente conocida "guerra de los navegadores", los intereses comerciales de las dos compañías luchaban por el mercado de usuarios de Internet.

Netscape 4 e Internet Explorer 4 incorporaron soporte para una tecnología que les permitiera dar estilos a los diseños Web, pero dejaba mucho que desear, especialmente en Netscape 4, ya que no era posible su incorporación salvo en un muy mínimo número de características, puesto que la manera de tratar los estándares era muy diferente y por eso ni aún así se podía asegurar una visualización correcta, mientras tanto, Internet Explorer 4 tenía un mejor manejo de los estilos, porque, permitía parcialmente el uso de un mayor número de características. Esta tecnología fue denominada ***CSS*** (Cascading Style Sheets).

Las Hojas de Estilo en Cascada o CSS, son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de las páginas Web.

Existen varias versiones de CSS:

CSS1: fue la primera especificación que desarrolló el World Wide Web Consortium (W3C). Las hojas de estilo CSS1 describen el formato del texto y de los componentes de una página (fuente, color, tamaño, etc.).

CSS2: permite ubicar elementos de XHTML en diferentes capas o layers, cuyo posicionamiento no tiene que seguir el flujo HTML. Plantea el modelo de cajas donde

cada bloque se define como una caja que se coloca en un lugar concreto. CSS2 soporta su versión anterior CSS1 con algunas excepciones.

En la actualidad, la World Wide Web Consortium (W3C) trabaja desarrollando una nueva versión, la **CSS3**, pero los navegadores todavía no son capaces de entenderlo correctamente. [9]

CSS ya no es una novedad y es posible utilizar ciertas instrucciones más amplias, como el control de otras características gráficas tales como imágenes y colores de fondo, márgenes exactos y bordes, para evitar el laborioso y a veces poco gratificante diseño de tablas complejas para un *"layout"*, que incluye frecuentemente tablas anidadas y complicados algoritmos de combinación de celdas, características que hacen al archivo muy pesado para descargar, porque inundan el código con la extensa serie de etiquetas requeridas.

CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

Funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos. Las hojas de estilo están compuestas por una o más de una regla aplicadas a un documento HTML o XML. La regla tiene dos partes: un selector y la declaración. A su vez la declaración está compuesta por una propiedad y el valor que se le asigne.


```
h1 {color: red;}
```
`h1` es el selector
`{color: red;}` es la declaración

El selector funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración. La declaración es la parte de la regla que establece cuál será el efecto. En el ejemplo anterior, el selector `h1` indica que todos los elementos `h1` se verán afectados por la declaración donde se establece que la propiedad `color` va a tener el valor `red` (rojo) para todos los elementos `h1` del documento o documentos que estén vinculados a esa hoja de estilos.

Las tres formas más conocidas de dar estilo a un documento son las siguientes:

- Utilizando una hoja de estilo externa que estará vinculada a un documento a través del elemento `<link>`, el cual debe ir situado en la sección `<head>`.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN">
<html>
  <head>
    <title>Título</title>
    <link rel="stylesheet" type="text/css"
      href="http://www.w3.org/css/officeFloats.css" />
  </head>
  <body>
  </body>
</html>
```

- Utilizando el elemento `<style>`, en el interior del documento al que se le quiere dar estilo, y que generalmente se situaría en la sección `<head>`. De esta forma los estilos serán reconocidos antes de que la página se cargue por completo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN">
<html>
  <head>
    <title>hoja de estilo interna</title>
    <style type="text/css">

      body {
```

```
        padding-left: 11em;
        font-family: Georgia, "Times New Roman", serif;
        color: red;
        background-color: #d8da3d;
      }

      h1 {
        font-family: Helvetica, Geneva, Arial, sans-serif;
      }

    </style>
  </head>
  <body>
    <h1>Aquí se aplicará el estilo de letra para el
Título</h1>
  </body>
</html>
```

- Utilizando estilos directamente sobre aquellos elementos que lo permiten a través del atributo `<style>` dentro de `<body>`. Pero este tipo de estilo pierde las ventajas que ofrecen las hojas de estilo al mezclarse el contenido con la presentación.

Algunas normas básicas a la hora de crear una CSS son las siguientes:

- En el siguiente ejemplo, `h1{color: red;}`, el *selector*, `<h1>`, le dice al navegador la parte del documento que se verá afectada por esa regla. Los selectores pueden aparecer individualmente o agrupados, separándolos con comas:

```
h1, h2, h3 {
  color: red;
}
```

o lo que es lo mismo

```
h1 {color: red;}
h2 {color: red;}
h3 {color: red;}
```

- La *propiedad*, que en este caso sería `color`, especifica qué aspecto se va a cambiar. En este ejemplo la propiedad cambiada será el color. Las propiedades que se desean modificar en una CSS para un mismo selector pueden agruparse, pero será necesario separar cada una de ellas con un punto y coma.

```
p {text-align:center;color:red}
```

Normalmente se describe una propiedad por línea, de la siguiente manera:

```
h1 {
  padding-left: 11em;
  font-family: Georgia, Times New Roman,Times, serif;
  color: red;
  background-color: #d8da3d;
}
```

- El *valor*, en este caso `red`, establece el valor de la propiedad. Es importante recordar que si el valor está formado por más de una palabra, hay que ponerlo entre comillas.

```
p {font-family: "sans serif";}
```

4.4 XML (Extended Markup Language)

El XML proviene de un lenguaje que inventó IBM por los años 60's. Este lenguaje surgió gracias a la necesidad que tenía la empresa para resolver sus problemas asociados al tratamiento de documentos en diferentes plataformas.

El principal problema era que cada aplicación utilizaba sus propias marcas para describir los diferentes elementos. Las *marcas* son códigos que indican a un programa cómo debe tratar su contenido y así, si se desea que un texto aparezca con un formato determinado, dicho texto debe ir delimitado por la correspondiente

marca que indique como debe ser mostrado en pantalla o impreso. Y lo mismo ocurre con todas las demás características de cualquier texto.

Conociendo este sistema y conociendo a la perfección el sistema de marcas de cada aplicación, sería posible pasar información de un sistema a otro sin necesidad de perder el formato indicado. La forma que IBM creó para solventar esto se basaba en tratar las marcas como texto accesible desde cualquier sistema, texto plano, código ASCII. Y la norma se denominó GML (General Modeling Language).

Más tarde GML pasó a manos de ISO y se convirtió en SGML (ISO 8879), Standart Generalized Markup Language. Esta norma es la que se aplica desde entonces a todos los lenguajes de marcas, cuyos ejemplos más conocidos son el HTML y el RTF.

XML como tal, comenzó en 1996 y la primera versión salió a la luz el 10 de febrero de 1998. La primera definición que apareció fue: *Sistema para definir validar y compartir formatos de documentos en la Web.*

Durante el año 1998 XML tuvo un crecimiento exponencial, y con ello sus apariciones en medios de comunicación, menciones en páginas Web, soporte software, etc.
Los lenguajes de marcas no son equivalentes a los lenguajes de programación aunque se definan igualmente como "lenguajes". Son sistemas complejos de descripción de información, normalmente documentos, que si se ajustan a SGML, se pueden controlar desde cualquier editor ASCII. Las marcas más utilizadas suelen describirse por textos descriptivos encerrados entre signos de "menor que" (<) y "mayor que" (>), siendo lo más usual que existan una marca de principio y otra de final.

Existen tres utilizaciones básicas de los lenguajes de marcas:

1. los que sirven principalmente para describir su contenido,
2. los que sirven más que nada para definir su formato y
3. los que realizan las dos funciones indistintamente.

Las aplicaciones de bases de datos son buenas referencias del primer sistema, los programas de tratamiento de textos son ejemplos típicos del segundo tipo, y aunque no lo parezca, el HTML es la muestra más conocida del tercer modelo.

XML (Extensible Markup Language) es un Lenguaje de Etiquetado Extensible muy simple, pero estricto, que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML.

XML es un formato que permite la lectura de datos a través de diferentes aplicaciones y ofrece grandes ventajas en su utilización.

- XML es un conjunto de reglas usadas para definir etiquetas semánticas que organizan un documento en diferentes partes.
- No necesita actualizaciones de versiones para que pueda funcionar en futuros navegadores. Los identificadores pueden crearse de manera simple y ser adaptados en el acto en internet/intranet por medio de un validador de documentos (parser).
- Se caracteriza por ser consistente, homogéneo y tener gran variedad de identificadores descriptivos en comparación a los atributos de la etiqueta <META> del HTML.
- La integración de los datos en un archivo XML, no impide el intercambio de documentos entre aplicaciones.
- La extensibilidad y flexibilidad de este lenguaje permite agrupar una variedad amplia de aplicaciones, desde páginas web hasta bases de datos.
- Permite gestionar y manipular los datos desde el propio cliente web.

- Su comportamiento es más estable y actualizable con las aplicaciones Web basadas en el desarrollo tradicional con bases de datos, incluyendo enlaces bidireccionales y almacenados de forma externa.
- Integración de datos de un documento XML en cualquier formato deseado de manera directa.

Un documento XML tiene dos estructuras, una lógica y otra física.

Físicamente, el documento está compuesto por unidades llamadas entidades. Una entidad puede hacer referencia a otra entidad, causando que esta se incluya en el documento. Cada documento comienza con una entidad documento, también llamada raíz.

Lógicamente, el documento está compuesto de declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento, todos los cuales están indicados por una marca explícita. Las estructuras lógica y física deben encajar de manera adecuada:

Los documentos XML se dividen en dos grupos, documentos bien formados y documentos válidos.

Bien formados: Son todos los que cumplen las especificaciones del lenguaje respecto a las reglas sintácticas sin estar sujetos a unos elementos fijados en un DTD (Definición de Tipo de Documento). De hecho los documentos XML deben tener una estructura jerárquica muy estricta y los documentos bien formados deben cumplirla.

Válidos: Además de estar bien formados, siguen una estructura y una semántica determinada por un DTD: sus elementos y sobre todo la estructura jerárquica que define el DTD, además de los atributos, deben ajustarse a lo que el DTD dicte.

Ejemplo:

```
<?xml version=" 1.0 " encoding=" UTF-8 " standalone= " yes "?>
<ficha>
<nombre> Juan </nombre>
<apellido> Pérez </apellido>
<direccion> Av. Siempre Viva 380 </direccion>
</ficha>
```

Todos los documentos XML deben empezar con una línea de encabezado la cual indica que lo que la sigue es XML. Aunque puede ser opcional, es recomendable incluirla.

Esta línea contiene varios atributos, algunos obligatorios y otros no:

Version: Indica la versión de XML usada en el documento. Es obligatorio ponerlo, a no ser que sea un documento externo a otro que ya lo incluía.

Encoding: La forma en que se ha codificado el documento. Por defecto es UTF-8, aunque podrían ponerse otras, como UTF-16, US-ASCII, ISO-8859-1, etc. No es obligatorio salvo que sea un documento externo a otro principal.

Standalone: Indica que el documento va acompañado de un DTD o no, en principio no hay porqué ponerlo, porque luego se indica el DTD si se necesita.

La declaración de tipo de documento, define qué tipo de documento se esta creado para poder ser procesado correctamente.

En la declaración se define el tipo de documento, y dónde encontrar la información sobre su Definición de Tipo de Documento, mediante un identificador público (PUBLIC): que hace referencia a dicha DTD o identificador universal de recursos (URI): precedido de la palabra SYSTEM.

Ejemplos:

```
<!DOCTYPE MESAJE SYSTEM "mesaje.dtd">
<!DOCTYPE HTML PUBLIC "-/ /W3C/ /DTD HTML 3.2 Final/ /EN">
<!DOCTYPE LABEL SYSTEM
"http://www.mipagina.com/dtds/label.dtd">
```

4.4.1 Documentos XML bien formados

Un documento XML se dice que está bien formado si encaja con las especificaciones XML de producción, lo que implica:

Estructura jerárquica de elementos

Los documentos XML deben seguir una estructura estrictamente jerárquica con respecto a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra. Así mismo, los elementos con contenido, deben estar correctamente cerrados. Como se muestra en el ejemplo siguiente:

```
<li>En XML la <b> estructura <i> es </i> jerárquica </b>.</li>
```

Etiquetas vacías.

HTML permite elementos sin contenido. XML también, pero la etiqueta debe ser de la siguiente forma <elemento sin contenido/>.

```
<li>En XML, es <br/> más restrictivo.</li>
```

Un solo elemento raíz

Los documentos XML sólo permiten un elemento raíz, por lo tanto todos los demás serán parte de éste. Es decir, la jerarquía de elemento de un documento XML bien formado sólo puede tener un elemento inicial.

Valores de atributos

Los valores de atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

```
<a herf="http://www.mipagina.com/">
```

Tipos de letras y espacios en blanco

XML es sensible al tipo de letra que se utiliza, es decir, trata las mayúsculas y minúsculas como caracteres diferentes. Por lo tanto, los elementos definidos como "FICHA", "Ficha", "ficha" y "fiCha" serían elementos diferentes.

Existe un conjunto de caracteres denominados "espacios en blanco" que los procesadores XML tratan de forma diferente en el marcado XML. Estos caracteres son los "espacios", tabuladores, retornos de carro y los saltos de línea.

La especificación XML 1.0 permite el uso de espacios en blanco para hacer más legible el código, y en general son ignorados por los procesadores XML.

4.4.2 Nombrando cosas

Al utilizar XML, es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. En XML los nombres tienen algunas características en común.

No se pueden crear nombres que empiecen con la cadena "xml", "xML", "XML" o cualquier otra variante. Las letras y rayas se pueden usar en cualquier parte del nombre. También se pueden incluir dígitos, guiones y caracteres de punto, pero no se puede empezar por ninguno de ellos. El resto de caracteres, como algunos símbolos, y espacios en blanco, no se pueden usar.

4.4.3 Marcado y datos

Las construcciones con etiquetas, referencias de entidad y declaraciones se denominan marcas. Éstas son las partes del documento que el procesador XML espera entender. El resto del documento que se encuentra entre las marcas son los datos que resultan entendibles por las personas.

Es sencillo reconocer las marcas en un documento XML. Son aquellas porciones que empiezan con "<" y acaban con ">", o bien, en el caso de las referencias de entidad, empiezan por "&" y acaban con ";".

4.4.4 Elementos

Los elementos XML pueden tener contenido (más elementos, caracteres, o ambos a la vez), o bien ser elementos vacíos.

Un elemento con contenido es, por ejemplo:

```
<nombre>Chapulín Colorado</nombre>
<aviso tipo="emergencia" gravedad="mortal">Que no cunda el
pánico</aviso>
```

Siempre empieza con una <etiqueta> que puede contener atributos o no, y termina con una </etiqueta> que debe tener el mismo nombre. Al contrario que HTML, en XML siempre se debe "cerrar" un elemento.

Hay que tener en cuenta que el símbolo "<" siempre se interpreta como inicio de una etiqueta XML.
Un elemento vacío, es el que no tiene contenido. Por ejemplo;

```
<identificador DNI="23123244"/>
<linea-horizontal/>
```

Al no tener una etiqueta de cierre que delimite un contenido, se utiliza la forma `<etiqueta/>`, que puede contener atributos o no.

4.4.5 Atributos

Como se ha mencionado antes, los elementos pueden tener atributos, que son una manera de incorporar características o propiedades a los elementos de un documento.

Por ejemplo, un elemento "historia" puede tener un atributo "tipo" y un atributo "calidad", con valores "mexicano" y "bueno" respectivamente.

```
<historia tipo="mexicano" calidad="bueno"> Estaban María y José
en su casa.. </historia>
```

En una Definición de Tipo de Documento, se especifican los atributos que pueden tener cada tipo de elemento, así como sus valores y tipos de valor posible.

Al igual que en otras cadenas literales de XML, los atributos pueden estar marcados entre comillas simples (') o doble ("). Cuando se usa uno para delimitar el valor del atributo, el otro tipo se puede usar dentro.

```
<verdura clase="zanahoria" longitud='15" y media'>
<cita texto="'Hola buenos días', dijo él">
```

A veces, un elemento con contenido, puede modelarse como un elemento vacío con atributos. Un concepto se puede representar de muy diversas formas, pero una vez elegida una, es aconsejable fijarla en el DTD, y usar siempre la misma, consistentemente dentro de un documento XML.

```
<gato><nombre>Micifús</nombre><raza>Persa</raza></gato>
<gato raza="Persa">Micifús</gato>
<gato raza="Persa" nombre="Micifús"/>
```

4.4.6 Entidades Predefinidas

En XML 1.0, se definen cinco entidades para representar caracteres especiales y que no se interpreten como marcado en el procesador XML. Es decir, así se puede usar

el carácter "<" sin que se interprete como el comienzo de una etiqueta XML, por ejemplo.

Tabla 5.2 Entidades de XML.

Entidad	Carácter
&amp;	&
&lt;	<
&gt;	>
&apos;	'
&quot;	"

4.4.7 Secciones CDATA

Existe otra construcción en XML que permite especificar datos, utilizando cualquier carácter, especial o no, sin que se interprete como marcado XML. La razón de esta construcción llamada CDATA (Character DATA) es que a veces es necesario para los autores de documentos XML, poder leerlo fácilmente sin tener que descifrar los códigos de entidades. Especialmente cuando son muchos.

Como ejemplo, el siguiente (primero usando entidades predefinidas y luego con un bloque CDATA).

```
<parrafo>Lo siguiente es un ejemplo de HTML.</html>
<ejemplo>
&lt;html>
&lt;head>&lt;title>Rock &amp; Roll&lt;/title>&lt;/head>
</ejemplo>
<ejemplo>
<![CDATA[
<html>
<head><title>Rock & Roll</title></head>]]>
</ejemplo>
```

Como hemos visto dentro de una sección CDATA podemos poner cualquier cosa, que no será interpretada como algo que no es así. Existe una excepción y es la

cadena "]]>" con el que termina el bloque CDATA. Esta cadena no puede utilizarse dentro de una sección CDATA.

4.5 Objeto XMLHttpRequest

El Objeto *XMLHttpRequest* fue creado por Alex Hopmann, un desarrollador que trabajaba para Microsoft. Alex Hopmann ingresó a Microsoft en Noviembre de 1996, tiempo después fue cambiado a Redmond en la primavera de 1997. Comenzó trabajando en algunos estándares de Internet relacionados con el futuro de Outlook. Específicamente, trabajaba con meta-datos para sitios Web incluyendo una propuesta llamada "Colecciones Web".

Posteriormente, conoció a Jean Paoli, Jean trabajaba en ese entonces con XML y sospechaba que algún día sería algo muy grande. Rápidamente Alex tomo ventaja de esta tecnología y publicó "Colecciones Web usando XML", a lo que la gente respondió instantáneamente ya que XML era un tema de discusión actual.

Esto provocó cierta inquietud sobre la gente que trabajaba en el nuevo Outlook, lo que causó que Alex comenzara a trabajar sobre Visual Basic y crear la primera versión del XMLHTTP. Ésta primera versión no soportaba las llamadas asíncronas pero eso no importó para hacer uso de éste.

La finalidad del XMLHTTP era realizar llamadas al servidor, pero para realizar esas llamadas necesitaba de controles ActiveX, por lo que se procedió a trabajar sobre el componente.

Shawn Bracewell era un desarrollador del equipo OWA (Outlook Web Access) de Microsoft, quien reescribió todo el código y lo hizo mas sólido, además, le agregó el soporte asíncrono para posteriormente añadirlo al Internet Explorer.

XMLHttpRequest es una interfaz para realizar llamadas mediante HTTP, por lo que es recomendable un buen conocimiento de este protocolo. Es importante el manejo

correcto de la cache en el servidor HTTP, en los proxy cache intermedios y en el navegador WEB. Otro punto importante es el manejo de hojas de caracteres, la codificación y decodificación de texto y su identificación mediante cabeceras HTTP y MIME.

XMLHttpRequest utiliza UTF-8 para la codificación de cadenas de texto y objetos binarios. La única excepción a esta regla es la transmisión de cadenas XML cuando estos datos XML especifiquen explícitamente una codificación alternativa, en caso contrario se usará UTF-8 por defecto. Es importante tener esto en cuenta en entornos dónde se mezclen varias codificaciones, por ejemplo, pueden producirse errores de visualización de caracteres al incorporar funcionalidad AJAX a una página WEB codificada con ISO 8859-1.

La interfaz se presenta encapsulada en una clase. Para utilizarla la aplicación cliente debe crear una nueva instancia mediante el constructor adecuado. Es posible realizar peticiones síncronas y asíncronas al servidor; en una llamada asíncrona el flujo de proceso no se detiene a esperar la respuesta como se haría en una llamada síncrona, si no que se define una función que se ejecutará cuando se complete la petición: un manejador de evento.

XMLHTTPRequest permite la transmisión de información entre el servidor y el cliente, es decir el navegador, sin necesidad de que la página Web tenga que ser cargada nuevamente. Este principio ha sido usado por Google para mejorar la experiencia de sus usuarios y según muchos expertos ha sido la base del éxito de los servicios de Google.

Para poder utilizar el *XMLHTTPRequest*, es necesario antes que otra cosa, obtener una instancia del objeto mediante un método o rutina. Existen diversas formas de crear la instancia entre navegadores, el Internet Explorer permite realizar conexiones

Cross-Domain[1], es decir, permite hacer solicitudes entre diferentes dominios, y el Firefox no lo permite.

Explorer integra un objeto ActiveX para crear la instancia necesaria el cual es el encargado de establecer la conexión, para otros navegadores simplemente llama al objeto como *XMLHTTPRequest*.

```
function conexion(url)
     {
          if(window.XMLHttpRequest)
          {
               // XMLHttpRequest nativo
               gSolicitud = new XMLHttpRequest();
               gSolicitud.onreadystatechange = eventosServidor;
               gSolicitud.open("GET", url, true);
               gSolicitud.send(null);
          }
          else if(window.ActiveXObject)
          {
               // XMLHttpRequest ActiveX (Internet Explorer -
Windows)
               gSolicitud = new
ActiveXObject("Microsoft.XMLHTTP");
               if(gSolicitud)
               {
                    gSolicitud.onreadystatechange =
eventosServidor;
                    gSolicitud.open("GET", url, true);
                    gSolicitud.send();
               }
          }
     }
```

[1] Se refiere a una situación en la que se utilizan conexiones a varios servidores o a uno de firma invisible que provee de algún servicio.

Tabla 5.3 Propiedades del Objeto XMLHTTPRequest.

Propiedad	Descripción
`readyState`	Devuelve el estado del objeto como sigue: 0 = sin inicializar, 1 = cargando, 2 = cargado, 3 = interactivo y 4 = completado.
`responseText`	Devuelve la respuesta como una cadena
`responseXML`	Devuelve la respuesta como XML. Esta propiedad devuelve un objeto documento XML, que puede ser examinado usando las propiedades y métodos del árbol del DOM.
`Status`	Devuelve el estado como un número (p. ej. 404 para "Not Found" y 200 para "OK").
`statusText`	Devuelve el estado como una cadena (p. ej. "Not Found" o "OK").

Tabla 5.4 Métodos del Objeto XMLHTTPRequest.

Método	Descripción
`abort()`	Cancela la petición en curso
`getAllResponseHeaders()`	Devuelve el conjunto de cabeceras HTTP como una cadena.
`getResponseHeader (nombreCabecer )`	Devuelve el valor de la cabecera HTTP especificada.
`open ( método, URL [, asíncrono [, nombreUsuario [, clave]]] )`	Especifica el método, URL y otros atributos opcionales de una petición. El parámetro de método puede tomar los valores "GET", "POST", o "PUT" ("GET" y "POST" son dos formas para solicitar datos, con "GET" los parámetros de la petición se codifican en la URL y con "POST" en las cabeceras de HTTP). El parámetro *URL* puede ser una URL relativa o completa. El parámetro *asíncrono* especifica si la petición será gestionada asíncronamente o no. Un valor *true* indica que el proceso del script continúa después del método send(), sin esperar a la respuesta, y *false* indica que el script se detiene hasta que se complete la operación, tras lo cual se reanuda la ejecución. En el caso asíncrono se especifican manejadores de eventos, que se ejecutan ante cada cambio de estado y permiten tratar los resultados de la consulta una vez que se reciben, o bien gestionar eventuales errores.
`send (contenido)`	Envía la petición
`setRequestHeader (etiqueta,valor)`	Añade un par etiqueta/valor a la cabecera HTTP a enviar.

Tabla 5.5 Eventos del Objeto XMLHTTPRequest.

Propiedad	Descripción
`onreadystatechange`	Función que se activa con cada cambio de estado.

Este capítulo muestra como funciona cada una de las tecnologías que conforman AJAX, parece poco complicado pensar que cada una de ellas tiene una función específica, sin embrago, cada una está enfocada a realizar una tarea.

JavaScript es el controlador de todas porque de una u otra forma hace el enlace de las demás. *CSS*, se encarga de darle estilo al contenido de la página brindando una vista agradable, siempre y cuando se sepa aplicar. El *DOM* es el encargado de acceder a las etiquetas HTML, con esto se consigue colocar o manipular información en un objeto y/o etiqueta especifica. *XML,* funciona como contenedor de información que esta siempre disponible para entregar datos en cuanto se le requieran con la ventaja de permanecer en el navegador durante el uso de la página. Y por último, *XMLHTTPRequest*, que es el medio de comunicación entre el documento HTML y XML, de esta manera se consigue que la presentación de la información se mucho mas ágil.

En el próximo capítulo se muestra un ejemplo de código en AJAX. Ese ejemplo es un código muy sencillo, en el cual despliega el menú de un restaurante en forma clasificada, según el tipo de comida que se seleccione y el contenido es generado dinámicamente.

Capítulo 6 CASO PRÁCTICO

Para comprender el funcionamiento de AJAX fue necesario crear un pequeño ejemplo. Este ejercicio tiene la función de generar una consulta y devolver los resultados de una opción del menú de un restaurant seleccionado.

El código del ejemplo quedo de la siguiente manera:

```
var gSolicitud; // Para objeto XMLHttpRequest
```

Para empezar, se define una variable para trabajar con el objeto `XMLHttpRequest` de manera global, que en este caso es `gSolicitud`.

```
function detalleCategoria(listaCategorias)
{
     var opcionSeleccionada = listaCategorias.selectedIndex;

     if(opcionSeleccionada == 0 || opcionSeleccionada == -1)
     {
          alert("Debe seleccionar una Categoria");
          listaCategorias.focus();
          return false;
     }
```

Se comprueba que realmente se haya seleccionado una opción, de lo contrario no continua la preparación de la información para que pueda ser enviada.

```
     var fecha = new Date();
     var marcaTiempo = encodeURIComponent(fecha.getTime());

     var claveCategoria =
encodeURIComponent(listaCategorias.options[opcionSeleccionada].
value);

     var urlParametros = "?claveCategoria=" + claveCategoria +
"&marcaTiempo=" + marcaTiempo;
```

```
     var url = " http://localhost/xml_multiples/procesa_xml.php
" + urlParametros;

conexion(url);
}
```

Una vez que se comprobó que se hizo la selección de una opción, se preparan todas las variables que se van a enviar, codificándolas mediante el `encodeURIComponent()` con la finalidad de evitar que Internet Explorer guarde en caché el resultado de la solicitud al servidor y asigna a la variable `urlParametros` la cadena de parámetros para posteriormente crear completamente la dirección con la que se va a establecer la comunicación y enviarla a la función que se encargará de hacer la conexión `conexion(url)`.

```
function conexion(url)
{
     if(window.XMLHttpRequest)
     {
          // XMLHttpRequest nativo
          gSolicitud = new XMLHttpRequest();
          gSolicitud.onreadystatechange = eventosServidor;
          gSolicitud.open("GET", url, true);
          gSolicitud.send(null);
     }
     else if(window.ActiveXObject)
     {
          // XMLHttpRequest ActiveX (Internet Explorer -
Windows)            gSolicitud = new
ActiveXObject("Microsoft.XMLHTTP");
          if(gSolicitud)
          {
               gSolicitud.onreadystatechange = eventosServidor;
               gSolicitud.open("GET", url, true);
               gSolicitud.send();
          }
     }
}
```

La función `conexion()` establece la comunicación con el servidor, es decir, se comunica con la página generadora del XML y recibe los resultados.

Para establecer la conexión es necesario obtener el objeto `XMLHttpRequest`, conteniéndolo en la variable `gSolicitud` declarada con anterioridad para poder manejar el objeto.

La propiedad `onreadystatechange` determina los estados por los que pasa el proceso de "envío-recepción" proporcionados por la función `eventosServidor()`.

El método `open()` establece la forma de envío a través de un "GET" o un "POST", la ruta del servidor acompañada de los parámetros y la manera de conexión ya sea asíncrona `(TRUE)` o síncrona `(FALSE)`.

Y por último, el método `send()` envía la información al servidor y recibe los resultados en formato XML en este caso.

```
function eventosServidor()
{
if(gSolicitud.readyState == 4) // Estado de la solicitud, 4 =
Proceso se completo
{
          if(gSolicitud.status == 200) // HTTP status, 200 =
Proceso concluyo correctamente
{

var nodoClaveCategoria =
gSolicitud.responseXML.getElementsByTagName("clavecategoria");
var nodoCategoria =
gSolicitud.responseXML.getElementsByTagName("categoriadescripci
on");
var nodoProducto =
gSolicitud.responseXML.getElementsByTagName("producto");
var nodoProveedor =
gSolicitud.responseXML.getElementsByTagName("proveedor");
var nodoExistencia =
gSolicitud.responseXML.getElementsByTagName("existencia");
```

```
renglonBorrarRango("tablaProductos", 1, 0)

var nodosCuantos = nodoClaveCategoria.length;

for(var contador = 0; contador < nodosCuantos; contador++)
{
          claveCategoria =
nodoClaveCategoria[contador].childNodes[0].nodeValue;
          categoria =
nodoCategoria[contador].childNodes[0].nodeValue;
          producto =
nodoProducto[contador].childNodes[0].nodeValue;
          proveedor =
nodoProveedor[contador].childNodes[0].nodeValue;
          existencia =
nodoExistencia[contador].childNodes[0].nodeValue;

          renglonAgregar("tablaProductos", 0, claveCategoria,
categoria, producto, proveedor, existencia);
}
}
else
          alert("Se presentaron problemas al procesar la
informacion:\n" + gSolicitud.statusText);

     }
}
```

La propiedad `readyState` determina el estado del objeto `XMLHttpRequest`, al lograr conseguir el objeto devolverá el valor de `4` que significa que se consiguió, posteriormente el propiedad `status` determina si se a logrado concluir el proceso de envío-recepción, devolviendo el valor de `200`, en caso de no ser este el valor, se presentará un mensaje de error.

Cuando se consigue `gSolicitud.readyState == 4` *y* `gSolicitud.status == 200`, *se obtiene el numero de hijos que tiene el documento XML para poder hacer un recorrido completo del archivo, a continuación se obtiene el nombre de cada uno de los nodos y se limpia o borran los renglones de la tabla con la función*

`renglonBorrarRango()` donde se desplegarán los datos nuevos dentro de el documento HTML.

A continuación se hace el recorrido de todos los nodos de los hijos del documento XML y se obtienen sus valores, una vez que sucede esta acción se crea dinámicamente la tabla de datos con la función `renglonAgregar()` con el valor del nodo correspondiente.

```
function renglonBorrarRango(tablaID, renglonInicio,
renglonFinal)
{
     var tabla = document.getElementById(tablaID);
     var cuantosBorrar = tabla.rows.length - renglonInicio;
     var contador = 1;

while(contador <= cuantosBorrar)
     {
          tabla.deleteRow(tabla.rows.length - 1);
          ++contador;
     }
}
```

La función `renglonBorrarRango()` es la encargada de borrar las celdas de la tabla en la que se despliega la información en caso de que la haya. Obtiene el total de los renglones y posteriormente elimina renglón por renglón.

```
function renglonAgregar(tablaID, posicion, claveCategoria,
categoria, producto, proveedor, existencia)
{
     var tabla = document.getElementById(tablaID);

     if(posicion > 0)
          var renglon = tabla.insertRow(posicion);
     else
          var renglon = tabla.insertRow(tabla.rows.length);


     var celda = renglon.insertCell(0);
     celda.innerHTML = claveCategoria;
```

```
        celda = renglon.insertCell(1);
        celda.innerHTML = categoria;

        celda = renglon.insertCell(2);
        celda.innerHTML = producto;

        celda = renglon.insertCell(3);
        celda.innerHTML = proveedor;

        celda = renglon.insertCell(4);
        celda.innerHTML = existencia;

        renglonColor(tabla);

        renglonResalta(tabla);
}
```

La función `renglonAgregar()` construye cada una de las celdas de la tabla en la que se despliega la información, además de asignar los valores contenidos dentro del documento XML, la propiedad `innerHTML` es la encargada de escribir dentro de un elemento de HTML.

```
function renglonColor(tabla)
{
        var renglones = tabla.rows.length;
        var renglonInicio = 1;
        var renglonFinal = renglones - 1;
        var claseAlternar = "tablaAlterno1";

        for(contador = renglonInicio; contador <= renglonFinal;
++contador)
        {
                var renglon = tabla.rows[contador];

                if(contador == 1 && renglones > 2)
                        var claseAlternar = renglon.className;

                renglon.setAttribute("className", claseAlternar); //
Para Internet Explorer
                renglon.setAttribute("class", claseAlternar); // Para
Firefox
```

```
            // Alternar color
            if(claseAlternar == "tablaAlterno2")
                 claseAlternar = "tablaAlterno1";
            else
                 claseAlternar = "tablaAlterno2";

      }
}

function resaltar(renglon, claseRegular, claseResaltar, estado)
{
      if(estado == 'on')
      {
            renglon.className = claseResaltar;

      }
      else
      {
            renglon.className = claseRegular;
      }
}

function renglonResalta(tabla)
{
      var navegador = defineNavegador();
      var renglones = tabla.rows.length;
      var renglonInicio = 1;
      var renglonFinal = renglones - 1;

      for(contador = renglonInicio; contador <= renglonFinal;
++contador)
      {
            var renglon = tabla.rows[contador];
                  var claseAlternar = renglon.className;

            if(navegador == "ie")
            {
                 if(claseAlternar == "tablaAlterno1")
                 {
                       renglon.onmouseout =  function()
{
resaltar(this,'tablaAlterno1','tablaResaltar','off');
}

renglon.onmouseover = function()
{
```

```
resaltar(this,'tablaAlterno1','tablaResaltar','on');
}
                }
                else
                {
                     renglon.onmouseout =  function()
{
resaltar(this,'tablaAlterno2','tablaResaltar','off');
}
                     renglon.onmouseover = function()
{
resaltar(this,'tablaAlterno2','tablaResaltar','on');
}
                }
           }
           else
           {
                renglon.setAttribute("onMouseOver",
"resaltar(this,'" + renglon.className +
"','tablaResaltar','on')");
                renglon.setAttribute("onMouseOut",
"resaltar(this,'" + renglon.className +
"','tablaResaltar','off')");
           }
     }
}
```

Las funciones `renglonColor()` y `resaltar()` son las encargadas de controlar y asignar los estilos mediante CSS y la función `renglonResalta()` permite la interacción entre cada uno de los renglones asignándole un nuevo estilo para su mayor apreciación.

Código HTML:

```
<html>
<head>
     <meta http-equiv="Content-Type" content="text/html;
charset=iso-8859-1">
   <title>AJAX - XML</title>
     <link href="estilos.css" rel="stylesheet" type="text/css">
     <script type="text/javascript"
src="defineNavegador.js"></script>
```

```
     <script type="text/javascript" src="ajax.js"></script>
</head>
   <body>
<form action="" target="" method="post" name="captura"
id="captura" autocomplete="off">
     <table name="opciones" id="opciones" width="98%"
align="center" border="0" cellspacing="0" cellpadding="0"
class="tabla">
          <tr class="tablaTitulo">
               <td>
                    Categorias
                    <select name="categorias"
id="name="categorias"" size="1"
onChange="detalleCategoria(this);">
                    <option value="0">-- Selecione una
categoria --</option>
                    <option value="1">Bebidas</option>
                    <option value="6">Carnes</option>
                    <option value="2">Condimentos</option>
                    <option value="5">Granos/Cereales</option>
                    <option value="4">Lacteos</option>
                    <option value="8">Mariscos</option>
                    <option value="3">Postres</option>
                    <option value="7">Vegetales</option>
                    </select>
               </td>
          </tr>
     </table>
     <table name="tablaProductos" id="tablaProductos"
width="98%"  align="center" border="0" cellspacing="0"
cellpadding="0" class="tabla">
          <tr class="tablaTitulo">
               <td class="bordeTituloInferior">Clave</td>
               <td class="bordeTituloInferior">Categoria</td>
               <td class="bordeTituloInferior">Producto</td>
               <td class="bordeTituloInferior">Proveedor</td>
                         <td
class="bordeTituloInferior">Existencia</td>
          </tr>
     </table>
</form>
   </body>
</html>
```

Al momento de ejecutar la aplicación, el contenido de la página es generado dinámicamente, pero es interesante ver que en ningún momento se puede apreciar algún instante de recarga de página.

Los datos que se presentan como resultado de la consulta, son extraídos de una base de datos, la cual es consultada a través de una interfaz en PHP y esta a su ves es desplegada en formato XML para que el objeto *XMLHTTPRequest* pueda acceder a ella y manipular el contenido vía *JavaScript*.

En una consulta clásica, directamente en una interfaz hecha en algún lenguaje servidor, se podría apreciar como las peticiones hacen notorio como se brinca de página en página hasta presentar el desplegado de la información.

AJAX demuestra como mientras se hacen las ejecuciones de algunos controles, se podrían hacer algunas otras ejecuciones sin verse interrumpidas entre si mismas y de esta manera hace que sea mas funcional e interactiva.

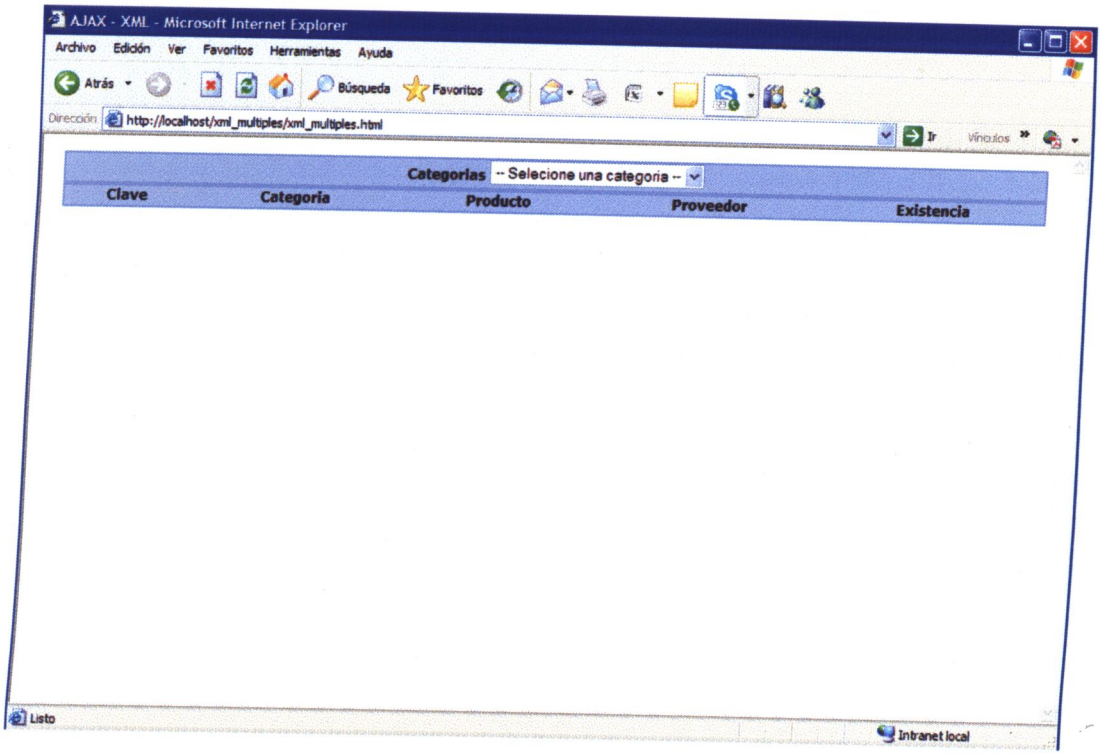

Figura 5.1 Ejemplo del funcionamiento de aplicación AJAX

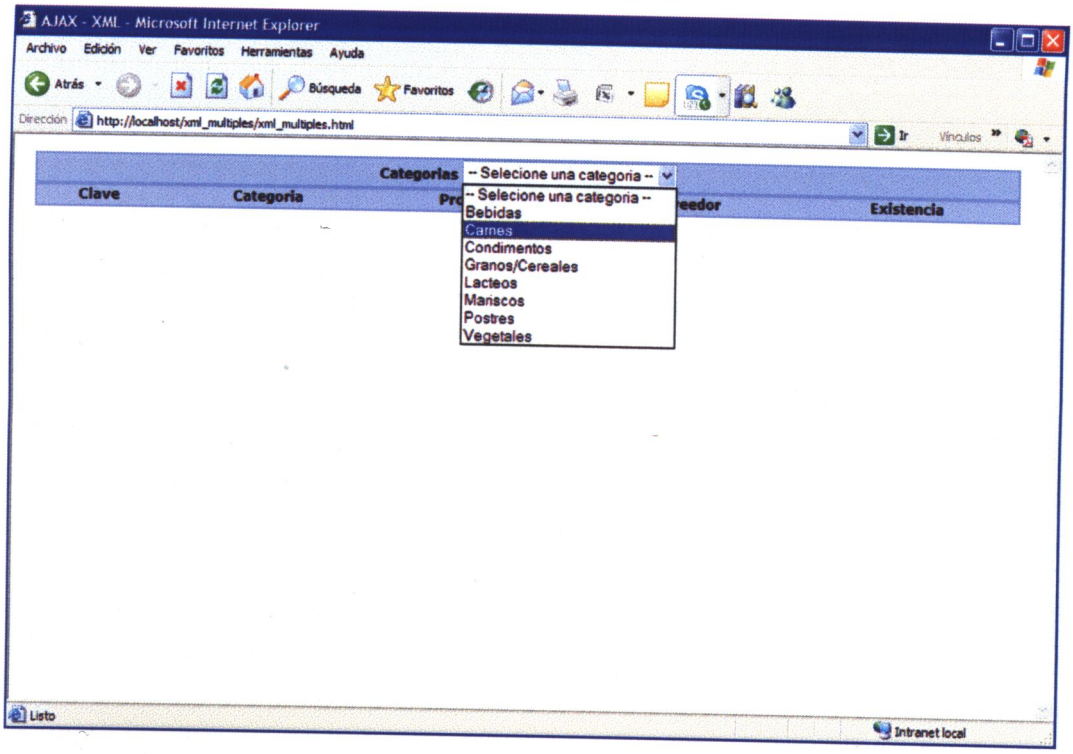

Figura 5.2 Ejemplo del funcionamiento de aplicación AJAX

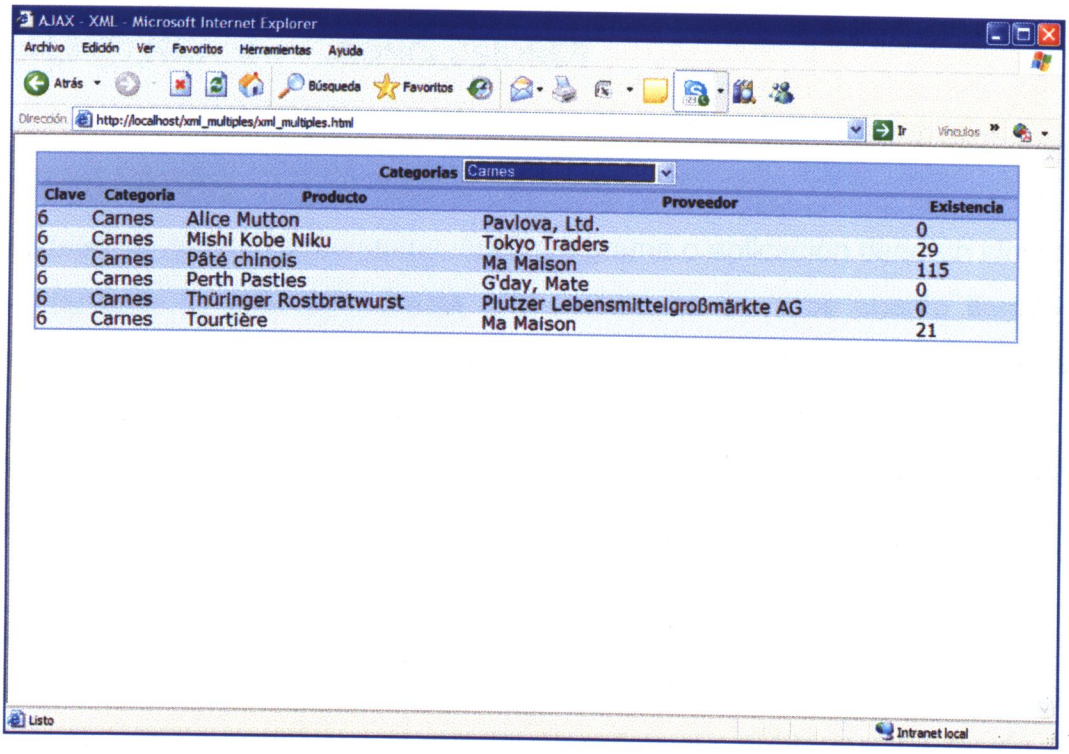

Figura 5.3 Ejemplo del funcionamiento de aplicación AJAX

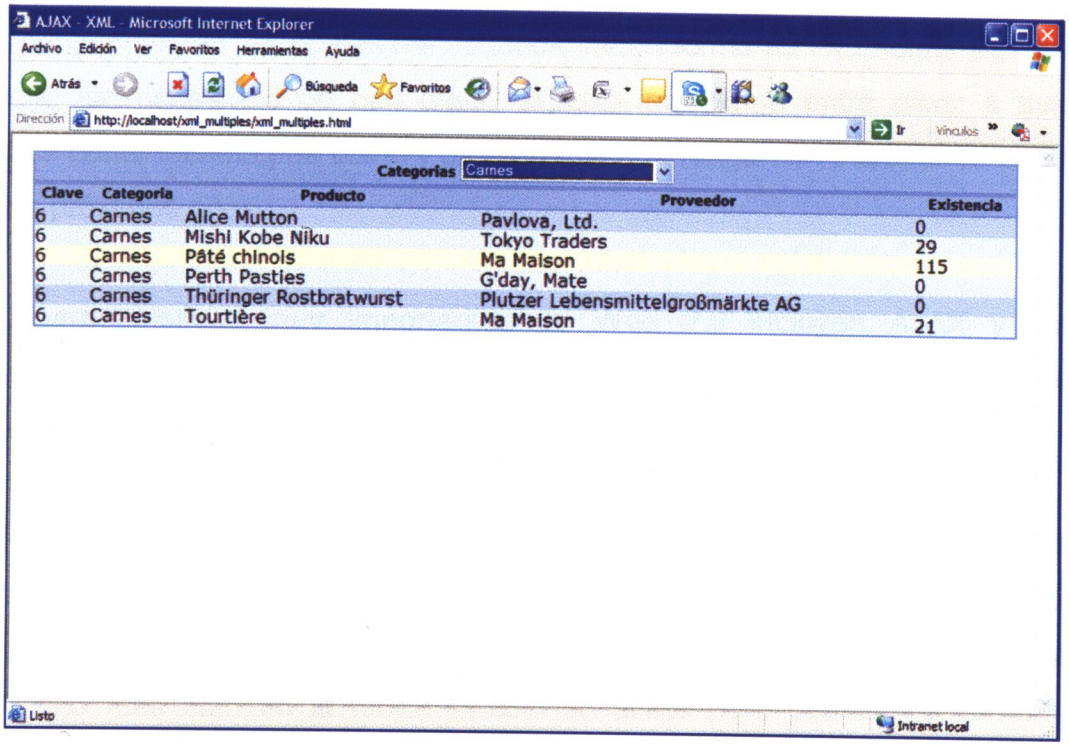

Figura 5.4 Ejemplo del funcionamiento de aplicación AJAX

Capítulo 7 CONCLUSIONES

6.0 Conclusiones

Como bien se sabe en informática, cada programa o aplicación tiene un ciclo de vida, AJAX también lo tiene pero hasta el momento no se sabe cuanto tiempo estará vigente. Lo único que se sabe es que está presente e irá evolucionando; por ello, las aplicaciones Web inician una nueva generación de aplicaciones orientadas al fácil manejo de sus interfaces, rapidez y funcionalidad, pero se busca, por otro lado, percibir el impacto que esta causado o puede causar en un futuro.

Con esta evolución se abre paso a un nuevo término, *"La Web 2.0"*. La *Web 2.0,* no es otra cosa más que la forma en que el usuario tiene contacto con las aplicaciones y es únicamente un punto con el que se identifica el cambio que ha tenido *Internet.* Este cambio es atribuido a la forma en que se desarrollan aplicaciones más independientes, en las cuales los usuarios tienen forma de interactuar de forma mas directa, sin necesidad de que algún experto intervenga entre ellos, a esto se le conoce como tecnologías *using* y es probable que tengan un incremento exponencial en los próximos años.

Dos características muy importantes de Web 2.0 incluyen AJAX y contenido generado por el usuario. En la actualidad, se está comenzando a adoptar estos dos conceptos, ocasionando problemas de accesibilidad nunca antes vistos.

Las páginas Web basadas en AJAX requieren soporte para JavaScript, pero tecnologías más preparadas para ayudar pueden ahora soportar algunos tipos de JavaScript. La principal preocupación de accesibilidad no radica por lo tanto en su uso, sino más bien en la forma la cual se utiliza para causar los cambios en la página.

Un ejemplo de estas aplicaciones es la utilización deslizadores de arrastrar-y-soltar para permitir a los usuarios expander y angostar un amplio rango de criterios de filtrado. Actualizando automáticamente para mostrar cuántos resultados conforman los criterios seleccionados del usuario.

El ejemplo muestra qué tan sorprendente puede ser la usabilidad para muchos usuarios Web. Pero es totalmente imposible para lectores de pantalla y usuarios que se manejan sólo con el teclado.

Otro ejemplo muy común es el de los sitios de contenido generado por los usuarios, como los blogs y wikis que están siendo cada vez más comunes.

6.1 Perspectivas de AJAX

- Permite obtener un comportamiento muy parecido al de aplicaciones de escritorio en la Web.
- Supone una nueva era en el diseño de aplicaciones Web aproximándolas desde una perspectiva de usuario a la experiencia de uso de aplicaciones stand-alone.
- Facilidad y rapidez de uso, la comunicación cliente-servidor es menos notoria similar al software residente en el propio equipo.
- Permite compartir archivos multimedia e interactuar con ellos.
- . Diseño de servicios que contengan efectos, acudiendo al modelo tradicional de refresco de páginas.
- Retroalimentación al usuario a través de la interfaz. A un usuario acostumbrado a un modelo tradicional de interacción Web, el refresco de página le informa que el sistema ha recibido su petición.
- Acceso a información mediante indexación y marcado de *Bookmarks*.

Con AJAX hay que ser muy cautelosos al diseñar flujos de pantallas ya que se puede caer en el error de ocultar información que puede ser idónea.

6.2 Tendencias de AJAX

Hay tres factores principales que modelarán en el futuro la accesibilidad Web: AJAX, contenido generado por el usuario. La creciente prominencia de esos factores podría llevar a algunos de los siguientes:

1. La accesibilidad estará cada vez menos condicionada por la manera en que funciona la aplicación.

Con la llegada de nuevas tecnologías y la vaga naturaleza, tecnológicamente neutral de la nueva, la accesibilidad se está convirtiendo en algo cada vez menos condicionado por las guías. Esto significa que emplear a expertos en accesibilidad cobrará una importancia cada vez mayor para las organizaciones, ya que interpretar correctamente los lineamientos será cada vez más difícil.

2. Las aplicaciones Web serán cada vez más parecidas a las stand-alone.

Por primera vez la usabilidad y accesibilidad están muy parejas entre sí con *interfaces* interactivas muy parecidas, tal es el caso de el Microsoft Outlook. El contenido generado por el usuario es probable que ofrezca una accesibilidad pobre ya que se está haciendo cada vez más común en la Web. Este tipo de contenido está siendo creado tan rápidamente que va ser imposible controlarlo desde el punto de vista de la accesibilidad.

3. JavaScript, PDF & Flash ya no serán rechazados en las aplicaciones Web.

En WCAG (Web Content Accessibility Guideline) 1.0, a los administradores de la Web y desarrolladores se les dijo básicamente que sus sitios Web no deberían basarse en ninguna de esas tres tecnologías. WCAG 2.0, por el contrario, no estipula esto y ahora está soportado por muchas tecnologías actuales.

Definitivamente es uno de los logros más exitosos dentro de Internet, porque involucra varias tecnologías, *AJAX,* posee grandes beneficios en las aplicaciones Web actuales, rompiendo con los paradigmas de programación antiguos.

Esta técnica aporta grandes cualidades, como lo son, la interactividad entre el servidor y el usuario ya que toda la comunicación se hace sin necesidad de esperar mientras se refresca la pantalla para obtener los resultados de una acción.

Por ejemplo, suponiendo que se hace una consulta sencilla a una base de datos, en otros tiempos, tendríamos que esperar a que el servidor enviara los datos y el navegador desplegara en la pantalla la información, a demás de volver a colocar todos los elementos que forman la página perdiendo de vista todo en la pantalla y creando tiempos de espera, que en algunas ocasiones resulta molesto. Con *AJAX* se puede lograr la misma consulta pero de una forma más dinámica, pero en lugar de hacer nuevamente la carga de todos los elementos, únicamente se hace el desplegado de la consulta en algún área que se indique como si fuera una pequeña página dentro de otra, esta consulta es mas rápida porque es menos la información que se envía y que se despliega.

De esta forma se ahora tiempo y ancho de banda y se da la apariencia de que se esta trabajando con una aplicación stand-alone o de escritorio como puede ser un *Word, un Excel, etc.*

Por otro lado, brinda una agradable experiencia en el uso de la aplicación, logrando que el usuario controle de mejor forma el contenido que se le presenta sin caer en las molestas recargas y sin que sea tediosa.

En un futuro no muy lejano, podría pasar que la mayoría de las aplicaciones serán manejadas vía Internet y muy probablemente abra paso a otro tipo de aplicaciones, como para dispositivos móviles, que están limitados por sus recursos pero son en la actualidad parte de la mayoría de la gente, por ejemplo.

Es posible que uno de los grandes impedimentos que puedan detener este tipo de implementaciones, sea el propio usuario, ya que está acostumbrado a ciertos funcionamientos y sea difícil adaptarse a otro tipo de aplicaciones, mientras tanto tenemos que darnos a la tarea de aprender y difundir los grandes beneficios que AJAX tiene y ampliar nuestras expectativas, a saber que los sitios web pueden aportarnos mucho más de lo que hasta hoy nos han aportado y poco a poco irnos acostumbrando a utilizar sitios web basados en este tipo de tecnologías.

BIBLIOGRAFÍA

[1] O. Darie Cristian, Chereches-Tosa, Brinzarea Bogdan, Bucica Mihai.*AJAX and PHP Building Responsive Web Aplications*. Ed. Packt Publishing. Brimingham, UK. Marzo 2006.

[2] Crane Dave, Pascarello Eric with Darren James. *AJAX IN ACTION.*Ed. Manning Publications. USA. 2006.

[3] Fernando Berzal, Juan Carlos Cubero & Francisco J. Cortijo, *Desarrollo Profesional de Aplicaciones Web con ASP.NET*, iKor Consulting

[4] Rafael López Lita, Francisco Fernández Beltrán, *La comunicación local por Internet,*Universitat Jaume I. Publicaciones, 2005

[5] Ryan Asleson, Nathaniel T. Schutta, *Foundations of Ajax,* Apress, 2005.

[6] Laurence Moroney, *Beginning Web Development, Silverlight, and ASP.NET AJAX: From Novice to Professional,* Apress, 2008.

[7] Hossein Bidgoli, *The Internet Encyclopedia,* John Wiley and Sons, 2004.

[8] Kris Jamsa, *.NET Web Services Solutions,* Wiley_Default, 2003.

[9] Michel Dreyfus, *Hojas de estilo: Cascading Style Sheet: CSS,* Marcombo, 2001.

[10] Mathematics and Computer Science Delft University of Technology, *Software Engineering Research Group Department of Software Technology Faculty of Electrical Engineering*, ISSN 1872-5392.

[11] http://www.alzado.org/articulo.php?id_art=494 visitado en el mes de febrero, marzo y abril de 2008.

[12] http://www.javahispano.org/news.item.action?id=336613008 visitado en el mes de marzo de 2008.

[13] http://www.w3.org/DOM/ visitado en el mes de mayo y junio de 2007.

[14] http://www.sidar.org/recur/desdi/traduc/es/xml/xml10p/xml10p.htm visitado en el mes de junio de 2008.

[15] http://www.prototypejs.org/2007/11/7/prototype-1-6-0-script-aculo-us-1-8-0-and-the-bungee-book-now-available visitado en el mes de febrero y marzo de 2008.

[16] http://www.alzado.org/articulo.php?id_art=457 visitado en el mes de marzo y abril de 2008.

[17] http://ajax.javabeat.net/articles/2007/05/comparison-of-ajax-frameworks-prototype-gwt-dwr-thinware/ visitado en el mes de abril y mayo de 2007.

[18] http://www.ajaxman.net/category/ajax/ visitado en el mes de abril y mayo de 2007.

[19] http://ajaxian.com/archives/ajaxiancom-2006-survey-results visitado en el mes de abril y mayo de 2007.

[20] http://www.loretahur.net/AprendiendoCSS/versiones-de-css/ visitado en el mes de junio y julio de 2007.

[21] http://www.learnthenet.com/spanish/html/50wtools.htm visitado en el mes de mayo de 2008.

[22] http://www.olwebs.com/articulo.php?id_contenido=7 visitado en el mes de julio de 2007.

[23] http://www.gaiasur.com.ar/infoteca/siggraph99/diseno-de-interfaces-y-usabilidad.html#5 visitado en el mes de octubre y noviembre de 2006.

[24] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/rmscpt/Html/rmscpt6.asp visitado en el mes de enero y febrero de 2007.

[25] http://www.baquia.com/noticias.php?id=10977 visitado en el mes de abril y mayo de 2007.

[26] http://www.ortizmania.com/online/articulo.asp?art=6 visitado en el mes de mayo y junio de 2007.

[27] http://www.htmlpoint.com/dhtml/20/index.html visitado en el mes de mayo y junio de 2007.

[28] http://www.alexhopmann.com/story-of-xmlhttp/ visitado en el mes de mayo y junio de 2007.

[29] http://ajaxian.com/archives/ajaxiancom-2006-survey-results visitado en el *mes de agosto y septiembre de 2007.*

[30] http://www.wrox.com/WileyCDA/Section/id-306214.html visitado en el mes de septiembre y octubre de 2007.

[31] http://www.jorgeivanmeza.com/blog/2008/03/26/introspeccion-de-objetos-con-prototype/ visitado en el mes de septiembre y octubre de 2007.

[32] http://www.usolab.com/articulos/desafios_interfaz_web_2.php visitado en el mes de febrero y marzo de 2008.

[33] http://www.w3c.es/Divulgacion/Guiasbreves/ServiciosWeb visitado en el mes de marzo y abril de 2007.

[34] http://observatorio.cnice.mec.es/modules.php?op=modload&name=News&file=article&sid=363 visitado en el mes de febrero, marzo y abril de 2004.

[35] http://www.w3c.es/Consorcio/ visitado en el mes de marzo y abril de 2007.

[36] http://www.iec.csic.es/CRIPTonOMICon/javascript/ visitado en el mes de octubre y noviembre de 2006.

[37] http://www.unav.es/cti/manuales/TutorialJavaScript/lecciones/leccion1.html visitado en el mes de mayo y junio de 2007.

[38] http://www.w3c.es/Divulgacion/Guiasbreves/HojasEstilo visitado en el mes de mayo y junio de 2007

[39] http://geneura.ugr.es/~maribel/xml/introduccion/index.shtml#11 visitado en el mes de mayo y junio de 2007.

[40] http://www.w3c.es/Divulgacion/Guiasbreves/TecnologiasXML visitado en el mes de mayo y junio de 2007.

[41] http://www.ulpgc.es/otros/tutoriales/xml/ visitado en el mes de mayo y junio de 2007.

[42] http://fresno.cnice.mecd.es/~avaler3/?id=ajax-y-otros-remiendos visitado en el mes de septiembre y octubre de 2007.

[43] http://www.lsi.us.es/docencia/get.php?id=352 visitado en el mes de marzo y abril de 2008.

[44] http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node11.html visitado en el mes de marzo y abril de 2008.

[45] http://iteso.mx/~carlosc/pagina/documentos/usabilidad.htm visitado en el mes de marzo y abril de 2008.

[46] http://homepages.mty.itesm.mx/al450951/ visitado en el mes de marzo y abril de 2008.

[47] http://directwebremoting.org/dwr/overview/dwr visitado en el mes de julio, agosto de 2008.

[48] http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm visitado en el mes de julio, agosto de 2008.

[49] http://www.w3schools.com/Ajax/Default.Asp visitado en el mes de noviembre de 2008.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO DE TERMINOS

Protocolo: Reglas que permiten el intercambio de datos entre computadoras que forman parte de una red, es decir, establecen la comunicación.

Aplicación Stand-Alone: Es una aplicación autónoma o independiente que funciona por si misma que no requiere estar conectada en red o requiere de algún otro recurso en otra computadora.

TCP/IP: Es un protocolo de comunicación entre computadoras que se basa en dos capas. La capa superior es el protocolo TCP: Transmission Control Protocol, que se encarga de descomponer un mensaje o archivo en pequeños paquetes de información para que viajen por la red; cuando esos paquetes llegan a destino son recibidos por otro protocolo TCP que los arma para que vuelvan a ser el mensaje original. La capa inferior es el protocolo IP: Internet Protocol, que maneja el aspecto de direcciones, para que cada paquete llegue a su destino correcto.

Interfaz: Parte de un programa que permite el flujo de información entre un usuario y la aplicación, o entre la aplicación y otros programas o periféricos.

Navegador: es un programa cliente de Internet con el que se conecta un usuario a un servidor Web, permitiendo con esto ver, leer y escuchar información que está en la World Wide Web.

Scripts o Lenguaje de Guiones: Son un conjunto de instrucciones ejecutados por un intérprete, creados usualmente en un archivo de texto.

Interactividad: Es la capacidad que tienen los usuarios de intervenir en el desarrollo de procesos en los que auxilian las aplicaciones informáticas.

Usabilidad: Es un término empleado para denotar la facilidad con que las personas pueden utilizar una herramienta en particular. Usabilidad también puede hacer referencia al método de medida de la usabilidad y el estudio de los principios de la elegancia y efectividad de los objetos.

JAVA: Es un lenguaje de programación diseñado para usar un entorno de computación distribuida de la Internet. Se usa para crear pequeños programas de aplicación para la Internet.

Applet: Son pequeños programas escritos en lenguaje Java, diseñados para ser ejecutados desde internet, se pueden colocar en un servidor, junto con el resto de ficheros que componen un sitio Web

(documentos HTML, ficheros de imagen, sonido, etc.) para lograr múltiples efectos con texto, imágenes, sonidos, etc.

Frame: Es un elemento de HTML, que permite dividir la pantalla en varias áreas independientes unas de otras, y por tanto con contenidos distintos, aunque puedan estar relacionados.

Behaviors: Interacción entre objeto de una página Web; por ejemplo, el cambio en la visualización de elemento de una página cuando se coloca el apuntador del ratón sobre un vínculo.

Aplicaciones RIA: Acrónimo de Rich Internet Applications (Aplicaciones Ricas de Internet). Son aplicaciones con más ventajas que las aplicaciones Web tradicionales.

Primitivas de red: Instrucciones utilizadas en una red para realizar el manejo de información.

Layout: Arreglo, disposición, boceto (de un anuncio); compaginación; ubicación; localización.

MIME: (Multipurpouse Internet Mail Extensions - Extensiones de Correo Internet Multipropósito). Serie de especificaciones dirigidas al intercambio transparente de todo tipo de archivos a través de Internet.

CERN de Suiza: (francés: Conseil Européen pour la Recherche Nucléaire).El Consejo Europeo para la Investigación Nuclear.

SOAP: (Simple Object Access Protocol) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML.

WSDL: Web Services Description Language, Es un protocolo que describe la forma de comunicación e interacción de los servicios Web.