Análisis de la tecnología AJAX y Web 2.0

Autor: Santiago Fiore

Director: Javier Blanqué

Universidad Nacional de Luján

Int. Ruta 5 y 7

6700 Luján, Buenos Aires

República Argentina

Octubre de 2006

Análisis de la tecnología AJAX y Web 2.0

Santiago Fiore
Universidad Nacional de Luján
Int. Ruta 5 y 7
6700 Luján, Buenos Aires
República Argentina
santiagofiore@gmail.com

Resumen

El presente trabajo exhibe un estudio acerca de las tecnologías Web 2.0 y AJAX, con el objetivo de brindar un amplio ámbito conceptual acerca de los distintos aspectos relacionados a las mismas.
Se parte de las reseñas históricas que denotan sus orígenes, detalles de sus características hasta llegar a su aporte a la comunidad tecnológica en la actualidad.
En una segunda instancia, se hace hincapié en el uso de la tecnología AJAX, proponiendo ciertas consideraciones al momento de realizar una aplicación web: ágil, interactiva, liviana, sencilla y segura, teniendo en cuenta algunos entornos de trabajo y librerías mas utilizados, así como también algunas cuestiones relacionadas con el diseño y la seguridad para eludir amenazas.
Se concluye con un ejemplo práctico en tecnología AJAX, basándose en algunos aspectos antes mencionados.

Palabras clave:
AJAX, Javascript, WEB 2.0

Agradecimientos:

INTRODUCCIÓN.

1. FUNDAMENTOS DE AJAX, LA NUEVA CARA PARA LA WEB.

1.1 Surgimiento de AJAX.

1.1.1 Historia de los navegadores.

1.1.2 Tecnologías de Aplicaciones de Internet de Formato Enriquecido.

1.1.2.1 Evolución de la aplicación web.

1.1.2.2 Java Applet.

1.1.2.3 Java WebStart.

1.1.2.4 Macromedia Flash.

1.1.2.5 Macromedia Flex.

1.1.2.6 OpenLaszlo.

1.1.2.7 Mozilla XML User Interface Language (XUL)

1.1.2.8 DHTML.

1.1.3 AJAX.

1.1.3.1 Concepto.

1.1.3.1.1 Conjunto de Tecnologías.

1.1.3.1.2 La Arquitectura.

1.1.3.2 Elementos claves de AJAX

1.1.3.2.1 Asincronía.

1.1.3.2.2 Javascript.

1.1.3.2.2.1 El por qué.

1.1.3.2.2.2 El inicio.

1.1.3.2.3 XML.

2. WEB 2.0.

2.1 Historia.

2.2 Características y concepto.

2.3 Premisas fundamentales.

3. LIBRERÍAS Y FRAMEWORKS.

3.1 Conceptos básicos.

3.2 Librerías y frameworks de Javascript.

3.2.1 Prototype.

3.2.2 Script.aculo.us.

3.2.3 Dojo.

3.2.4 DWR Direct Web Remoting.

3.2.5 jQuery.

3.2.6 Rico.

3.2.7 JSON / JSON-RPC.

3.2.8 Sajax.

3.2.9 Yahoo! User Interface Library.

3.2.10 Google Web Toolkit.

4. PATRONES DE DISEÑO.

4.1 Concepto de patrones.

4.1.1 Patrones de diseño de AJAX.

4.1.1.1 Carga Predictiva (Predictive Fetch).

4.1.1.2 Regulación de Envío (Submission Throttling).

4.1.1.3 Actualización Periódica (Periodic Refresh).

4.1.1.4 Descarga en Múltiples Etapas (Multi-Stage Download).

4.1.1.5 Llamada XMLHttpRequest (XMLHttpRequest Call).

4.1.1.6 Mensaje XML (XML Message).

4.1.1.7 Arrastrar-y-Soltar (Drag-And-Drop).

4.1.1.8 Ventana Emergente (Popup).

4.1.1.9 Tiempo de Espera (Time Out).

5. SEGURIDAD.

5.1 Cuestiones de Seguridad.

5.1.1 Código de Sitio-Entrecruzado (Cross-Site Scripting - XSS).

5.1.2 Aumento en la Superficie de Ataque.

INTRODUCCIÓN.

La investigación para realizar este trabajo parte del estudio del surgimiento de los navegadores web, ya que son la herramienta fundamental del entorno, sin ellos no seria posible la visualización de documentos y ejecución de aplicaciones.
Se realiza un análisis general de las características de las aplicaciones de internet en formato enriquecido y de lo que éstas ofrecen al usuario, profundizando sobre una de ellas: AJAX, teniendo en cuenta la postura de diferentes autores, quienes estudian los distintos elementos claves que forman esta tecnología.
Aparejado a este tema se encuentra el surgimiento del concepto Web 2.0, del cual se detallan sus orígenes y características.
El objetivo es dar una visión general de los conceptos, herramientas que se utilizan y las cuestiones de diseño y seguridad que se deben tener en cuenta al momento de afrontar el desarrollo de una aplicación web.

El trabajo se divide en seis capítulos:

Capitulo 1 Fundamentos de AJAX.
Realiza una breve reseña histórica acerca de los navegadores, además de detallar la evolución de la aplicación web, describiendo las distintas tecnologías de aplicaciones de internet.
Se introduce el concepto de AJAX, desde los puntos de vista de los distintos autores; donde se describe su arquitectura, el conjunto de tecnologías que la conforman y sus elementos claves, marcando relevancia en la Asincronia, el lenguaje Javascript y el lenguaje XML.

Capitulo 2 Web 2.0.
Introduce con un relato histórico conciso acerca del surgimiento de la web 2.0, dando lugar a las comparativas con las web anteriores. Realiza una descripción

del concepto y las características de esta nueva web, así como también las premisas fundamentales que dieron lugar a su evolución.

Capitulo 3 Librerías y Frameworks.
Cubre conceptos básicos acerca de las librerías y los frameworks, que se utilizan en el lenguaje Javascript. Se describen, y en ciertos casos se ejemplifican, algunas de las librerías y los frameworks más populares y de más uso; con el objetivo de conocer el aporte que los mismos brindan al momento de desarrollar una aplicación.

Capitulo 4 Patrones de diseño.
Introduce el concepto básico acerca de los patrones de diseño, explicando el objetivo para el cual son aplicados. Se analizan una cierta cantidad de patrones que ofrecen solución, a problemas comunes, que surgen al momento de desarrollar una aplicación web, teniendo en cuenta sus pros y contras.

Capitulo 5 Seguridad.
Refiere a ciertas consideraciones en cuestión de seguridad en aplicaciones web. Se detallan distintos ataques y vulnerabilidades a tener en cuenta al momento del diseño de una aplicación, de manera de no poner en riesgo en funcionamiento de la misma, ni tampoco la información del usuario final con la cual interactúa.

Capitulo 6 Ejemplo práctico en tecnología AJAX.
Se interpretan en forma práctica, a través de ejemplos, las diferencias existentes entre el modelo sincrónico y el modelo asincrónico.
Se concluye con el desarrollo de una aplicación que incluye los conceptos, consideraciones y cuestiones abordados a lo largo del texto (a nivel de librerías, frameworks, diseño y seguridad).

1. FUNDAMENTOS DE AJAX, LA NUEVA CARA PARA LA WEB.

1.1 Surgimiento de AJAX.

1.1.1 Historia de los navegadores.

Ryan Asleson y Nathaniel Schutta [Asleson-Schutta06], al igual que lo establecido en wikipedia [Wiki01], detallan que: el primer navegador que dio lugar a que el desarrollo de los navegadores web y el de la Web misma sean paralelos fue: WorldWideWeb, desarrollado por: Tim Berners-Lee en el CERN (Organización Europea para la Investigación Nuclear) a finales de 1990 y principios de 1991, este sólo funcionaba en computadoras NeXT (conocidas también como cubos NeXT) que también eran utilizadas como los primeros servidores Web.

Marc Abrams y Shahrooz Feizabadi [Abrams-Feizabadi98] por otra parte explican que Tim Berners-Lee y su equipo en el CERN prepararon el camino para el desarrollo futuro de la web al introducir su servidor y navegador; el protocolo utilizado para la comunicación entre los clientes y el servidor, el Protocolo de Transferencia de Hipertexto (HyperText Transfer Protocol - HTTP); el lenguaje utilizado para elaborar documentos web, Lenguaje de Marcado de Hipertexto (HyperText Markup Language - HTML); y una secuencia de caracteres, utilizada para nombrar recursos, como documentos e imágenes en la web, por su localización, el Localizador Uniforme de Recurso (Uniform Resource Locator - URL).

En un nivel mas detallado sobre lo mencionado, el Consorcio de la World Wide Web (World Wide Web Consortium - W3C) [W3C99] expone que el Lenguaje de Marcado de Hipertexto es el lenguaje que predominante para la construcción de páginas web, dado que se utiliza para describir la estructura y el contenido del documento por medio de etiquetas, de forma tal de complementar el texto con objetos como imágenes. HTML puede describir la apariencia de un documento, al

incluir un script (código ejecutable, por ejemplo Javascript, quien ofrece dinamismo a través de la inclusión de efectos y animaciones), el cual puede afectar el comportamiento de navegadores web.

Aunque hubo varios navegadores, la explosión en popularidad fue en 1993 con Mosaic, desarrollado entre otros por Marc Andreessen en NCSA (Centro Nacional de Aplicaciones de Supercomputación), que inicialmente funcionaba para UNIX y luego lo hizo para plataformas Windows y Macintosh. Con este movimiento, muchas empresas obtuvieron las licencias para crear sus propios navegadores comerciales (entre ellos Spry Mosaic y Spyglass Mosaic).

Por su parte Marc Andreessen, fundo su propia empresa: ”Netscape Communications Corporation”, dando lugar a su navegador Netscape Navigator, el cual presentaba mejoras a nivel usabilidad y confiabilidad con respecto a Mosaic, pudiéndose establecer fuertemente y dominando el mercado de navegadores.
Sin ningún producto en el mercado Microsoft Corporation inmediatamente lanzo su navegador: el Internet Explorer (ex Spyglass Mosaic), que sólo ofrecía soporte para lenguajes interpretados (scripting) y la primera implementación comercial de: las hojas de estilo en cascada (CSS). De esta manera se dio inicio a lo que se conoció como: la batalla de los navegadores, en la cual tras varios años de pelea, Microsoft, consiguió destronar a Netscape del negocio de los navegadores distribuyendo su navegador de forma gratuita.
Netscape Corporation libera el código fuente de su navegador, y da a luz a: Mozilla Foundation junto al proyecto: Mozilla, que fue reescrito completamente para poder usar como base nuevos widgets multiplataforma, basados en XML (conocidos como XUL) consiguiéndose así un navegador de gran calidad que corriera en varias plataformas.
Uno de los navegadores derivados de este proyecto es: Mozilla Firefox, que ofrece muchas mejoras y se presenta como más ligero con respecto al navegador que le dio origen.

Si bien omitimos comentar en detalle los distintos navegadores que se han creado a lo largo de la historia, hoy en día los navegadores más importantes que mantienen la competencia viva son: Internet Explorer y Mozilla Firefox ofreciendo mejoras, funcionalidades, estándares web y el soporte a las extensiones.
Durante el 2008 verán la luz las versiones 3 de Firefox y 8 de Internet Explorer.
Otros navegadores utilizados son Opera, Konqueror, Safari y Camino para la plataforma Macintosh.
A continuación una tabla comparativa de navegadores y las plataformas que estos soportan:

Navegador / Plataforma	Windows	Linux-Unix	Mac OS X
Internet Explorer	**x**		**x**
Mozilla Firefox	**x**	**x**	**x**
Opera	**x**	**x**	**x**
Mozilla Seamonkey	**x**	**x**	**x**
Mozilla Camino			**x**
Safari	**x**		**x**
Konqueror		**x**	

1.1.2 Tecnologías de Aplicaciones de Internet de Formato Enriquecido.

Para poder entender el mundo de AJAX es necesario primordialmente conocer su historia, sus orígenes y cómo es que se ha ido incursionando acorde pasa el tiempo en las maneras de programar sitios web dinámicos con las herramientas que esta tecnología usa y las aplicaciones de entorno web que con ésta se consigue.

Para ello, nos debemos remontar hasta el punto donde surgen las: “Aplicaciones de Internet de Formato Enriquecido” o “AIE” de ahora en adelante (del ingles Rich Internet Application o RIA) y qué se entienden por ellas. Pero antes de empezar a explicarlas, se debe hacer una escala en el concepto que les dio origen, la “Experiencia de Usuario en Formato Enriquecido” (del ingles Rich User

Experience), para poder entender de esta manera, las razones por las cuales surge este nuevo mundo de AJAX.

Sang Shin [Shin-Ajax07] explica que cuando se habla de la “Experiencia de Usuario en Formato Enriquecido”, uno debe tener en mente como es el funcionamiento de una típica aplicación de escritorio, cuando uno utiliza una aplicación se espera que ésta responda a lo que se le solicito de manera rápida y casi intuitiva, como un viaje de ida y vuelta, se envía una orden, se recibe una respuesta de inmediato. Por ejemplo cuando nos paramos con el puntero del mouse sobre una carpeta y se despliega una pequeña descripción con los archivos que este contiene con breve resumen de los mismos (tipo, tamaño, carpetas, fechas y etc).

Al usar las aplicaciones de escritorio se puede apreciar que los eventos y acciones ocurren con mayor naturalidad, muchas veces no es necesario hacer click sobre algún botón para tener un resultado o llevar a cabo determinada tarea, aún solo con el gesto o movimiento del mouse se pueden obtener resultados que se desean.

La experiencia de usuario en formato enriquecido está ligada al uso que él hace de las nuevas tecnologías en aplicaciones web, siendo éstas colectivamente llamadas AIE.

Las AIE son aplicaciones web que presentan las características y funcionalidades de las tradicionales aplicaciones de escritorio. Éstas transfieren el procesamiento necesario para la interfaz de usuario al cliente web, pero dejan la carga de datos en segundo plano en el servidor de aplicaciones [Wiki02]. (Ej.: el mantenimiento del estado del programa, el dato, etc.)

1.1.2.1 Evolución de la aplicación web.

Las AIE establecen un antes y un después en la programación de sitios web. Las aplicaciones web convencionales presentan marcadas características, así como lo establece Sang Shin [Shin-Ajax07]:

* Toda su actividad está centrada alrededor de una arquitectura: cliente-servidor, con una débil participación del cliente, donde todo el procesamiento está hecho del lado del servidor, y del lado del cliente se limita a mostrar páginas de contenido estático [Wiki02].
* La interacción del usuario era, básicamente, un ciclo repetido de pasos “click, esperar, actualizar”, esto es, la totalidad de la página se actualizaba si existía algún evento o interacción hacia o desde el servidor, ya sea si se enviaba algún dato al servidor o se recibía algún dato del mismo.
* El usuario debía esperar hasta que la respuesta llegara por parte del servidor, la cual se podía ver afectaba por el tráfico en la red haciendo el proceso más lento, esto se conoce como: interacción sincrónica.
* El flujo de las páginas web convencionales es: “page-driven” (del ingles conducido por páginas), la aplicación se basaba en varias páginas que iban cambiando a la siguiente según lo determinaba el servidor.

Para Sang Shin [Shin-Ajax07] el comportamiento habitual de una aplicación web, muestra varias limitaciones, por ejemplo:

* Es lento, tanto su funcionamiento como en su tiempo de respuesta, ya que el usuario no puede realizar ninguna acción hasta que reciba una respuesta.
* Pierde el contexto operativo de la web durante el tiempo de actualización, dado que toda la página debe ser actualizada. Esto se refiere a pérdida del puntero del mouse, pérdida de la posición en la cual uno se encontraba (ya

que cuando la página es actualizada se debe recuperar la posición anterior) y pérdida de la información de la página, la cual se obtendrá al momento que se tenga la respuesta por parte del servidor.

* Y las restricciones propias de las aplicaciones básicas, las cuales carecen de “widget” (windows gadget, del ingles dispositivos de ventanas).

Debido a las limitaciones de las web convencionales y a la necesidad de realimentación en tiempo real de sus acciones, por parte de los usuarios, es que las AIE fueron ganando protagonismo.

Según Yakov Fain [Fain07] las AIE han restaurado el poder de las aplicaciones de escritorio, dentro del entorno de la descarga de una página web, ya que éstas se ejecutan en una máquina virtual y tienen el potencial de convertirse en una aplicación de escritorio con todas sus características, Yakov Fain [Fain07], adhiere y hace notar el funcionamiento actual de las AIE respecto a sus comienzos, donde sus características se reducían a simplemente mostrar una página web proveniente de un servidor, mientras que en la actualidad al ejecutarse del lado del cliente permite realizar tareas de manipulación de datos localmente.

Las AIE no requieren que la totalidad de la página sea actualizada para que se actualicen sus datos, ya que su tiempo de respuesta es mucho más rápido y el de recarga es mucho mas bajo.

Para Vaibhav Gadge [Gadge06] una ventaja significativa es que las AIE actualizan un bloque de la página y no la página entera como las aplicaciones convencionales, que dependían directamente de la inteligencia del servidor, al otro extremo de la red. Al actualizar un bloque la aplicación se hace más rápida, más usable y ofrece una mejor experiencia visual para los usuarios.

Cameron O'Rourke [O'Rourke04] dice que se pueden presentar interfaces gráficas de usuario de respuestas ágiles y enriquecedoras.

Cameron O'Rourke [O'Rourke04], agrega que las AIE utilizan robustos motores del lado del cliente, permitiendo que los datos puedan ser almacenados en el cache del mismo, de manera de presentar una interfaz de usuario de respuesta mas ágil, con pocas idas y vueltas al servidor como si se hace con una interfaz visual HTML normal, Yakov Fain [Fain07] coincide con éstos beneficios.

También coinciden en que las AIE eliminan el ida y vuelta de las páginas y mejoran sustancialmente la performance de las mismas haciendo mucho del procesamiento del lado del cliente, mas que las aplicaciones web convencionales.

Como para sustentar su punto, Yakov Fain [Fain07], agrega que una AIE combina los beneficios de usar la web como un modelo de despliegue de bajo costo, con una experiencia de usuario en formato enriquecido que es tan buena como una aplicación de escritorio actual, esto mismo también es contemplado por Cameron O'Rourke [O'Rourke04].

A continuación, la figura 1 muestra el resumen descriptivo de la arquitectura típica de una AIE:

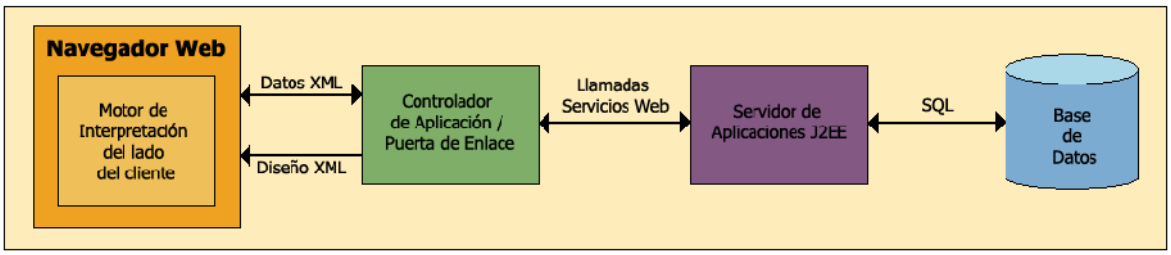


Figura 1: Típica Arquitectura de una Aplicación de Internet de Formato Enriquecido [FIG1].

Cameron O'Rourke [O'Rourke04] describe que el XML se utiliza generalmente como un formato para la transferencia de datos y a veces suele describir

formularios plantilla (que se definen con un cierto estilo). En muchos casos, el cliente puede permanecer conectado con la fuente de datos, de esta manera el servidor puede actualizarlo en tiempo real. El acceso a una base de datos se logra con llamadas a Servicios Web (también conocidos en ingles como Web Services).

Algunos ejemplos de las tecnologías de AIE son: Java Applet, Java WebStart, Macromedia Flash, Macromedia Flex, OpenLaszlo, Mozilla XML User Interface Language (XUL), DHTML y AJAX las cuales se describirán a continuación detallando sus ventajas y desventajas que presentan:

1.1.2.2 Java Applet.

Para Sang Shin [Shin-Ajax07], el Java Applet es una aplicación que es descargada y ejecutada dentro del contexto del explorador, al ser una aplicación propia de Java, puede utilizar APIs completas del mismo lenguaje, y hay ciertas acciones que pueden hacer, como: la adaptación del flujo de datos (custom data streaming), manipulación de gráficos y operaciones avanzadas de interfaz de usuario, las cuales no pueden realizarse con otras aplicaciones de este tipo. Además agrega que la tecnología Applet ha estado rondando desde el comienzo de la tecnología Java y que tiene un esquema bien establecido. Por otra parte describe que la desventaja de este, es que el tiempo de descarga del código puede ser significativo (dado su tamaño expresado en bytes), especialmente cuando debe ser descargado cada vez que un usuario lo accede. Yakov Fain [Fain07] agrega a esta falla que los applets no ofrecen descargar la versión apropiada del JVM junto con el applet, mientras que tecnologías como: Flash, en su instalación express, si lo hacen.

Para Nancy Cluts [Cluts98] si bien los applets al estar escritos en lenguaje Java, proveen más funcionalidad que el HTML y el script, a su vez proveen

compatibilidad limitada de plataforma y navegador, estas limitaciones son las que dificultan el escribir el applet una vez y para que se ejecute donde sea.

Nancy Cluts [Cluts98] aporta a su comentario el dato que los applets son utilizados en menos del 1% del total de las páginas web.

Como recomendación, Sang Shin [Shin-Ajax07] propone "*usar applets si se está creando interfaces de usuario avanzadas levantando APIs completas de Java y el tiempo de descarga del código no es tema de importancia".*

1.1.2.3 Java WebStart.

Con la tecnología Java WebStart, Sang Shin [Shin-Ajax07] explica que ésta permite levantar una aplicación Java (sea de escritorio o un Applet) a través de la red.
Sang Shin [Shin-Ajax07] comenta las características de un Applet asociado con esta tecnología diciendo: que la fortaleza de un Applet es su modelo de distribución en la cual un usuario actualiza automáticamente el código, y su debilidad con respecto a la aplicación de escritorio, es que para poder utilizar un Applet se debe tener una conexión de red.

Así mismo, detalla la aplicación de escritorio Java diciendo que: la fortaleza de una aplicación de escritorio es que el usuario puede hacer uso de ella en modo desconectado, mientras que su debilidad queda expuesta ya que ésta debe ser instalada manualmente en cada cliente que la use, salvo que se use Java Web Start.

Raghavan Srinivas [Srinivas01] asiente con la característica de la disponibilidad de trabajar desconectado, dado que una aplicación puede ser utilizada en situaciones donde el uso del navegador no es posible.

Como conclusión de estas comparaciones Sang Shin [Shin-Ajax07], establece que al usar la tecnología Java WebStart, se puede instalar una aplicación de escritorio como si fuera un Applet, una vez instalada la aplicación Java funciona como si fuera una aplicación de escritorio independiente. Raghavan Srinivas [Srinivas01] se suma diciendo que las aplicaciones sean ejecutadas independientemente del navegador, por ejemplo un atajo de escritorio, haciendo ejecutar la aplicación Web de forma similar a una aplicación nativa.

Sang Shin [Shin-Ajax07] además agrega que cada vez que se usa la aplicación, se chequea si existe una nueva versión.

La desventaja que tiene Java WebStart es que el viejo sistema base JRE (que comprende JDK 1.1 y menores) no funciona con esta tecnología, ya que se introdujo con el JDK 1.2 y la primera vez que se descarga el tiempo de la misma puede ser significativo (dado su tamaño, al igual que ocurre con un Java Applet).

1.1.2.4 Macromedia[1] Flash.

Otra de las tecnologías que expone Sang Shin [Shin-Ajax07] es la de Macromedia Flash. Flash fue diseñado para reproducir películas interactivas y tiene su propio lenguaje conocido como ActionScript. La implementación de ejemplos de la tecnología Flash es hecha por Macromedia Flex (código propietario) o Laszlo (basado en código abierto).

Por su parte, Marc Domenig [Domenig05], detalla que la base de la tecnología Flash incluye un entorno de ejecución (el Flash player), un archivo de formato binario para películas (SWF), y un lenguaje de scripting (ActionScript), cómodas

[1] Macromedia hasta el año 2005, posteriormente Adobe.

herramientas de usuario final que generan SWF y soportan programación ActionScript están disponibles para el diseño de películas y sitios Web.

Para Sang Shin [Shin-Ajax07] lo admirable de Flash es que es muy bueno mostrando gráficos de vectores, su contra es que el navegador debe tener un plugin de Flash para poder ejecutarse; y otra contra más es que ActionScript es una tecnología propietaria.

Para Cameron O'Rourke [O'Rourke04] la primera desventaja que tiene Flash es que tiene soporte limitado para XML y para los estándares de Servicios Web, y la inmadurez del ambiente como una herramienta de desarrollo de aplicación. La primera ventaja es la facilidad con la que se pueden crear animaciones complejas y la posibilidad de complementos (add-ons) por parte de terceros.

Para Marc Domenig [Domenig05], los componentes actuales de Flash no soportan una arquitectura pura de cliente liviano, (del ingles thin-client architecture, se refiere a una arquitectura donde del lado del cliente solo se encuentra un motor de presentación y donde se lleva a cabo la ejecución del código; y del lado del servidor se encuentra la lógica de presentación, la lógica de negocio y los datos), ya que originalmente no fueron diseñados para tal fin.

1.1.2.5 Macromedia Flex.

Para Vaibhav Gadge [Gadge06], la tecnología Macromedia Flex es una interfaz de usuario basada en Flash. Provee un servidor de presentación Flex que corre sobre un servidor de aplicaciones y genera los archivos Flash dinámicamente desde el servidor y los envía al navegador. Esos archivos Flash son ejecutados dentro del Flash player de un navegador, y permite interactuar con usuarios, llevar a cabo operaciones y solicitudes de conectividad hacia el servidor.

Cameron O'Rourke [O'Rourke04], comenta que ActionScript se utiliza para escribir MXML, de manera que las interfaces de usuario puedan ser compiladas e interpretadas al vuelo por el Flash Player; a lo que Vaibhav Gadge [Gadge06] adiciona que esto es para manejar eventos de usuario, eventos del sistema, o construir complejo modelo de datos (presentación de datos, validación de datos y servicio de datos de forma separada). En mas detalle, Bruce Eckel [Eckel07] y Vaibhav Gadge [Gadge06], coinciden que las aplicaciones Flex se compilan directamente en archivos SWFs (binarios de Flash), los cuales luego son compilados con la técnica “justo a tiempo” (Just-In-Time, JIT) en tiempo de ejecución de Flash, para mayor velocidad.

Vaibhav Gadge [Gadge06] detalla que Flex provee una enriquecedora y extensa librería de clases MXML para componentes visuales, contenedores y servicios de objetos remotos, y acceso a datos del lado del servidor e interacción con Servicios Web.

Para Cameron O'Rourke [O'Rourke04] la principal desventaja, tanto de Flash como de Flex, es el soporte limitado para XML y estándares de Servicios Web y la relativa inmadurez del entorno como una herramienta de desarrollo de aplicación.

La principal ventaja de Flash y Flex es la facilidad para crear muestras animadas complejas y la disponibilidad de agregados o mejoras desarrollados por terceros.

1.1.2.6 OpenLaszlo.

La tecnología OpenLazlo presenta una plataforma que consiste en el lenguaje de programación LZX y el servidor OpenLazlo (compilador de LZX) [Wiki03].

LZX es un lenguaje orientado a objetos, basado en etiquetas, que hace uso de XML y Javascript, y se utiliza para escribir aplicaciones clientes, para crear

dinámicamente archivos Flash, comenta Vaibhav Gadge [Gadge06], y crear aplicaciones que pueden ser desplegadas tanto como DHTML o películas de Flash, según la opinión de Yakov Fain [Fain07]. Esas características, explica William Grosso [Grosso05], son las que se llevan a cabo en el servidor, quien compila aplicaciones (o archivos) LZX en películas Flash, las cuales son luego enviadas a un explorador web.

Vaibhav Gadge [Gadge06] coincide con la opinión de Grosso sobre el comportamiento del compilador de LZX, pero además agrega que el intercambio de datos actual es en formularios XML, y los controles LZX usan Xpath para referirse al XML.

Una de las características que menciona Vaibhav Gadge [Gadge06] es que LZX tiene la capacidad de hacer solicitudes del tipo http y a Servicios Web al servidor, en segundo plano, sin actualizar la página. Esto suma a que solo se necesita tener instalado el Flash Player en un explorador para que OpenLazlo funcione.

De esta manera se establece una competencia directa con Flash, ya que OpenLazlo está definido bajo licencia de código abierto, mientras que Flash ofrece su producto con una licencia no libre (es decir, comercial).

1.1.2.7 Mozilla XML User Interface Language (XUL).

La siguiente tecnología pertenece al proyecto Mozilla, Cameron O'Rourke [O'Rourke04] explica que XUL es un lenguaje de interfaz de usuario basado en XML. Se utiliza para crear la interfaz de usuario de aplicaciones que se ejecutan en navegadores Mozilla, así como también, en otros motores de interpretación por ejemplo Zulu (un componente de Flash MX) y Thinlets (una implementación de Java).

Los motores de interpretación para XUL son muy livianos (inferior a los 100 KB) y pueden consumir y producir datos XML. Vaibhav Gadge [Gadge06] agrega, mas específicamente que XUL describe controles de interfaces de usuario, además de proveer todas las clases de los controles populares de la Internet enriquecida, por ejemplo: menús, estructuras de árboles, menús pop-up (menús desplegables).

XUL utiliza DOM para guardar el árbol de nodos, una vez que los archivos XUL son cargados, luego analiza (parsea) y convierte todas las etiquetas en una estructura de jerarquía de nodos, para que por medio de métodos que proveen las funciones XUL se pueda examinar y modificar la estructura DOM.

Para Cameron O'Rourke [O'Rourke04], la principal desventaja de XUL es que no está respaldada por una importante entidad comercial; y sus principales ventajas son: su integración con el motor Gecko, el motor de navegación de Mozilla, (amplio acceso a un enorme arreglo de estándares Web), y el hecho que XUL es un lenguaje muy expresivo y compacto, comparado con muchos otros lenguajes XML de descripción de interfaz de usuario.

1.1.2.8 DHTML.

El HTML Dinámico o DHTML, como indica Sang Shin [Shin-Ajax07], consiste básicamente en una combinación de Javascript, DOM y CSS. DHTML ha sido utilizada para crear aplicaciones Web interactivas y de respuesta ágil (con un mayor nivel de respuesta).

Para Nancy Cluts [Cluts98] DHTML permite interactividad en páginas HTML estáticas. Con DHTML, se puede manipular cualquier elemento de una página, cambiar estilos y colores, y la posición de los mismos “al vuelo” (pero no cuando la página ya está cargada). DHTML incluye un Modelo de Objetos de Documento para permitir la programabilidad, la capacidad de manipular los datos del lado del

cliente, y el soporte para scriptlets, que son fragmentos de HTML reusables y scripts que pueden ser utilizados para mostrar el contenido interactivo.

Aún así Sang Shin [Shin-Ajax07] manifiesta que las DHTML, por si solas, no proveen comunicación asíncrona, lo cual significa que se requiere que la totalidad de la página sea actualizada, concluyendo que esa es la razón por la que las DHTML tienen éxito limitado.

1.1.3 AJAX.

Los conceptos sobre AJAX que se detallarán a continuación, fueron elaborados a partir de distintos puntos de vista que ofrecen las máximas autoridades del campo, especializadas en desarrollos AJAX. Por supuesto que existen contradicciones, pero esto se debe a que para muchos autores la tecnología AJAX es nueva, podría decirse que "en construcción", y no existe en la actualidad, una visión clara y homogénea acerca de ésta.
Trataremos de dar una descripción crítica y detallada de esta tecnología y de mostrar los conceptos donde hay contradicciones por parte de los distintos autores.

1.1.3.1 Concepto.

Para Sang Shin [Shin-Ajax07] AJAX debe pensarse como DHTML, al cual se le agrega la capacidad de comunicación asincrónica a través del objeto XMLHttpRequest de Javascript. Siendo una de las tecnologías AIE mas viables del momento, por la industria a la que dio lugar, existen para ella varias herramientas (toolkits) y frameworks (estructuras de desarrollo de software), además de esto no hay necesidad de descargar algún plugin para que esta tecnología funcione; en desventaja a lo antes establecido, existen incompatibilidades con algunos

navegadores, el código Javascript no es de fácil mantenimiento y depuración (también conocido como debugging).

Es el caso de Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05], quienes buscan definir que es AJAX, y para esto dan 2 significados distintos (o visiones distintas):

* AJAX puede ser visto como un conjunto de tecnologías.
* AJAX puede ser visto como una arquitectura.

1.1.3.1.1 Conjunto de Tecnologías.

Desde el punto de vista de James Garrett [Garrett05] AJAX no es una tecnología, sino que en realidad son varias tecnologías, cada una de las cuales prosperan por derecho propio, juntándose en nuevos y poderosos caminos. Agrega que AJAX incorpora:

* Presentaciones en base a estándares usando XHTML y hojas de estilo (CSS Cascading Style Sheets)
* Interacción y muestra dinámica utilizando Modelo del Objeto del Documento (DOM Document Object Model)
* Intercambio y manipulación de datos utilizando XML y XSTL
* Recuperación de datos de forma asincrónica utilizando el objeto XMLHttpRequest
* Y Javascript que une todo en su conjunto.

Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05] coinciden plenamente en este pensamiento.

Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06], consideran que AJAX nació en Febrero del 2005, resguardándose en el ensayo que James Garrett [Garrett05] utiliza para definir a este conjunto de tecnologías. Y que todas estas tecnologías mencionadas por Garrett (*HTML / XHTML, CSS, DOM, XML, XSLT, XMLHttp* y *Javascript*) están disponibles en una solución AJAX, siendo requeridas solo 3: HTML / XHTML, DOM, y Javascript. XHTML se necesita para mostrar la información, DOM se necesita para cambiar una parte de una página XHTML sin tener que recargarla completamente, y Javascript se necesita para iniciar una comunicación cliente-servidor, para luego poder manipular el DOM y poder así actualizar la página. Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] creen que las otras tecnologías de la lista sirven de forma complementaria en una solución AJAX pero no son necesarias.

Sin embargo, desde que James Garrett [Garrett05] publicó su ensayo en Marzo del 2005 ha recibido una gran cantidad de preguntas, entre las cuales consideró una de ellas en especial que genero mayor atención para entender cada una de las tecnologías que conforman a AJAX, se preguntó el porque de la necesidad de darle ese nombre, y la respuesta inmediata fue la necesidad de ofrecer una palabra corta que se refiera a *"Asincronía + Javascript + CSS + DOM + XMLHttpRequest"* al momento de conversar con los clientes. Siendo el lenguaje ***Javascript*** y el lenguaje ***XML*** por medio de la ***Asincronía***, los que forman el fuerte de AJAX, los cuales se detallaran a lo largo del texto.

1.1.3.1.2 La Arquitectura.

Según Gehtland, Galbraith y Almaer [Gehtland-Galbraith-Almaer05], la interesante evolución que provocan los conceptos incluidos en AJAX, esta en como se puede construir una aplicación web, para hacer notar el cambio que trajo consigo AJAX, detallan primero algunas de las características de la arquitectura convencional de la web, donde se define una página por cada evento en la aplicación, a su vez

cada evento o acción retorna una página completa al navegador y ésta ultima es la interpretada o visualizada por el usuario. Agregan que la web comenzó siendo un gran repositorio de documentos, donde cada uno de ellos se entrelazaba con otros, de manera que se compartía datos y documentos, no existía tanta interactividad como ahora.

Esto concuerda con lo dicho por, Jesse James Garrett [Garrett05] donde explica que el modelo clásico de la aplicación web trabaja de la siguiente manera (figura 2): "*muchas de las acciones de los usuarios en la interfaz apuntan a una solicitud HTTP que vuelva del servidor Web. El servidor hace algún proceso y luego retorna una página HTML al cliente. Apoyando lo comentado en los puntos anteriores expone que, mientras el servidor está procesando , ¿qué es lo que hace el usuario? Esperar y por cada tarea que realiza el usuario debe esperar un tiempo a veces significativo.* "

Esto es así dado que la naturaleza de la interacción de esta web es empezar-parar-empezar-parar (lo que demuestra la sincronía figura 3). A continuación la gráfica que muestra el modelo clásico de una aplicación web:

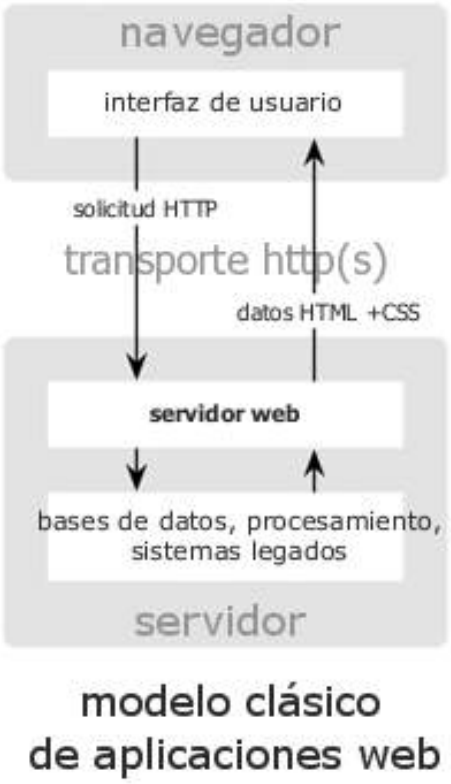


Figura 2: Modelo tradicional de una aplicación Web [FIG2].

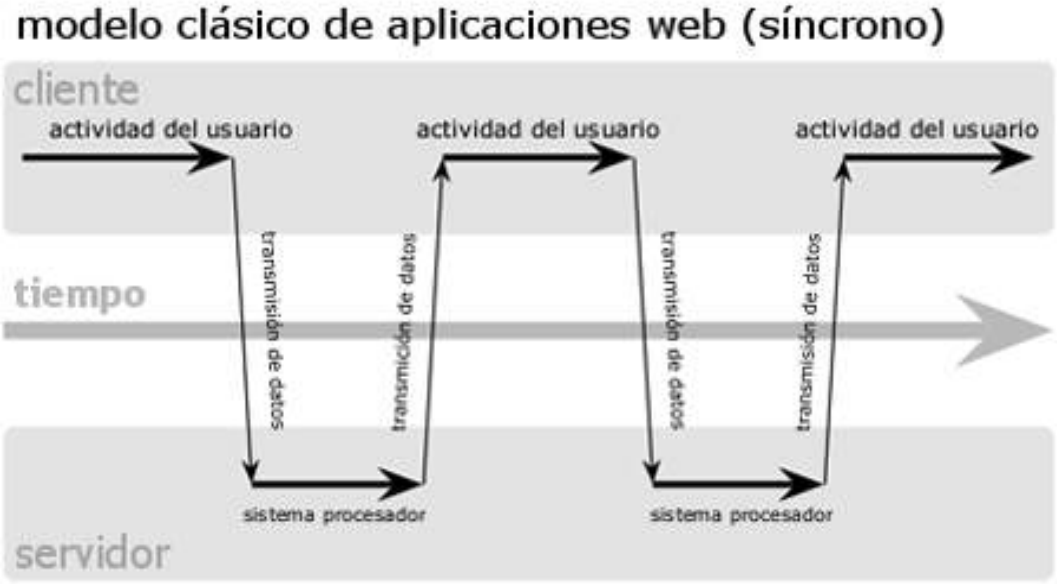


Figura 3: La interacción sincrónica de una aplicación Web tradicional [FIG3].

(Ver Anexo A.1)

Gehtland, Galbraith y Almaer [Gehtland-Galbraith-Almaer05], sostienen que AJAX permitió una nueva arquitectura, las partes más importantes son:

* *Pequeños eventos del lado de Servidor:* hoy los componentes de una aplicación web pueden realizar pequeñas peticiones al servidor, para traer algo de información y actualiza una porción de la página que está siendo vista, no se recarga la página entera.
* *Asincronía:* las peticiones enviadas al servidor no causan el bloqueo del navegador hasta que este pueda dar una respuesta, sino que permiten que el usuario pueda seguir utilizando otras partes de la aplicación, mientras la interfaz de usuario es actualizada dando luego aviso en algunos casos al usuario que la petición ha sido cumplida.
* *Eventos e Interactividad:* los navegadores nuevos pueden capturar muchos de los eventos que el usuario utiliza comúnmente cuando interactúa con un sistema operativo, algunos de los eventos que desata el usuario pueden disparar una petición asincrónica.

1.1.3.2 Elementos claves de AJAX.

1.1.3.2.1 Asincronía.

En su ensayo James Garrett [Garrett05] explica que una aplicación AJAX elimina la naturaleza comenzar-parar de la interacción de la web, introduciendo un intermediario (un motor AJAX según Garrett) entre el usuario y el servidor, y que pareciera que al agregar una capa mas a la aplicación, ésta se hiciera menos ágil en sus respuestas, pero ocurre lo contrario.

En lugar de cargar una página web, al comienzo de una sesión, el navegador carga un motor AJAX, escrito en Javascript y generalmente camuflado en un frame oculto.

Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] expresan la diferencias entre el modelo tradicional de una aplicación web, donde el navegador era responsable de iniciar las peticiones hacia, y procesar las mismas desde el servidor web; y el modelo de una aplicación AJAX, citando lo que en su ensayo Garrett llama el motor AJAX, lo que para ellos consideran que en realidad es solo un objeto o función Javascript que se invoca cada vez que la información es solicitada desde el servidor.

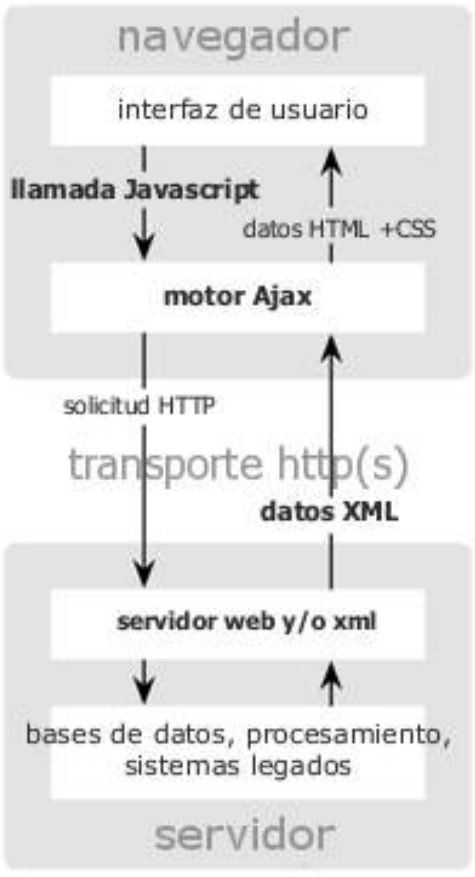
Continuando con la explicación James Garrett [Garrett05] expresa que el motor AJAX es responsable de dos tareas simultáneamente, por un lado la interpretación de la interfase que el usuario visualiza; y por otro lado la comunicación con el servidor por pedido de usuario.

El motor AJAX permite que la interacción del usuario con la aplicación ocurra asincrónicamente, con esto se refiere a la comunicación independiente con el servidor; por lo que de esta manera el usuario generalmente no visualizara una ventana blanca en el navegador, a la espera de respuesta de parte del servidor.

En la figura 4, y en consecuencia la figura 5, James Garrett [Garrett05] permite visualizar el funcionamiento asincrónico que ofrece el motor AJAX, cada acción de usuario normalmente generaría una petición HTTP, pero en este modelo la acción se convierte en una llamada Javascript al motor AJAX, y cualquier respuesta a la acción de usuario no requiere una nueva llamada al servidor (por ejemplo una validación o la edición de datos de la página), sino que el motor se encarga de eso; si el motor necesita alguna respuesta por parte del servidor (por ejemplo el envió de un formulario con datos para su procesamiento o cargar alguna nueva página), el motor mismo es quien hace las peticiones asincrónicamente (es decir

que la ejecución del código no espera una respuesta de un servidor remoto para continuar), utilizando para esto XML.

Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] coinciden con Garrett y además agregan a esto último que el único requisito para el motor es poder interpretar y entender los datos (sea texto plano o XML), ya que cuando este recibe la respuesta desde el servidor, ésta da lugar a una acción, la cual sufre varios cambios hasta llegar a la interfaz de usuario final con la información que le ha sido provista. Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] concluyen que, dado que este proceso involucra menor transferencia de información que el modelo tradicional, la interfaz de usuario se actualiza más rápido.



modelo Ajax de aplicaciones web

Figura 4: Modelo AJAX de una aplicación Web [FIG4].

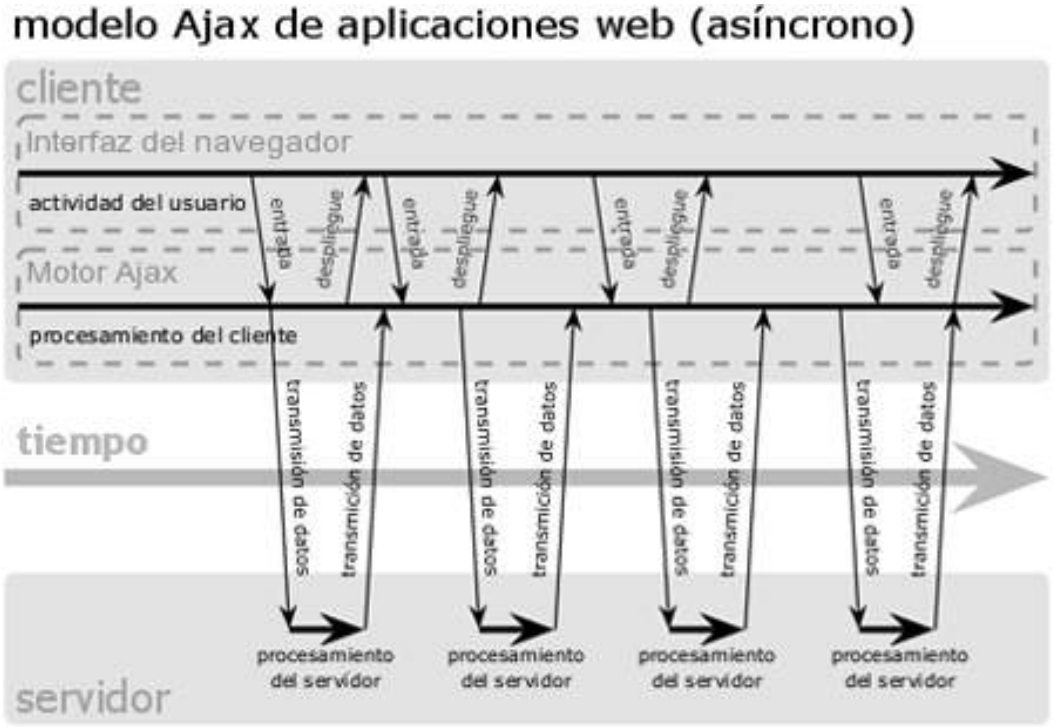


Figura 5: La interacción asincrónica de una aplicación AJAX [FIG5].

(Ver Anexo A.2)

1.1.3.2.2 Javascript.

1.1.3.2.2.1 El por qué.

Haciendo un poco de historia Christian Heilmann [Heilmann06] comenta que en el comienzo de la web, existía HTML y CGI (Common Gateway Interface, en español Interfaz de Puente Común). Puesto así, el HTML define las partes de un documento de texto e instruye al navegador como mostrarlo, por lo que si se quiere modificar algo en el documento o usar los datos que el usuario haya ingresado en el mismo, se necesita hacer un viaje de ida y vuelta al servidor.

Con este ejemplo lo que Heilmann [Heilmann06] quiere explicar es que utilizando tecnología dinámica (ASP, PHP o JSP) se puede enviar información desde los formularios o desde los parámetros, al servidor, quien luego realizara cálculos,

testeos, acciones con la base de datos, etc. El servidor de aplicaciones asociado con esas tecnologías, escribe un documento HTML para mostrar los resultados, quien es luego retornado al navegador para que sea visible. Explicado ya este funcionamiento, es aquí donde radica el problema, dado que cada vez que existe un cambio, el proceso completo antes descripto debe ser repetido, por lo que la página es recargada, lo cual lo hace muy incomodo y lento.

Christian Heilmann [Heilmann06] nota que dada la necesidad de una solución que permita a los desarrolladores Web dar una respuesta inmediata al usuario y cambiar una página HTML sin tener que recargar la misma desde el servidor, es que da lugar a Javascript.

Cualquier información, ya sea tanto algún calculo como la verificación de datos de un formulario, no necesariamente viene del servidor, dado que Javascript es ejecutado por el navegador del lado del cliente (o usuario) pudiendo llevar a cabo estas acciones.

1.1.3.2.2.2 El inicio.

Christian Heilmann [Heilmann06] relata que Netscape creó el lenguaje Javascript en 1996 (mas específicamente en Diciembre de 1995 según Danny Goodman [Goodman01] quien coincide en la historia); y lo incluyo en el navegador Netscape Navigator 2.0 por medio de un interprete que podía leer y ejecutar código Javascript que se agregaba a las páginas HTML.

Además agrega que Javascript comenzó siendo Livescript, pero Netscape luego le cambio el nombre, aún cuando no existe una conexión real entre Java y Javascript. A esto ultimo hace alusión Danny Goodman [Goodman01] dado que Netscape tenia muy buenas razones de marketing para adoptar el nombre Javascript, las cuales se fundamentaban en la popularidad que estaba tomando en

ese entonces el lenguaje Java, aún cuando ambos lenguajes eran muy diferentes entre si.

Para Danny Goodman [Goodman01], Javascript sirve para cubrir 2 propósitos con la misma sintaxis. Uno de ellos es como lenguaje de scripting, que los administradores de servidores Web pueden utilizar para manejar el servidor y conectar sus páginas a otros servicios. El otro propósito es del lado del cliente, dado que las funciones escritas en Javascript se realizan en las páginas Web de muchas maneras (por ejemplo que la información ingresada por un usuario sea válida, entre otras cosas, sin tener que enviar la información hasta el servidor para ser validada).

Por otra parte Christian Heilmann [Heilmann06] explica, dado que la popularidad de Javascript ha crecido constantemente, ahora es soportado por la mayoría de los navegadores modernos. Existen diferencias en la manera que los diferentes navegadores implementan Javascript aún cuando la base del lenguaje es la misma.

Una de las ventajas de Javascript es que es más fácil desarrollar en él, en comparación con lenguajes de programación de alto nivel compilables (como C) o lenguajes de scripting del lado del servidor (como Perl); ya que no se necesita ninguna compilación y no es ejecutado como un módulo de un servidor. Todo lo que se necesita para escribir, ejecutar, y depurar Javascript es un editor de texto y un navegador. Aunque todavía las herramientas de desarrollo de ciclo completo para Javascript no están del todo desarrolladas, de manera que depurar un programa realizado en Javascript a veces tiene sus contratiempos.

Los meritos que se le atribuyen a Javascript según Christian Heilmann [Heilmann06] y en los cuales se encuentran coincidencias con los mencionados por Danny Goodman [Goodman01] son:

* *Menor interacción con el servidor:* dado que el usuario valida lo que ingresa antes que la página sea enviada al servidor, esto ayuda a reducir tráfico de mensajes en la red y carga de trabajo en el servidor.
* *Respuesta inmediata hacia el usuario:* el usuario no deberá esperar a que la página se recargue para verificar si ha ocurrido algún olvido al ingresar algo.
* *Incremento en la usabilidad permitiendo que el usuario cambie e interactué con la interfaz de usuario sin tener que recargar la página.*
* *Incremento en la interactividad:* crea interfaces que reaccionan cuando el usuario las activa disparando un evento vía mouse o teclado.
* *Interfaces enriquecidas:* incluyen componentes arrastrar-y-soltar, efectos persianas o diapositivas, así como también contenidos y estilos.
* *Entorno Liviano:* permite descargar archivos de pequeño tamaño los cuales se obtienen del cache del navegador una vez que se hayan cargado.
* *Preprocesar los datos del usuario antes que sean enviados al servidor*

1.1.3.2.3 XML.

Para poder comprender el momento en que toma parte XML, se debe hacer una breve reseña histórica, para ello David Hunter et al [Hunter et al 04] explican que en un principio, se planteaba la idea de un formato de datos universal que combinara las ventajas de los archivos binarios, por ejemplo la eficiencia y la capacidad de almacenamiento de información; con las ventajas de los archivos de texto, siendo estos universalmente intercambiables, de manera de poder llevar a cabo el intercambio de información entre diferentes programas. Hunter et al [Hunter et al 04] expresan que el primer intento fue SGML (del ingles Standard Generalized Markup Language - Lenguaje de Marcas Estándar Generalizado) que fue diseñado para estandarizar la manera de agregar meta datos para cualquier

propósito, pero a los efectos del uso cotidiano, resultó un lenguaje demasiado complicado.

Además agregan que HTML (HyperText Markup Language - Lenguaje de Marcas de Hipertexto), está basado en SGML, dado que utiliza 'TAGs' (etiquetas) semejantes para mostrar la información, y vincular diferentes fragmentos de información. HTML es un subconjunto modificado y muy reducido del SGML.

Sas Jacobs [Jacobs06] coincide con ellos en el punto que por lo complicado del lenguaje SGML, este no es el mas adecuado para el intercambio de datos en la web, aún cuando HTML haya tenido éxito, está limitado a sólo mostrar documentos en un navegador, esto es así, ya que las etiquetas no dan ninguna información acerca del contenido que abarcan, sino que solo son instrucciones de cómo mostrar el contenido; es por eso que XML apunta a ser mucho más simple.

Con esto Hunter et al [Hunter et al 04] explican que el XML (eXtensible Markup Language - Lenguaje de Marcas Extensible) fue creado para cubrir las falencias de los otros lenguajes, al ser un subconjunto de SGML, tiene los mismos objetivos, pero elimina considerablemente la complejidad. Al ser compatible con SGML, la sintaxis que se utiliza para generar un documento XML es la misma que para uno definido con la sintaxis SGML, y esto lleva a que puede ser leído con las herramientas que existen para SGML.

Por su parte Jacobs [Jacobs06] establece que un punto importante en XML es que no es un lenguaje sino que es un metalenguaje que se utiliza para construir otros lenguajes y vocabularios (describiendo las reglas que los definen), punto en el que coinciden las apreciaciones de Hunter et al. Prosiguiendo Jacobs [Jacobs06] explica que los lenguajes son diferentes pero todos contienen etiquetas que marcan su contenido, la elección de nombres de etiquetas y sus estructuras son flexibles, por lo que existen grupos de lenguajes que concuerda en un vocabulario XML estándar para poder compartir la información entre si. Establece que XHTML

es un ejemplo de XML, ya que describe un conjunto estándar de etiquetas que se deben de utilizar de una forma especifica, por ejemplo, que cada página XHTML se divide en 2 secciones, donde en cada una de ellas se puede incluir una cierta cantidad de etiquetas, (siendo algunas etiquetas exclusivas de cada sección).

Resumiendo para Jacobs [Jacobs06] XML es un dialecto extremadamente simple del SGML, cuyo objetivo es permitir que el SGML genérico sea puesto en servicios (en un servidor), recibido y procesado en la web de la misma manera que hoy es posible con el HTML, dado que XML fue diseñado para facilitar la implementación y interoperabilidad tanto con SGML como con HTML.

2. WEB 2.0.

2.1 Historia.

En sus orígenes las páginas Web estaban programadas en código HTML y se presentaban con características estáticas y monótonas, por ejemplo “fondo blanco y enlaces azules” [Garitano], de manera que su contenido no era actualizado frecuentemente; este tipo de páginas pertenecían a la Web 1.0.

Con el tiempo las páginas fueron adquiriendo mayor protagonismo, por lo que fue elección de muchos dar un paso más adelante, y de ésta manera migrar hacia la nueva tecnología escribiendo las páginas utilizando el lenguaje PHP, que ofrecía mayor dinamismo e interacción entre el usuario final y la página, manteniendo la actualización de las mismas accediendo para esto a bases de datos, también comenzaron a evolucionar los sistemas de gestión de contenido o CMS quienes mostraban páginas HTML dinámicas, las cuales eran creadas al momento de su visualización desde una base de datos actualizada, a este grupo de páginas a veces se las llamaba Web 1.5 [Wiki04].

Poco a poco la hoja de estilo o CSS, fue ganándose un lugar según los objetivos que se buscaba con la evolución de la web, en ambos sentidos, el conseguir hits (visitas) y la estética visual eran considerados como unos factores muy importantes [Wiki04], para poder de esta manera marcar independencia entre el contenido y el diseño de la página. En muchos casos este movimiento llevo a la mejora parcial y / o total de las páginas web “el HTML fue sustituido por XHTML y poco a poco las páginas web fueron evolucionando y mutando” [Garitano], al punto que asociaciones internacionales tomaron intervención para establecer un orden a seguir (o 'estándar'), con objeto de organizar los elementos que conforman la gran

red de redes (Internet), “...la W3C (y en menor medida ISO) se encarga de marcar las pautas y los estándares Web” [Garitano].

Todo esto llevó a que las páginas Web oficien de puntos de encuentro, cuyo propósito es buscar la interrelación entre los usuarios formando redes sociales, estas particularidades son las que se destacan en la Web 2.0.

El origen del término Web 2.0 tiene lugar el 24 de Octubre del 2004 “... en la “Web 2.0 Conference” cuando Dale Dougherty de O'Reilly Media y Craig Cline de MediaLive hablaban del renacimiento y la evolución de la web.” [Garitano]; de dicha conferencia a partir del brainstorming (tormenta de ideas) y de ejemplos propuestos se formularon interpretaciones acerca de la Web 2.0, surgiendo las comparaciones entre las distintas webs, algunas de las cuales se desarrollarán mas adelante y pueden apreciarse en la tabla 1, son:

Web 1.0	**Web 2.0**
DoubleClick	Google AdSense, Overture (¡Yahoo! Search Marketing)
Britannica	Wikipedia
Webs personales	Blogging
Publicar	Participación
Fidelización (Stickiness)	Sindicación

Tabla 1: Tabla comparativa entre Web 1.0 y Web 2.0. [TAB1]

Debido que, para la realización de la tabla comparativa entre estas tecnologías, se apoyaron en ejemplos concretos (por ejemplo sitios web, conceptos, servicios, aplicaciones web), la cantidad de los mismos, llevo a que la tabla creciera cada vez mas, según lo comentado por Tim O'Reilly [O'Reilly06], por lo que surgió el interrogante que es lo que llevo a poder establecer cuales eran los enfoques que pertenecían a la Web 1.0 y cuales a la Web 2.0; para esto los responsables de la comparativa, comenzaron a analizar las características y a establecer principios de los casos de éxito de la Web 1.0 y los casos que mejor defienden el concepto de Web 2.0.

Para llevar esto a cabo y poder comprender de manera mas fácil lo que la Web 2.0 significa, partiendo del análisis de los ejemplos mas interesantes de la Web 2.0, se valieron del mapa meme de la Web 2.0, desarrollado en una sesión de *brainstorming* durante el FOO Camp (evento organizado por O'Reilly Media), el cual resume la cercanía conceptual gráfica acerca de la Web 2.0, como se muestra en la figura 6:

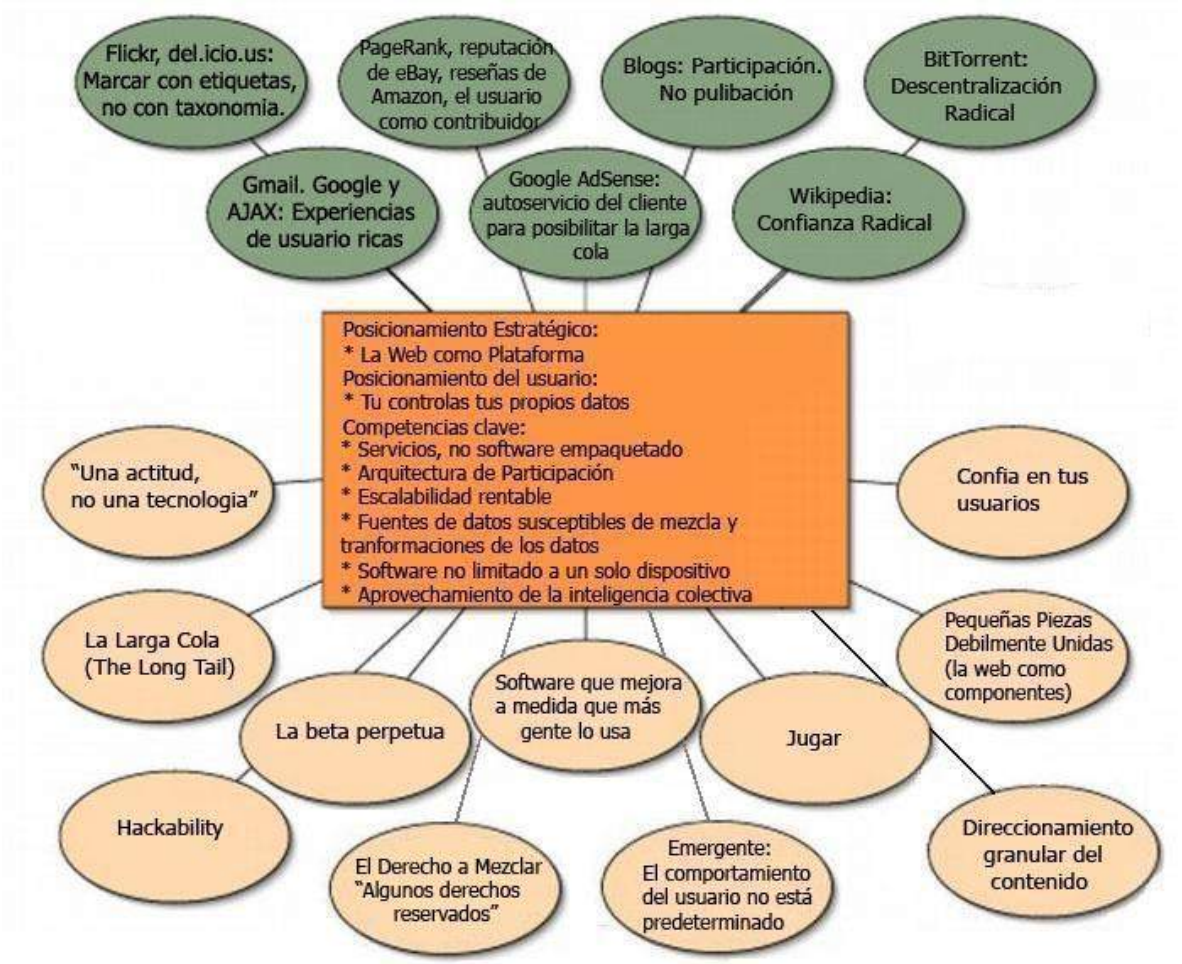


Figura 6: Mapa meme de la Web 2.0 [FIG6].

Tim O'Reilly [O'Reilly06] explica que DoubleClick con su modelo de negocio apuntaba solo a una parte del mercado, limitándose a unos pocos miles de sitios web grandes, apostando a la publicación de contenidos informativos (publicidad)

que se demandaban por parte de los anunciantes; mientras que el éxito de Google AdSense dio frutos gracias al poder colectivo de los sitios web pequeños (característico de la participación de los usuarios) que forman la gran mayoría de los contenidos de la web, por medio de la colocación de anuncios de texto simples y dependientes del contexto en el cual el usuario se encuentre navegando, siendo muy apreciados por los consumidores y para nada intrusivos (ni banners, ni popups).

Otro principio de la web 2.0, en este caso es que el servicio cuanta mas gente lo utilice, más mejora automáticamente.

Continuando con la explicación de Tim O'Reilly [O'Reilly06], Wikipedia ofrece una enciclopedia que es editada por cualquier usuario de la web, y corregida por cualquier otro, basándose en la confianza mutua entre los mismos; Britannica ofrecía un sistema profesional de gestión de contenidos, que la diferencia del caso de Wikipedia, mencionado antes).

Una voz crítica del sistema de Wikipedia, al cual brinda una mirada despectiva, es Nicholas Carr [Carr05] quien se presenta como uno de sus críticos acérrimos, opina que la Wikipedia es útil, para la obtención de distintas perspectivas sobre un tema, pero no es lo suficientemente buena, ya que a un nivel fundamental es poco confiable y su redacción suele ser decepcionante.

Por otra lado, en un informe especial llevado a cabo por la revista Nature respecto a la competencia entre Wikipedia y la Enciclopedia Britannica, Jim Giles [Giles05] establece que Wikipedia esta creciendo rápido (respecto a la cantidad de artículos que se han agregado con soporte para varios idiomas, a la cantidad de usuarios registrados y a la cantidad de visitas que recibe), así como también crece la calidad de los contenidos de los artículos. En una investigación liderada por Nature, se comparó la cobertura científica de Wikipedia y la Enciclopedia Britannica, en varios de sus artículos, para poder determinar la exactitud de la

información. Esta comparación dio como resultado una serie de errores serios, tales como interpretaciones erróneas de conceptos importantes, cuyo porcentaje se repartía en partes iguales para cada enciclopedia. Aun cuando los márgenes de errores pueden variar, atribuyéndole a Wikipedia la mayor cantidad, se detalla una ventaja de la Wikipedia la cual da lugar a que se actualicen los contenidos de los artículos hasta que el mismo alcance una calidad especifica, de manera de poder etiquetarlo como “estable” y apuntar a una calidad del tipo de contenido de Britannica o aun mejor.

Por ultimo, Tim O'Reilly [O'Reilly06] cita la característica establecida por el auge del blogging, que son páginas personales con formato de diario (al igual que las del comienzo de la web), pero en las cuales por medio de una tecnología llamada RSS, se permite que los usuarios no sólo se enlacen con una página, sino que puedan suscribirse a la misma, siendo notificados de los cambios. En mayor detalle, los blogs hacen uso de la sindicación o redifusión de contenidos para poder comunicar sus actualizaciones a los usuarios. En contrario a esto, la fidelización era una característica de los sitios web publicados que tenía como objetivo mantener la atención del usuario por un largo periodo, de manera que el usuario visitara múltiples páginas dentro del mismo sitio. Estableciéndose así una diferencia entre las web estáticas y las dinámicas.

2.2 Características y concepto.

A partir de los conceptos, practicas, ideas y principios que se expusieron de la Web 2.0 en la conferencia, algunos de los cuales se detallan en el mapa meme que se muestra en la figura 6, y de las tecnologías y técnicas que ésta utiliza, se pusieron de manifiesto las características de la Web 2.0 [Garitano] y [Wiki04]:

* *Transforma el software de escritorio hacia la plataforma del web.*
* *Respeto a los estándares de XHTML:* versión XML del HTML la cual sigue especificaciones propias del lenguaje XML.

* *Separación de contenido del diseño con uso de hojas de estilo (CSS):* de manera de poder centralizar colores tamaños y fuentes de las distintas etiquetas que forman la página.
* *Uso de Microformatos.*
* *Sindicación (redifusión) de contenidos (datos en RSS/ATOM):* permiten a los usuarios finales usar el contenido de la web en otro contexto, sea en otra web o en una aplicación de escritorio
* *Agregación de contenidos (datos en RSS/ATOM).*
* *Técnicas de aplicaciones de formato enriquecido no intrusivas (más conocido como enriquecimiento de las aplicaciones de Internet, con el uso de AJAX, Flash, Flex, Lazlo o Ruby on Rails):* permite programar páginas dinámicas.
* *Facilita el posicionamiento con URLs sencillas y con significado.*
* *Facilita las interacciones.*
* *Ahorra tiempo al usuario.*
* *Simplifica la usabilidad del sitio web.*
* *Facilita la publicación, la investigación y la consulta de contenidos web.*
* *Utilización de redes sociales al manejar usuarios y comunidades*.
* *Da mayor control a los usuarios en el manejo de su información.*
* *Soporte para publicar en un blog.*
* *Uso de Java Web Start:* permite correr aplicaciones de Java que se encuentran del lado del servidor web.
* *Uso de XUL:* permite desarrollar una interfaz de usuario portable, fácil y rápidamente.

Con la exposición de estas características, la conferencia apuntó a intentar aclarar a que nos estamos refiriendo cuando hacemos mención de la Web 2.0.

Más allá de estas características, existen posturas en las cuales se manifiesta una posición más simplificada y a la vez más técnica, como es el caso de Tim Bray [Bray06] quien luego del análisis de ciertos ejemplos considerados de la web 2.0, propone que "*la única novedad importante es que la red es de lectura/escritura. Todo lo que cuenta se desprende de ello."* En apoyo a lo expuesto, Dion Hinchliffe [Hinchcliffe06] aporta su comentario diciendo que este fenómeno toma sentido gracias a "*la existencia de áreas de lectura/escritura de la web ampliamente accesibles y de una amplía población calificada interesada, dispuesta y capaz de contribuir y de consumir contenido de lectura/escritura.*". Por su parte, Tim O'Reilly [O'Reilly-UB06], comenta respecto a la mejora de los contenidos de una aplicación estableciendo que "...*una verdadera aplicación web 2.0 es una que mejora mientras más personas la usan. Google se vuelve mas dinámico cada vez que alguien realiza un enlace en la web… ...realiza una búsqueda... ...hace click en un anuncio. Y esto actúa inmediatamente sobre esa información mejorando la experiencia para todos los demás... ...el corazón verdadero de web 2.0 es la capacidad de aprovechar la inteligencia colectiva*.", (además de las innovaciones que aportan quienes llevan a cabo las aplicaciones web 2.0, para poder ganar en el mercado).

2.3 Premisas fundamentales.

Entre los detalles y conceptos que ya se han citado, se mencionan algunos principios claves y practicas, que han llevado a la web a evolucionar tanto social, como técnica, económica y filosóficamente, de acuerdo a ciertas consideraciones esenciales que expone en su ensayo Tim O'Reilly [O'Reilly06] y por su parte comenta José Antonio del Moral [Moral06], algunas de ellas son:

* Según O'Reilly [O'Reilly06] los sitios Web 2.0 deben tender a los servicios y no al software como producto. La aplicación para armar un blog, el

alojamiento de un video en YouTube, la posibilidad de hacer mapas con Google Maps son servicios y no paquetes de software que se descargan e instalan. De acuerdo a esta postura, del Moral [Moral06] propone la inclusión de algunos de los siguiente servicios: blog, redes sociales, wikis y sistemas para compartir contenidos. Ejemplos: YouTube, Flickr, MySpace, Menéame, Microsiervos y Wikipedia. Dando lugar a una web simple, sencilla sin la necesidad de la descarga de software adicional por parte del usuario para que pueda utilizarla, y pudiendo realizar cualquier operación solo con el navegador abierto sin tener que ejecutar otros programas.

* En este caso, O'Reilly [O'Reilly06] propone modelos livianos de programación dado que la Web adopta cada vez con más fuerza estándares sencillos, de carga liviana, por ejemplo la sindicación por medio de RSS o la compatibilidad con XML, y el uso de AJAX para el reemplazo de sistemas complejos por técnicas más livianas. De igual manera del Moral [Moral06] manifiesta la idea de emplear tecnologías XML y AJAX, la cuales permiten la simplificación y facilitan las cosas al momento de usar internet.
* O'Reilly [O'Reilly06] propone que los usuarios deben ser tratados como co-desarrolladores, teniendo en cuenta las prácticas de desarrollo de software actuales que derivan en 'la versión beta perpetua' (por ejemplo el GMAIL de Google), en la cual se desarrolla un producto continuamente, incorporándole nuevas funcionalidades mensual, semanal, o incluso diariamente. Recurriendo al usuario para la prueba y aceptación de estas mejoras. Para del Moral [Moral06] el estado de una web 2.0 en beta (versión de prueba) refiere a la idea de un feedback (retroalimentación) entre el creador de la web y el usuario de la misma, el creador permite el uso de la web en su versión de prueba a cambio del testeo o el “control de calidad” por parte del usuario (aceptando siempre que se está en versión beta), para que el creador pueda mejorarla.

Pese a todo lo establecido hasta el momento, existen opiniones que manifiestan su desacuerdo contra las ideas que expone en su ensayo Tim O'Reilly [O'Reilly06]. Nicholas Carr [Carr05], por su parte, acusa de amorales a los promotores de la Web 2.0, al propagar por la web la idea de una fuerza moral, es decir, una necesidad personal de trascender, la cual no da lugar a ver a la red de manera objetiva; y de venerar a los amateurs y desconfiar de los profesionales. Entre los distintos puntos que formula en su ensayo, plantea que la internet es para realizar negocios y no para difundir ideas de una comunidad; la estructura de la web 2.0, por ejemplo la propia tipología de la red y las herramientas colaborativas pueden propagar ideas buenas y malas fácilmente, lo cual representa un verdadero peligro; la confianza de contenidos que se basan en datos de terceros que no han sido comprobados, riesgo proveniente de los amateurs. Una de las críticas donde más hace hincapié es en la facilitación medios de producción a las masas, negándole la propiedad del resultado de su trabajo, pudiendo dar lugar a que una minoría concentre el valor económico de todo ese esfuerzo distribuido y no retribuido.

3. LIBRERÍAS Y FRAMEWORKS.

3.1 Conceptos básicos.

A lo largo de este capitulo se describirán de forma particular algunas de las mas importantes librerías y frameworks del lenguaje Javascript, para ello es conveniente tener en claro los conceptos técnicos de librería y framework, daremos ahora una breve explicación de los mismos.

Una librería (o biblioteca) es un conjunto de procedimientos y funciones (subprogramas) agrupadas en un archivo con el fin de que puedan ser aprovechadas por otros programas. Al proceso de hacer accesibles estos subprogramas al programa principal se le llama: enlace [Wiki05].

Existen dos tipos de librería:

* las estáticas o de enlace estático: se enlazan (las referencias a rutinas en el programa para que apunten a su localización en la librería) en el momento de compilación
* las compartidas o de enlace dinámico: se enlazan en tiempo de ejecución.

Por su parte, un framework es una estructura definida de soporte en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, librerías y un lenguaje de scripting entre otros softwares para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los frameworks son diseñados como un intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional [Wiki06].

3.2 Librerías y frameworks de Javascript.

3.2.1 Prototype.

Prototype es un framework de Javascript que apunta a facilitar el desarrollo de aplicaciones web dinámicas. Cuenta con un kit de herramientas muy fácil de usar y una librería de AJAX de las mas sólida de todas, Prototype está convirtiéndose rápidamente en el código base de elección para los desarrolladores de aplicaciones web en todo el mundo, así lo expresa su creador Sam Stephenson [Prototype].

Dave Crane, Eric Pascarello y Darren James [Crane-Pascarello-James06] exponen que Prototype es una biblioteca de propósitos generales para la programación Javascript, con énfasis en extender el lenguaje de Javascript a un estilo más de programación orientada a objetos. El código Prototype puede ser difícil de leer, pero usar Prototype y las librerías que están basadas en el, es muy sencillo. Prototype puede ser visto como una librería de librerías para los desarrolladores, siendo mas probable que se utilicen librerías basadas en el, que usar a Prototype mismo. Puntos en los que también coinciden, salvando las distancias, Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05].

Por su parte Dave Crane, Eric Pascarello y Darren James [Crane-Pascarello-James06] definen que Prototype permite que un objeto extienda (relación de herencia en programación orientada a objetos) de otro, copiando todas la

propiedades y métodos del objeto padre al hijo (es decir, un objeto 'hijo' hereda las propiedades y métodos de un objeto 'padre').
A modo de ejemplo y para comprender este concepto Dave Crane, Eric Pascarello y Darren James [Crane-Pascarello-James06], detallan como definir una instancia específica que represente a un tren de pasajeros:

* La clase padre:

```
function Maquina(numRuedas,velocMax){
      this.numRuedas = numRuedas;
      this.velocMax = velocMax;
}
```

* La funcionalidad extendida como un objeto (ya que Prototype permite definir el comportamiento extendido como un objeto y luego extender el objeto base con el objeto anterior):

```
function Vagon(vagonCant){
      this.vagonCant = vagonCant;
      this.agregarVagon = function(){
            this.vagonCant++;
      }
      this.eliminarVagon = function(){
            this.vagonCant--;
      }
}
```

* Fusión entre los objetos padre y extensión (relación padre-hijo entre las instancias), da como resultado un objeto con el comportamiento especifico:

```
var padre = new Maquina(24, 100);
var extension = new Vagon(12);
var TrenDePasajeros = Object.extend(padre, extension);
```

Según Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05], Prototype agrega una serie de métodos de gran utilidad, por ejemplo funciones accesibles globalmente, o extender a tipos existentes en Javascript (como se menciono antes). Los tipos Javascript pueden ser extendidos en tiempo de ejecución sin modificar el código original, como resultado, Prototype brinda atajos extremadamente útiles a la funcionalidad común, presentándola de la forma más natural posible: como propiedades y métodos de los tipos.

Entre las distintas características que introduce Prototype (*"extend"* es una de ellas como se ha visto), Dave Crane, Eric Pascarello y Darren James [Crane-Pascarello-James06] mencionan el método $, el cual encapsula una de las tareas mas comunes en la programación DHTML, el obtener el elemento de un documento base a partir de su ID (identificador), una línea común de Javascript en una aplicación DHMTL seria:

```
var miElemento = document.getElementById('idElemento');
```

Prototype la reemplaza por la siguiente línea:

```
var miElemento = $('idElemento');
```

Una desventaja de esta notación como lo detalla Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05], es que Prototype no chequea si el ID que se pasa es valido dentro del documento, por lo que si en ese caso, no se encontrara el elemento con dicho ID el resultado de la variable seria nulo.

3.2.2 Script.aculo.us.

Scriptaculous es una librería de efectos visuales bien documentada, escrita en Javascript basada en Prototype; la cual incluye demostraciones, ejemplos de

aplicaciones y una librería de arrastrar-y-soltar (drag-and-drop), así lo expone, su creador y quien lidera la lista de autores, Thomas Fuchs [Scriptaculous].

Según la postura de Dave Crane, Eric Pascarello y Darren James [Crane-Pascarello-James06], las librerías de Scriptaculous son componentes de interfaz de usuario basadas en Prototype. Mas en detalle, Scriptaculous brinda dos grandes funcionalidades, por un lado se tiene la librería de efectos y por otro la librería de arrastrar-y-soltar.

La librería de efectos define un rango de efectos visuales animados que pueden ser aplicados a elementos DOM, para que se pueda cambiar su tamaño, posición y transparencia. Scriptaculous permite que los efectos se combinen fácilmente, brindando para ello un número predefinido de efectos secundarios, tales como el efecto *Puff*, hace que un elemento sea más grande y más transparente hasta que se desvanece totalmente. Otro efecto para destacar es *Parallel*, que posibilita la ejecución simultánea de múltiples efectos.

Desde el punto de vista de Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05] acerca de Scriptaculous, comentan que mientras que Prototype se enfocó en extender la base de las capacidades de Javascript y DOM, Scriptaculous permitió que los desarrolladores web hagan que las páginas HTML luzcan y actúen como cualquier otra plataforma de cliente en formato enriquecido.
Estos autores ofrecen un mayor detalle de cómo la librería de efectos está dividida en 5 efectos principales:

- ***Opacity***: permite cambiar el porcentaje de opacidad de 0% a 100%
- ***Highlight****:* permite cambiar el color de fondo de un elemento desde un color inicial hasta un color final pasando por el espectro entre ellos.
- ***MoveBy****:* permite controlar fácilmente el reposicionamiento de elementos.
- ***Scale****:* permite afectar el tamaño de elementos

- ***Parallel**:* permite la combinación en la ejecución de los efectos (como se menciono anteriormente)

Además de la serie de combinaciones que se le logran entre ellos. Cada efecto representa una transición entre dos estados que se produce a lo largo del tiempo. Todos los efectos tienen valores por defecto para los puntos de comienzo y finalización, así como también duración.

Los efectos son todos asincrónicos, esto significa que si lanzan varios efectos simultáneamente serán interpretados con esa simultaneidad, ya sea que los efectos tengan como destino diferentes elementos o que todos los efectos tengan como destino el mismo elemento.

Siguiendo con la segunda funcionalidad que ofrece Scriptaculous Dave Crane, Eric Pascarello y Darren James [Crane-Pascarello-James06] explican que la librería de arrastrar-y-soltar toma protagonismo a través de la clase *Sortable*, la cual toma como argumento a un elemento DOM padre, permitiendo la funcionalidad de arrastrar-y-soltar a todos sus hijos. Las opciones que se pueden pasar al constructor de la clase, son entre otras; la llamada de regreso cuando un elemento es arrastrado y soltado, los tipos de elementos hijos que van a ser arrastrados, objetos efectos, en este caso el objeto será ejecutado cuando el elemento se comienza a arrastrar, está en transito y es soltado.

Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05] comentan que las llamadas de regreso son asincrónicas y además advierten que en algunos navegadores aparecen cuadros de diálogos (conocidos como “pop-up”) ya sea de alertas o confirmaciones, que permiten que el efecto continué pero no sea visible hasta tanto el usuario cierre ese dialogo. Por lo que si la duración del efecto pasa, para el momento en que el usuario cerró la ventana, este habrá finalizado sin que el usuario haya podido verlo.

3.2.3 Dojo.

Dojo es un kit de herramienta (toolkit) de HTML dinámica (DHTML) de código abierto escrito en Javascript, que provee varias librerías para utilizar con AJAX, incluyendo widget, un evento modelo, y el uso de mensajeria utilizando XMLHttpRequest y otras técnicas.

Dojo facilita el desarrollo de aplicaciones web que utilicen tecnología AJAX haciéndolas mas fácil de usar, flexibles y funcionales. Además permite construir interfaces de usuario más fácilmente, prototipos interactivos de widget más rápido y animar transiciones, de esta manera describe Alex Russell [Dojo] quien se sitúa como líder del proyecto Dojo Toolkit.

Sang Shin [Shin-Dojo07] resume que Dojo es un conjunto de librerías de Javascript que apuntan a resolver un largo historial de problemas que se dan con DHTML por ejemplo la incompatibilidad de navegadores, además de permitir agregar capacidades dinámicas en las páginas web utilizando para ello widget y animaciones.

Por su parte Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05] establecen que Dojo es diferente de Prototype y Scriptaculous, dado que éstas dos, son librerías mas pequeñas, y centran su atención en las bondades de las interfaz de usuario junto al soporte para XMLHttpRequest; mientras que Dojo es esencialmente toda una plataforma para la construcción de aplicaciones de cliente, además de su XMLHttpRequest y módulos de efectos, incluye colecciones de librerías de Javascript, widget, un módulo de registro, un módulo de matemáticas, entre otras tantas.

La postura de Ryan Asleson y Nathaniel T. Schutta [Asleson-Schutta06], es que siendo Dojo uno de los framework más viejos, su meta como proyecto es construir

un kit de herramientas de DHTML que aproveche al objeto XMLHttpRequest enfocándose en cuestiones de usabilidad.

Una de las características de Dojo es el soporte para los botones atrás y adelante de un navegador, registrando un método de regreso cuando el usuario realiza un click, cualquiera sea el botón, hacia atrás o hacia delante.

Otras características que Sang Shin [Shin-Dojo07] destaca de Dojo son; su sistema de eventos orientado a aspectos, esto permite que se pueda controlar o manejar el evento antes o después que el mismo ocurra; proporciona etiquetas de marcado basadas en construcciones de interfaz de usuario (UI) por medio de los widgets, además permite que se puedan construir y reutilizar widget, sean construidos por uno mismo o por terceros.

3.2.4 DWR Direct Web Remoting.

DWR es una librería de Java y de Javascript de código abierto que permite escribir sitios web en AJAX. Además permite que el código que se encuentra en el navegador utilice las funciones de Java, que se están ejecutando en el servidor web, como si estuviera en el navegador (es por eso que DWR se llama *"remoto directo"*).

DWR consiste en dos partes principales: un Servlet Java que se está ejecutando del lado del servidor que procesa las solicitudes y envía las respuestas al navegador; y Javascript ejecutándose en el navegador que envía las solicitudes y dinámicamente actualiza la página web; de acuerdo a lo que se detalla en su sitio web [DWR].

Sang Shin [Shin-DWR07] detalla el funcionamiento de DWR, que genera dinámicamente una clase Javascript del lado del cliente desde una clase de Java, a partir de esto es que DWR permite escribir código Javascript.

La clase Javascript generada maneja detalles de *interacción remota*, esto se refiere a manejo de la comunicación asincrónica a través del uso del objeto XMLHttpRequest, la invocación de la función de llamada de regreso del lado del cliente y conversión de todos lo parámetros y valores retornados, entre el navegador y los procesos del servidor.

3.2.5 jQuery.

jQuery es una librería de Javascript que permite mantener el código de una página de manera simple y concisa manejando algunas de las complejidades que se dan con DOM y con la interacción de AJAX. jQuery incluye plugins opcionales que contienen funciones comunes para DOM, Eventos, Efectos y AJAX; así lo establece Mark Constable [Constable06].

De acuerdo a Christian Heilmann [Heilmann06], jQuery es un simple y liviano archivo de Javascript (ronda los 16 Kbytes) que se agrega a la cabecera de los documentos. Provee una increíble cantidad de métodos de utilidad para lograr tareas web específicas.

El concepto de jQuery es ofrecer acceso rápido a cualquier elemento del documento, para obtener el acceso rápido se tiene un método de utilidad llamado $(de todos los elementos), que puede tomar:

* Una construcción DOM, ejemplo: `$( document.body )`,
* Un selector CSS, ejemplo: `$( 'p a' )`, que es cada uno de los enlaces que se encuentran dentro del párrafo del documento,

* Una expresión Xpath, ejemplo: `$(" //a[@rel='nofollow'] ")`, que relaciona cada enlace del documento con un atributo llamado *rel* que tiene como valor a *nofollow.*

Otra de las particularidades que jQuery ofrece son los *métodos encadenables*, en donde se puede agregar cada método al ultimo concatenándolos.

En adición Mark Constable [Constable06], explica que cada vez que se llama a un método con un objeto jQuery, el método retorna el mismo objeto jQuery, por lo que, si se necesita llamar a múltiples métodos, se puede hacer sin tener que re escribir el selector.

A modo de ejemplo Heilmann [Heilmann06], comenta que en lugar de:

```
$p = $( 'p' );
$p.addClass( 'prueba' );
$p.show();
$p.html('ejemplo' );
```

se puede escribir

```
$( 'p' ).addClass( 'prueba' ).show().html('ejemplo');
```

En ambos casos hace lo mismos, toma cada párrafo del documento, agrega una clase CSS llamada *"prueba"*, muestra el párrafo (en el caso de estar oculto) y cambia el contenido HTML del párrafo a *"ejemplo"*.

3.2.6 Rico.

Rico es un framework multipropósito con soporte para AJAX, abarca cuestiones de interfaz de usuario por ejemplo animaciones, separación del contenido de la

lógica a través de comportamientos, arrastrar y soltar, y algunos widget precompilados, particularmente la grilla de datos. Rico está basado en Prototype. Así lo establecen en el sitio web los responsables del proyecto Bill Scott y Darren James [Rico].

Darren James junto con Dave Crane y Eric Pascarello [Crane-Pascarello-James06] sostiene (como se menciono antes con Bill Scott) que Rico, así como Scriptaculous, se basa en la librería Prototype y proporciona varios efectos altamente personalizables y funcionalidades de arrastrado y soltado. Pero en esta oportunidad estos tres autores agregan que Rico provee un concepto de objeto Comportamiento, esto es, un fragmento de código que puede ser aplicado a la parte de un árbol DOM para agregarle funcionalidad interactiva. Algunos de los ejemplos de Comportamientos son, el widget Acordeón, que anida un conjunto de elementos DOM dentro de un espacio determinado, permitiendo expandirse uno por vez.

Los Comportamientos en Rico proporcionan una forma simple de crear widgets reutilizables a partir de marcajes [etiquetas, marcas] comunes y además separar el contenido de la interactividad.

La característica final que estos tres autores mencionan es la ampliación a lo que se comento al comienzo, esto es el muy buen soporte con las solicitudes de AJAX al servidor, a través del objeto global de Rico, el motor AJAX. El motor AJAX define una respuesta con formato XML que consiste en un numero de elementos <response> (respuesta). El motor automáticamente la decodifica, dando lugar a 2 tipos de respuesta: las que actualizan directamente a elementos DOM; y las que actualizan los objetos Javascript

3.2.7 JSON / JSON-RPC.

La Notación de Objetos Javascript (JSON), como se explica en su sitio web [JSON] y del cual también hace referencia Sang Shin [Shin-JSON07], así como también Ryan Asleson y Nathaniel T. Schutta [Asleson-Schutta06], es un formato liviano de intercambio de datos. Es fácil para que las personas lo lean y escriban, así como también para que las maquinas lo analicen y lo generen. Además brinda mecanismos concisos para crear arreglos y objetos gráficos. JSON es un formato de texto que es completamente independiente del lenguaje aunque utiliza convenciones de lenguajes tales como C, C++, C#, Java, Javascript, Perl, Python y muchos otros. Estas propiedades hacen que sea un lenguaje de intercambio de dato ideal.

JSON se basa en 2 estructuras:

* Una colección de pares de nombre / valor. En varios lenguajes, esto se realiza como un *objeto*, registro, estructura, diccionario, tabla hash, lista de claves, o arreglos asociativos.
* Una lista ordenada de valores. En muchos lenguajes, esto se realiza como un *arreglo*, vector, lista, o secuencia.

Según lo que describe el sitio web [JSON-RPC], comentan Sang Shin [Shin-JSON07] y Ryan Asleson junto con Nathaniel T. Schutta [Asleson-Schutta06], JSON-RPC es un protocolo liviano de llamada a procedimiento remoto, con fuerte soporte para clientes Javascript.

El mecanismo general consiste en 2 pares que establecen una conexión de datos. Durante el tiempo de vida de la conexión, los pares pueden invocar a métodos que son proporcionados por el otro par. Para invocar un método remoto, se envía una solicitud. A menos que la solicitud sea una notificación debe ser respondida. En mayor detalle, la comunicación entre los pares consiste en:

* *Solicitud* (método de invocación): Un método remoto es invocado por una solicitud que es enviada a un servicio remoto.
* *Respuesta*: Cuando el método de invocación se completa, el servicio debe contestar con una respuesta.
* *Notificación*: Una notificación es una solicitud especial que no tiene una respuesta.

3.2.8 Sajax.

De acuerdo a lo que se expresa en su sito web [Sajax], Sajax es una herramienta de código abierto que hace que la programación de sitios web que utilizan AJAX, sean lo más fácilmente posibles de programar. Sajax hace que sea fácil llamar a las funciones PHP, Perl, ASP, Java o Python de las páginas vía Javascript sin tener que actualizar al navegador. Es decir que, Sajax permite llamar a funciones del lado del servidor por medio de Jasvascript, haciendo que este regrese. A modo de ejemplo, la llamada a un método Javascript `x_calcularAlgo()` ira hasta el servidor y llama a un método Java calcularAlgo(), que luego regresara un valor en Javascript a `x_calcularAlgo_cb()`.

En adición a lo antes expuesto Justin Gehtland, Ben Galbraith y Dion Almaer [Gehtland-Galbraith-Almaer05], sostienen que Sajax es uno de los primeros framework de AJAX disponibles, y que originalmente fue escrito para PHP. Permite la primera escritura de las funciones del lado del servidor que se necesitan, por ejemplo la comunicación con la base de datos y el regreso de los datos correctos, para luego enlazar directamente la interfaz de usuario del HTML con esas funciones.

3.2.9 Yahoo! User Interface Library.

La librería Yahoo! User Interface (YUI) Library es un conjunto de utilidades y controles, escritas en Javascript, para la creación de aplicaciones web de formato enriquecido por medio de la utilización de técnicas como DOM, DHTML y AJAX.

YUI Library también incluye varios recursos principales CSS. Todos los componentes de YUI Library han sido liberados bajo licencia de código abierto y son libres para todos los usos. Según lo que se encuentra citado en su sitio web [Yahoo!]

Por su parte Christian Heilmann [Heilmann06], al igual que lo antes expuesto permite la mejora de la aplicación web desde un comienzo, incluyendo prediseños de CSS para diferentes diseños y tamaño de fuentes (tipos). YUI Library proporciona buena documentación y ejemplos, los distintos componentes que la conforman están disponibles ya sea como archivos Javascript, o como versiones de archivos de tamaño optimizado (los cuales no presentan espacios en blanco y son compactados para obtener archivos de menor tamaño).

Christian Heilmann [Heilmann06] enumera alguno de los componentes de la librería YUI Library que son:

* Componente de animación para animar y disminuir o aumentar elementos
* Administrador de conexión para crear aplicaciones web
* Componente DOM para alcanzar y cambiar elementos y dinámicamente aplicarles clases de estilos (CSS)
* Componente de arrastrar y soltar
* Componente evento para el manejo de evento
* Control de autocompletado para los campos de los formularios
* Control de calendario para seleccionar los días para los formularios

* Control contenedor que crea elementos de la página codificables que pueden ser posicionados para cubrir el documento actual
* Control de menú para crear menús dinámicos
* Control deslizante
* Control de vista de árbol

A los cuales se les suma los que se aprecian en el sitio web de YUI Library [Yahoo!]:

* Control de botón que permite la creación de botones enriquecidos
* Control seleccionador de color que provee un interfaz visual enriquecida para la selección del color
* Control de tabla de dato (DataTable) provee una interfaz de programación de aplicaciones (API) simple para mostrar los datos en pantalla en la página web
* Control de registro que provee una manera fácil de leer o escribir los mensajes de registros con una simple línea de código
* Componente de vista de pestaña permite crear vistas de pestañas navegables
* Administrador del historial del navegador que permite grabar la navegación de los botones (adelante / atrás) en una aplicación web
* Cargador de imágenes que permite determinar los disparadores y limites de tiempo para iniciar la carga de una imagen

3.2.10 Google Web Toolkit.

Sang Shin [Shin-GWT07] con apoyo en lo detallado en el sitio web [GWT], describe a Google Web Toolkit (GWT) como un framework de desarrollo de

software en Java que permite crear aplicaciones AJAX fácilmente. GWT, además de desarrollar y depurar las mismas en lenguaje Java utilizando para ello herramientas y entorno de desarrollo Java que se prefiera. Una vez terminada aplicación, ya probada y depurada, se puede desplegar la aplicación. Durante la fase de despliegue, GWT compila y traduce el código Java a Javascript y HTML para que pueda ser compatible con cualquier navegador web.

Características de GWT entre otras:

* *Utilización del lenguaje Java:* no es necesario aprender Javascript
* *Utilización de APIs de Java:* no es necesario aprender las APIs de DOM
* *Componentes de la interfaz de usuarios dinámicos y re-utilizables:* permite crear un widget para construir otros.
* *Llamada Procedural Remota sencilla (RPC, Remote Procedural Call):* para comunicar al navegador que corre la aplicación con el servidor web, solo se necesita definir las clases de Java serializables para las solicitudes y respuestas. GWT serializa automáticamente las peticiones del navegador y de-serializa las repuestas desde el servidor web.
* *Administración del historial del navegador.*
* *Depuración en tiempo real:* aprovechando el sistema de depuración y manejo de excepciones que se incluyen en las IDEs que se utilizan para Java.
* *Compatibilidad con los navegadores.*
* *Integración con JUnit:* permite realizar pruebas unitarias en un depurador o un navegador.
* *Internacionalización:* permite crear aplicaciones y librerías de internacionalización de forma rápida y eficiente.
* *Interoperabilidad y control minucioso:* permite mezclar el código Javascript que se encuentra dentro de la aplicación Java, usando la

interfaz nativa de Javascript (Javascript Native Interface, JSNI), de manera de aprovechar las distintas librerías de Javascript.

Los beneficios de utilizar Java con GWT para desarrollos AJAX traduciendo Java a Javascript son; además que la tecnología Java en si ofrece una plataforma de desarrollo productiva, y junto con GWT, se puede convertir en una plataforma sólida para el desarrollo de aplicaciones AJAX; son los que se detallan a continuación:

* Posibilidad de utilizar cualquier IDE o herramienta de desarrollo Java como los ambientes de desarrollo Eclipse o NetBeans.
* El chequeo de tipo estático en Java aumenta la productividad al tiempo que reduce los errores.
* Los errores comunes en Javascript (errores de sintaxis, por ejemplo) son fácilmente detectados mientras se desarrolla la aplicación, y no cuando el usuario final lo está ejecutando.
* La “refactorización” automática en Java.
* Los diseños en Java basados en la programación orientada a objetos son fáciles de comunicar y entender, por lo que hace que el código base AJAX sea más comprensible con menos documentación.

En cuanto a la depuración y desarrollo de las aplicaciones en GWT, éstas pueden ser ejecutadas en dos modos:

* Modo alojado (hosted mode): en modo alojado, la aplicación corre como bytecodes de Java sobre la máquina virtual de Java (JVM). Generalmente consume más tiempo desarrollando en modo alojado, dado que corriéndola en la JVM se cuenta con todas las ventajas que proporciona Java para depurar usando una IDE. Para dar soporte al modo alojado, GWT cuenta con un navegador especial el cual está enlazado a la JVM.

* Modo Web (Web mode): en modo web, la aplicación corre como puro Javascript y HTML sobre un navegador, traducido desde el código fuente Java original con el compilador de GWT (Compilador Java-a-Javascript). Cuando la aplicación está terminada, se debe subir la aplicacion a un servidor web, para que los usuarios finales puedan accederla por medio de un navegador en “modo web”.

La arquitectura de GWT tiene cuatro componentes principales, como se detallan en la figura 7:

* Compilador GWT Java-a-Javascript: el compilador GWT Java-a-Javascript traduce del lenguaje de programación Java a Javascript. El compilador se utiliza cuando se necesita correr la aplicación en modo web.
* Navegador web “Alojado” de GWT: el navegador web de GWT permite correr y ejecutar aplicaciones GWT en modo alojado, donde lo que se corre son bytecodes de Java sobre la JVM sin compilarlos a Javascript. Para lograr esto, el navegador GWT embebe un controlador especial del navegador (un control del Internet Explorer sobre Windows o un control de Gecko / Mozilla sobre Linux) con enlaces dentro de la JVM.
* Emulación de librerías JRE: GWT contiene implementaciones en Javascript de las librerías de clases más usadas en Java (java.lang y java.util). El resto del estándar de librerías de Java no está soportado nativamente con GWT (java.io).
* Librería de clases de interfaz de usuario de GWT: las librerías de clases de interfaz de usuario de GWT son un conjunto de interfaces y clases personalizadas que permiten crear "widgets" para el navegador, por ejemplo botones, cajas de texto, imágenes y texto. Ésta es la principal librería de interfaz de usuario que se utiliza para crear aplicaciones GWT.

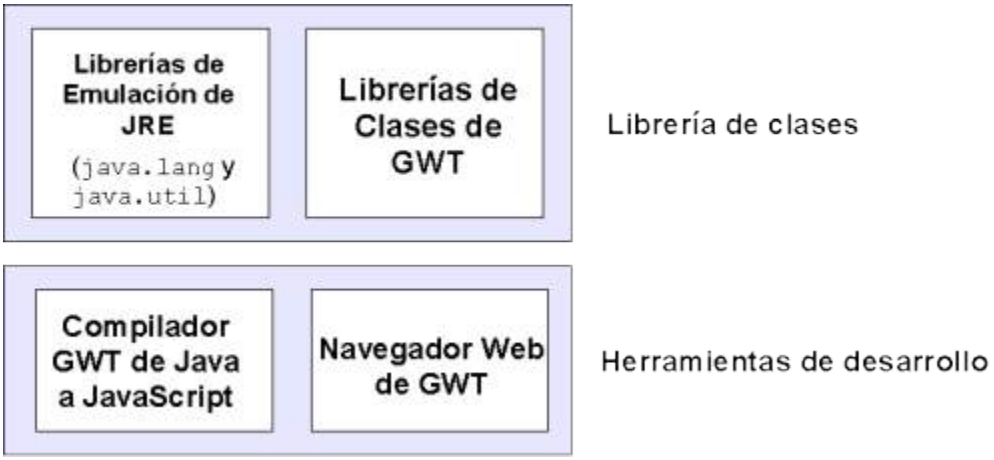


Figura 7: Componentes principales de GWT [FIG7].

4. PATRONES DE DISEÑO.

4.1 Concepto.

Un patrón de diseño es una descripción general (solución) para saber cómo resolver un problema común que suele ocurrir en el diseño de software. Para que una solución se considere como patrón debe poseer ciertas características, ya sea, haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores y ser reusable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias [Wiki07].

Un patrón de diseño sigue una estructura estándar y uniforme para la definición de su concepto, algunos de los componentes que la conforman son:

* Nombre
* Clasificación
* Intención: problema que resuelve
* Contexto Motivación: escenario de ejemplo.
* Aplicabilidad

* Participantes: descripción de las entidades participantes.
* Colaboraciones: Explicación de las interrelaciones que se dan entre los participantes.
* Consecuencias: positivas y negativas al momento de la aplicación del patrón.
* Implementación.
* Código de ejemplo: código fuente ejemplo.
* Usos conocidos: ejemplos de sistemas reales que lo usan.
* Patrones relacionados

Los objetivos a los que los patrones de diseño apuntan son entre otros [Wiki07]: proporcionar elementos reusables, evitar la búsqueda de soluciones a problemas ya conocidos y solucionados con anterioridad, crear un vocabulario común entre diseñadores y la estandarización en materia de diseño. Dejando de lado la idea de imponerse como alternativas de diseño frente a otras o eliminar la creatividad en el proceso de diseño.

El uso de patrones no es obligatorio, salvo el caso que se tenga un problema cuya solución la brinde el patrón (existen casos particulares donde la aplicación de patrones no es posible). Hacer abuso o forzar el uso de los patrones puede ser un error [Wiki07].

4.1.1 Patrones de diseño de AJAX.

4.1.1.1 Carga Predictiva (Predictive Fetch).

Para Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] este patrón surge a partir de las características que se muestran en una aplicación web tradicional, ya que ésta se presenta como una página con una

cierta cantidad de enlaces apuntando a distintos sitios, haciendo uso de lo que se conoce como: “carga por demanda”, donde por medio de la acción de un usuario se le solicita al servidor el dato exacto que quiere ser recuperado.
Si bien esta técnica está definida desde los comienzos de la web, tiene un efecto secundario que es el de forzar al usuario al modelo empezar-parar (propio de las web tradicionales).

Michael Mahemoff [Mahemoff06] por su parte coincide con estos 3 autores al decir que el anticiparse a la acción del usuario es un juego de adivinanzas, pero establece que el problema a resolver se funda en cómo hacer que el sistema responda más rápidamente a la actividad del usuario.

Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] proponen el uso del patrón de carga predictiva como solución posible dado que la aplicación AJAX supone qué es lo que el usuario va a hacer luego y recupera el dato apropiado. Michael Mahemoff [Mahemoff06] concuerda con ésta solución y en mayor detalle explica que la aplicación del patrón permite la precarga de los contenidos anticipándose a las probables acciones del usuario, buscando eliminar el retraso en determinadas acciones.

Mahemoff [Mahemoff06] comenta que la retroalimentación (feedback) instantánea está en la eficiencia por parte de la colaboración del usuario y que los problemas que se presentan en el uso del patrón son la distracción y la lentitud en la que el usuario responde a un estado transmitido por parte del servidor. Por otra parte determina en que ocasiones debería se utilizado dicho patrón:

* Cuando un usuario navega por un área virtual, por ejemplo una gran tabla. Precarga los resultados de movimientos en cada dirección. (Google Maps).

* Cuando un usuario realiza la conversión entre 2 monedas. Precarga los principales tipos de moneda.

* Cuando un usuario lee algunos artículos. Precarga todas las historias en su categoría preferida.

Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] detallan un ejemplo simple para ilustrar la aplicación del patrón, haciendo hincapié en el último punto mencionado por Mahemoff, suponiendo que un artículo online que está separado en 3 páginas, se considera lógico que si un usuario está interesado en la lectura de la primera página, también lo estará en la segunda y la tercera, por lo que si la primer página se cargó en pocos segundos, es seguro que la segunda página se descargue en segundo plano, guardándose en el cache del cliente (esto mismo es aplicable a la segunda y tercer página según el interés que presente el usuario). El patrón de carga predictiva hace uso del cache del lado del cliente para poder acumular todos los contenidos precargados

Por último, Michael Mahemoff [Mahemoff06] aconseja que tipo de información puede ser utilizada para poder anticiparse a las acciones del usuario, algunas de ellas son:

* El perfil de usuario, dado que su historial de navegación y perfil de visitas a un sitio dado proveen fuertes pistas.
* La actividad actual del usuario, es posible predecir que ese lo que el usuario hará luego a partir de su actividad actual.

A partir de esta información, es donde puede verse con mayor claridad el ejemplo del correo web de GMail que Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] explican, ya que cuando un persona está escribiendo un correo, la mayoría de las veces se conoce a la persona a quien se le escribe, por lo que se asume como lógico que la dirección de correo de esa persona ya está en la correspondiente agenda. Para facilitar las cosas se carga la agenda de direcciones en segundo plano para que llegado el momento de escribir el

destinatario, la agenda sugiera la dirección a partir de lo que la persona vaya ingresando.

4.1.1.2 Regulación de Envío (Submission Throttling).

Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] exponen la problemática de evitar la actualización de la página cuestionando cuando es importante enviar los datos de usuario. Dado que en una aplicación o sitio web tradicional, cada click da lugar a una solicitud al servidor, de esta manera el servidor tiene en cuenta todo lo que el cliente está haciendo. Mientras que con AJAX, el usuario interactúa con la aplicación o sitio web sin que se genere una solicitud adicional por cada click.

Una solución que ofrecen, es la de enviar los datos al servidor cada vez que un usuario realiza una acción. A modo de ejemplo cuando el usuario escribe una letra, ésta se envía inmediatamente al servidor, repitiéndose este proceso con cada letra que se escriba. El problema que se origina en la posibilidad de crear gran cantidad de solicitudes en un corto tiempo, complicando las prestaciones del servidor y aletargando a la interfaz de usuario.

Para poder brindar una alternativa a esta problemática es que Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] se apoyan en la utilización del patrón de regulación de envío. Este patrón permite almacenar los datos que se van a enviar al servidor, del lado del cliente para luego enviarlos en tiempos determinados. En términos mas específicos Michael Mahemoff [Mahemoff06] explica que el dato es retenido en un búfer basado en navegador y automáticamente es enviado a intervalos de tiempo fijos. Agregando que el propósito del almacenamiento y la regulación es encontrar el equilibrio entre la respuesta y los recursos; dado que esto se consigue aliviar el ancho de banda y disminuir el procesamiento del servidor.

Tanto Mahemoff [Mahemoff06] como Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] toman a Google Suggest como un claro ejemplo y la velocidad que le otorga el uso de este patrón, ya que no envía una solicitud por cada carácter que es escrito, sino que aguarda una cierta cantidad de tiempo y envía el texto que se haya escrito en ese momento; haciendo que la demora entre lo que se escribe y lo que se envía no sea relevante.

Como dato importante Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] advierten que dicho patrón no debe ser utilizado para datos de misión critica, ya que si la información debe ser enviada dentro de un tiempo especifico, se debería de usar formularios tradicionales para asegurar la correcta entrega de información a tiempo.

4.1.1.3 Actualización Periódica (Periodic Refresh).

Michael Mahemoff [Mahemoff06] detalla un caso donde un sitio web de venta de entradas, en el cual por cada evento de usuario que sucedía, debía mantener actualizada la cantidad de entradas que restaban; para ello se introdujo un temporizador que permitía llamadas a un servicio web el cual informaba acerca de las entradas vendidas. Este ejemplo es una introducción a la problemática de como hacer que una aplicación (o sitio web) mantenga informado al usuario de los cambios que suceden en el servidor, ofreciendo como solución el uso del patrón de actualización periódica.

Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] describen a este patrón como: el proceso de verificar nueva información en el servidor a intervalos de tiempo específicos. Dicho enfoque, también conocido como *sondeo*, requiere del navegador para realizar un seguimiento de cuando se debe llevar a cabo otra solicitud al servidor. El enfoque mas técnico por parte de

Michael Mahemoff [Mahemoff06] es que el navegador emite periódicamente una llamada XMLHttpRequest para obtener esta nueva información.

Según Mahemoff [Mahemoff06] los períodos de actualización difieren dependiendo del contexto, de los cuales se pueden definir 3 categorías; la interacción en tiempo real (definida en milisegundos como por ejemplo un chat web), monitoreo activo (definida en segundos como por ejemplo circuito cerrado de cámaras) y monitoreo casual (definido en minutos como por ejemplo resultados deportivos o la recepción de un nuevo correo como es el caso de GMail).

Por su parte Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] concluyen que el propósito básico de este patrón es el de notificar a los usuarios acerca de la información actualizada.

4.1.1.4 Descarga en Múltiples Etapas (Multi-Stage Download).

Uno de los problemas que perdura en la web es la velocidad con la que se descargan las páginas, Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett [Zakas-McPeak-Fawcett06] explican que cuando muchos usuarios usaban una conexión de 56 Kbps por modem, los diseñadores web debían estar alertas al tamaño de sus páginas; esto cambió con la introducción de la banda ancha, ya que muchos sitios pudieron actualizarse, incluyendo multimedia, mayor cantidad de fotos y demás contenidos. Si bien este cambio por un lado proporcionó mayor información para el usuario, por otro llevo a aletargar los tiempos de descargas dado que todo se carga en un orden aparentemente aleatorio. La solución propuesta por estos autores en este caso fue la utilización del patrón de descarga en múltiples etapas.
Este patrón se destaca porque se carga inicialmente en la página la funcionalidad más básica, una vez completa la carga, la página comienza a descargar los demás componentes que debieran aparecer. Si el usuario cierra la página antes

que se carguen la totalidad de los componentes, no existen inconveniente (la descarga se detiene con el cierre de la página).

Apoyando la solución propuesta Michael Mahemoff [Mahemoff06] expone que la forma de optimizar el rendimiento en la descarga es a través de la división en múltiples etapas de los contenidos que se van a descargar, de modo que el contenido más rápido y más importante llegará en primer lugar. Brindando un mayor nivel de detalle explica que, la página normalmente se divide en marcadores de posición (como por ejemplo, elementos div), con el contenido de cada marcador de posición descargado de forma independiente. De esta manera, las llamadas del XMLHttpRequest pueden emitirse simultánea o secuencialmente, y la página se carga a medida que las llamadas son retornadas.

La ventaja que destacan Zakas, McPeak y Fawcett [Zakas-McPeak-Fawcett06] a nivel del programador, es que este puede decidir qué se va a descargar y en que momento (tiempo especifico).

La propuesta de Mahemoff [Mahemoff06] es que una página sea dividida en bloques, cada uno de los cuales se descargara individualmente; además aconseja que los bloques pequeños (con contenido inicial) se aseguran que la descarga sea rápida. A modo de ejemplo las divisiones tendrían:

* Un bloque de información general acerca del sitio (nombre, iconos, resumen).
* Un bloque de navegación que muestre los enlaces dentro del sitio.
* Un bloque de información general acerca de la página actual (titulo, pequeñas imágenes)
* Uno o mas bloques con información auxiliar (publicidad, avisos legales).

4.1.1.5 Llamada XMLHttpRequest (XMLHttpRequest Call).

Michael Mahemoff [Mahemoff06] plantea cómo puede comunicarse el navegador con el servidor, partiendo de las características de las aplicaciones AJAX, por ejemplo:

* requerir una comunicación del tipo navegador-servidor, dado que la información que es generada por el usuario debe ser subida (al servidor) e información nueva del servidor debe ser descargada
* tener una idea fluida y contínua, la comunicación entre navegador-servidor debe ser discreta
* ser altamente sensible a respuestas, por lo que las llamadas deben incluir la transferencia de datos mínima
* brindar llamadas asincrónicas, las cuáles permiten que un usuario siga trabajando mientras las llamadas están en progreso (en segundo plano)

La solución que propone al planteo es el uso de objetos XMLHttpRequest para la comunicación navegador-servidor. Una de las restricciones que presenta Javascript, es que carece de un mecanismo portátil para la comunicación de red en general, pero el objeto XMLHttpRequest, el cual es soportado por la mayoría de los navegadores que aceptan Javascript, puede realizar llamadas a su servidor de origen y obtener los resultados. Lo cual permite realizar minuciosas llamadas al servidor y manejar las respuestas como se desee, al contrario de los formularios convencionales, que al ser enviados provocan que la página se actualice completamente.

4.1.1.6 Mensaje XML (XML Message).

Para poder desarrollar este patrón Michael Mahemoff [Mahemoff06] manifiesta la cuestión de cómo poder transferir datos entre el navegador y el servidor, dado que

las aplicaciones AJAX requieren que los mensajes sean transmitidos en ambos sentidos (desde el navegador hacia el servidor y viceversa); y además que tanto el navegador como el servidor tengan acceso al mensaje, esto se refiere a que el formato pueda ser accesible fácilmente en Javascript así como cualquier otro lenguaje que se utilice del lado del servidor. El recurso que en este caso ofrece es el de pasar los mensaje entre navegador y el servidor en formato XML. Como ya se ha mencionado, es posible pasar XML en ambas direcciones: del lado del servidor, es una forma común de describir los datos estructurados como texto; y del lado del navegador, las aplicaciones pueden descargar el XML con una llamada XMLHttpRequest para analizarlo de diferentes maneras. En el caso que el navegador necesite subir algún dato complejo, resulta fácil serializar el dato como un XML y subirlo en el cuerpo de una llamada XMLHttpRequest.

4.1.1.7 Arrastrar-y-Soltar (Drag-And-Drop).

En este patrón Michael Mahemoff [Mahemoff06] teoriza a partir de un ejemplo de examen de opción múltiple online, en el cual para responder las preguntas se provee una lista de sucesos históricos que requieren ser ubicados en un orden cronológico, de ésta manera, la respuesta se formulará a partir de mover los elementos hacia arriba y abajo hasta obtener la lista ordenada. Lo que aquí se cuestiona es como poder reorganizar los objetos en la página, ya que las relaciones entre los objetos (especialmente las visuales) resultan de gran importancia y las acciones que llevan a cabo los usuarios, por ejemplo la manipulación visual de la estructura, implican cambios es las mismas. Para resolver esta problemática acude al mecanismo de arrastrar-y-soltar que permite que el usuario reorganice directamente los elementos de la página. La sugerencia de Mahemoff es definir una región específica, que será el lugar donde se encuentren los elementos para ser levantados y posteriormente arrastrados (además de otra región que albergara los elementos que hayan sido arrastrados).

Según su punto de vista, existen pocas formas de implementar el arrastrar-y-soltar; ya sea reutilizar una librería existente, algunas de las cuales se presentan muy portables y poderosas o, aprovechar el arrastrar-y-soltar integrado la cual no es muy buena opción ya que sólo el Internet Explorer de Windows da soporte de forma explicita.

4.1.1.8 Ventana Emergente (Popup).

A menudo durante la navegación de un página, el usuario desea obtener mas información de un determinado elemento evitando desviarse de la página en la que se encuentra, partiendo de esto Michael Mahemoff [Mahemoff06] plantea como hacer que el usuario realice un tarea rápidamente sin tener que distraerse de la navegación principal, para lo cual propone el uso de Ventanas Emergentes temporales para la rápida realización de búsquedas y tareas, es decir, bloques de contenido que aparecen frente al contenido estándar. Existen distintos tipos de ventanas emergentes pero todos ofrecen lo mismo, mayor información con respecto a lo que se busca. Las formas de abrir una ventana emergente son, ya sea al hacer un click o al posicionándose con el puntero (del mouse) por un momento sobre el objeto que incluya el contenido de la ventana emergente. Mientras que las formas de cerrar una ventana emergente son, haciendo click en un botón especifico de cerrado (ejemplo una X en la esquina superior de la ventana) o, si la ventana emergente fue iniciada por colocar el puntero un tiempo determinado, se cerrara cuando se le aleje el puntero o, tras un tiempo de espera.

4.1.1.9 Tiempo de Espera (Time Out).

Una conducta muy común entre los usuarios, que trabajan con aplicaciones web las cuales reciben actualizaciones por parte del servidor, es la de dejar inactiva o abandonar la aplicación que se encuentra en el navegador por horas y a veces días. Con objetivo de brindar seguridad al perfil del usuario y la sensibilidad de los datos que maneja; Michael Mahemoff [Mahemoff06] sugiere que el navegador tenga un Tiempo de Espera para el usuario después de un periodo de inactividad, y que opcionalmente se le informe al servidor. Después del periodo de inactividad, la aplicación se suspende, solicitando que el usuario continúe utilizándola o la cierre.

Michael Mahemoff comenta que una vez que se detecta que el usuario está inactivo, el tiempo de espera puede utilizarse de varias formas, por ejemplo: parando la actualización periódica; borrando los datos, ya sea del perfil de usuario o datos con los que este trabaja, así como también las cookies; guardando los datos, antes de borrarlos se guardan en el servidor; informando al usuario, acerca de las actividades que se están realizando, de modo que sepa qué está pasando y qué debería hacer; y opcionalmente informándole al servidor, para que este actualice el registro del usuario tomando a cada petición entrante como signo de actividad de los usuarios.

El tiempo de espera consiste en un temporizador con cuenta regresiva, y donde cada actividad del usuario, por poco importante que sea, cancela el temporizador e inicia uno nuevo en su lugar.

5. SEGURIDAD.

5.1 Cuestiones de seguridad.

Así como AJAX brinda mejoras a las aplicaciones web a nivel de usabilidad, velocidad, dinamismo y atractivo con respecto a las tradicionales, dio lugar también a la creación de posibles ataques o amenazas si es que no se ha tenido en cuenta un adecuado diseño de seguridad. Dado que AJAX puede encontrarse tanto en el cliente como en el servidor, ambas partes suelen verse afectadas. A continuación se mencionaran algunas de las cuestiones que deben de tenerse en cuenta al momento de diseñar una aplicación AJAX.

5.1.1 Código de Sitio-Entrecruzado (Cross-Site Scripting - XSS).

Cross-Site Scripting (XSS) es un ataque basado en explotar las vulnerabilidades del sistema de validación de una página HTML, mas específicamente al problema de inserción, dado a que permite insertar código malicioso o fragmentos maliciosos en sitios web confiables, de acuerdo a la comunidad que conforma el Proyecto Abierto de Seguridad de Aplicaciones Web (PASAW) [OWASP-XSS].

Las vulnerabilidades en XSS pueden ocurrir cuando; el dato se inserta en una aplicación web a través de una fuente no confiable o, cuando el dato se incluye en contenido dinámico el cual se envía al usuario sin que se rechace el código como malicioso.

La postura de Jaswinder Hayre y Jayasankar Kelath [Hayre-Kelath06] al respecto es que con la introducción de AJAX, los atacantes pudieron explotar las vulnerabilidades de una forma mas encubierta, ya que (por ejemplo) mientras el usuario chequea su correo con una aplicación basada en AJAX, el código malicioso puede estar enviando correos a todos los contactos del usuario sin este lo note visualmente en el navegador.

Mientras que para Billy Hoffman [Hoffman06] gracias a las aplicaciones AJAX, XSS se propago como virus, dado que la carga XSS puede utilizar a las solicitudes de AJAX para insertarse de forma autónoma en las páginas, y fácilmente reinsertarle a la misma máquina más (otras) XSS, en el mismo momento o posteriormente, sin que se realice una actualización. En resumen un código malicioso XSS puede enviar múltiples solicitudes utilizando métodos complejos para propagarse así mismo sin que sea visible por el usuario.

De acuerdo a lo que se expone en el PASAW [OWASP-XSS], el contenido malicioso que se envía al navegador web tiene la forma de un segmento de Javascript, pero puede incluir HTML, Flash o cualquier código que el navegador pueda ejecutar.
La variedad de ataques que se basan en XSS es casi ilimitado, pero entre ellos se pueden detallar; la manipulación de datos privados por ejemplo las cookies o cualquier otra información de sesión los cuales son enviados al atacante, redirigir a la víctima a los contenidos web controlados por el atacante, ejecutar código malicioso en el sistema de la víctima.

5.1.2 Aumento en la Superficie de Ataque.

Billy Hoffman [Hoffman06] expone que a diferencia de las aplicaciones web tradicionales que residen en el servidor, las aplicaciones AJAX se entienden en el cliente y el servidor, el llevar a cabo esta implementación por parte de AJAX requiere una relación de confianza entre el cliente y el servidor, la cual en muchas ocasiones es aprovechada por un atacante.
Además agrega que una aplicación AJAX, envía muchas solicitudes pequeñas con el propósito de crear más entradas en la aplicación. Esas entradas ofrecen más vías de ingreso para que un atacante ingrese a la aplicación.

Para Jaswinder Hayre y Jayasankar Kelath [Hayre-Kelath06] AJAX también aumenta inevitablemente la complejidad general del sistema, puesto que al momento de adoptarlo, los programadores pueden codificar un gran numero de páginas del lado del servidor, donde cada página lleva a cabo una pequeña función (por ejemplo, el autocompletado del código postal a partir de la ciudad y provincia del usuario) en lo que respecta a la aplicación general. Esas pequeñas páginas representan un objetivo adicional para el atacante y un punto a tener en cuenta a nivel seguridad dado que representa una nueva vulnerabilidad.

Tanto Hoffman [Hoffman06] como Hayre y Kelath [Hayre-Kelath06] coinciden en la analogía donde suponen que una aplicación web tradicional representa una casa sin ventanas pero con una puerta trasera y otra delantera, por lo que los atacantes tienen dos puntos de ingreso; mientras que una aplicación AJAX representa una casa con muchas ventanas pero con puertas bloqueadas, aún así el atacante tiene muchas mas vías de ingreso que en la alternativa anterior.

5.1.3 Solicitud Falsa de Sitio-Entrecruzado (Cross-Site Request Forgery - XSRF).

Cross-Site Request Forgery (XSRF) es un viejo ataque en el que un navegador puede ser forzado a realizar solicitudes GET o POST a dominios-entrecruzados (cross-domains); solicitudes que pueden desencadenar un evento en la lógica de la aplicación que se esté ejecutando en el dominio-entrecruzado, así lo expone Shreeraj Shah [Shah11/06].

Desde la postura que ofrece la comunidad del PASAW [OWASP-XSRF], este ataque engaña a la víctima cargándole una página que contiene solicitudes maliciosas, esto se refiere al sentido que hereda la identidad y privilegios de la víctima para realizar una función no deseada en nombre de la víctima. Estos ataques generalmente apuntan a las funciones que causan un cambio de estado en el servidor pero también pueden utilizarse para acceder a datos confidenciales.

Ambas posturas coinciden en los objetivos de los ataques que perjudican al usuario, ya que estas solicitudes pueden ser para el cambio de una dirección de correo o una contraseña, cierre de sesión, cambio de información de una cuenta, recuperar información de cuentas, o cualquier otra función que proporcione el sitio web vulnerable.

Shreeraj Shah [Shah11/06] detalla que cuando el navegador realiza esta llamada se reproduce la cookie y adopta una identidad, siendo este el punto clave en la solicitud. Ya que si la aplicación evalúa a la cookie por si misma, puede concretarse el ataque.

Según lo que expone el PASAW [OWASP-XSRF], es posible almacenar los ataques XSRF en el mismo sitio vulnerable, ya sea simplemente guardando por ejemplo una etiqueta IMG en un campo que acepte HTML o por medio de un ataque complejo del tipo cross-site scripting. De modo que si el ataque se almacena en el sitio, la severidad del mismo se amplifica.

Por su parte Shreeraj Shah [Shah11/06] agrega que dada la comunicación que existe entre las aplicaciones AJAX y los servicios web, es posible realizar llamadas de sitio-entrecruzado (cross-site) a los mismos, lo cual llevaría a comprometer al perfil de la víctima con los servicios web relacionados.

5.1.4 Inserción de XPATH.

Según lo detallado por la comunidad del PASAW [OWASP-XPATH], la inserción XPath ocurre cuando un sitio web utiliza la información suministrada por el usuario para construir una consulta XPath para datos XML. Al enviar intencionalmente la información errónea al sitio web, un atacante puede determinar cómo están estructurados los datos XML o tener acceso a los mismos y aumentar los privilegios del sitio web, si se trata de los datos XML para la autenticación de usuario.

Por su parte, Shreeraj Shah [Shah10/06] comenta que XPATH permite realizar consultas a documentos XML, especificándole ciertos parámetros a buscar (atributos, patrones).
Además agrega que, las aplicaciones web consumen documentos XML largos y muchas veces toman entradas del usuario y forman declaraciones XPath, exponiendo de esta manera su vulnerabilidad. Dado que si la inserción de XPath se ejecuta exitosamente, el atacante puede omitir los mecanismos de autenticación o llevar a la pérdida de información confidencial.

La forma de bloquear el ataque, que propone Shah [Shah10/06], es por medio de una adecuada validación de entrada antes de pasar los valores a una declaración XPath.

De acuerdo con lo expuesto el PASAW [OWASP-XPATH], agrega que esa entrada debe ser depurada para verificar que no estropee la consulta XPath y retorne datos incorrectos.

5.1.5 Ejecución de Código AJAX Malicioso.

Shreeraj Shah [Shah10/06] explica que debido a que las llamadas de AJAX, por medio de objetos XMLHttpRequest, son muy silenciosas, un usuario no podría determinar si el navegador está realizándolas. Cuando un navegador realiza una llamada AJAX a algún sitio web, se reproduce una cookie por cada solicitud.

Esto puede dar lugar a potenciales oportunidades en materia de inseguridad, ya que cuando un usuario navega en una página en la que requiere conectarse y autenticarse con el servidor, se genera una cookie de sesión. Si el usuario continua navegando por otras páginas mientras su sesión esta abierta y llega a una página web de un atacante, la cual tiene escrito código AJAX silencioso, ésta puede realizar llamadas en segundo plano sin el consentimiento del usuario, obteniendo información critica de las páginas del mismo y enviándola al sitio web del atacante, pudiendo dar lugar a un problema de seguridad y a la pérdida de información confidencial.

5.1.6 Infección de XML.

De acuerdo a lo que expone Shreeraj Shah [Shah10/06] muchas aplicaciones web 2.0 hacen uso de bloques de XML, provenientes de clientes AJAX, para la comunicación entre navegadores y servidores. Es posible infectar un bloque de XML, ya que los atacantes pueden producir documentos XML con formato erróneo que puede alterar la lógica dependiendo del mecanismo de análisis que se utilice en el servidor.

Ampliando el alcance de este ataque Shah [Shah11/06] comenta que esta vulnerabilidad se traslada hacia los servicios web quienes en definitiva consumen mensajes XML, dando lugar a que los atacantes puedan insertar contenido malicioso.

5.1.7 Inserción de RSS / Atom.

Shreeraj Shah [Shah10/06] - [Shah11/06] detalla en sus artículos la problemática de la redifusión de contenidos web comentando que, los suministros de RSS / Atom son un medio común para compartir y actualizar información a través de internet, pudiendo ser utilizadas por aplicaciones web, portales, blogs, etc.
Un suministro (o feed) es un documento XML estándar, el cual puede ser consumido por una aplicación web y enviado al navegador del usuario. Una vez en el navegador el suministro es analizado e insertado en el DOM. Esto representa un caso crítico de seguridad, ya que si el suministro no es validado adecuadamente antes de su inserción, es posible introducir enlaces maliciosos o código Javascript para generar un ataque en el navegador (dando lugar a un ataque XSS). Por lo que es de suma importancia filtrar ciertos caracteres del lado del cliente antes del envio de datos al usuario final.

6. EJEMPLO PRÁCTICO EN TECNOLOGÍA AJAX.

El siguiente ejemplo consiste en una muy simple aplicación basada en la tecnología AJAX y los elementos que la conforman, el objetivo de la misma es visualizar la relación entre el cliente y el servidor (en este caso de modo local), a través de la creación de un objeto que permite la comunicación asincrónica entre ambos. Así como también, los efectos visuales y funcionalidades que ofrece por su parte Javascript, el uso del modelo de objetos de documento (DOM) y las hojas de estilo que ofrecen mayor atractivo.

La aplicación consta de varias paginas (ver Anexo A.3), cada una con una funcionalidad concreta y un estilo definido por la hoja de estilo *estilo.css*, la pagina principal es *index.php*, siendo esta la primera que visualiza el usuario al momento de interactuar con la aplicación. Su función es la de solicitar el nombre de usuario y la contraseña, de a quienes está permitidos acceder a la pagina en la cual reside la funcionalidad primordial, de manera de verificar si es posible el acceso a la misma o no. Para ello, se define un formulario en el cual se define el método (método POST) y la acción (redirecciona los valores hacia la pagina *login.php*) que se debe llevar a cabo, el cual se activa al momento que el usuario realiza un click en el botón, una vez ingresados los valores solicitados.

Los valores provenientes de la etapa anterior llegan a *login.php* para ser tratados. Como se puede apreciar, tanto el login como las demás paginas de esta aplicación que interactúan con la base de datos, incluyen a la clase *mysql.php* quien contiene las funciones que le permitirán conectarse a la misma para consultar la información que se desea. Dado que los valores fueron enviados de una pagina hacia la otra por medio del método POST, estos se rastrean por medio de una función propia del lenguaje PHP ($_POST) y del nombre de la etiqueta a la que pertenecían, y se almacenan en variables para luego utilizarlas en la consulta a la base de datos.

La función *md5* permite encriptar la contraseña, solo para brindar mayor seguridad.

Una vez ejecutada la consulta, se verifica si esta tuvo éxito o no, en el caso positivo (por ejemplo que los valores ingresados por el usuario coincidan con los que están en la base de datos) devolverá un resultado que permitirá registrar la sesión de dicho usuario, y redireccionandolo a la pagina *MarcaModeloTipo.php*; en caso contrario solo se alerta, emitiendo un cartel informativo.

En el caso de haberse registrado la sesión, será redireccionado ya al núcleo de la aplicación, dando inicio a la sesión, tomando el recaudo de que si el usuario no esta registrado correctamente se regresa a la pagina de inicio.

MarcaModeloTipo.php incluye varios archivos Javascript para llevar a cabo el dinamismo de la página. En esta se puede apreciar, como se lleva cabo la consulta hacia la base de datos solicitando las marcas de los vehículos que llenaran el combo, para que luego el usuario escoja el que desee. Al momento de insertar los distintos valores en el elemento HTML (los valores *option* del elemento *select*), le asigna al contenedor de los mismos un evento que llama a una función la cual se activa cuando el usuario cambia de un valor a otro en su elección. Esta función (*cargarCombo()*) esta definida dentro del archivo de Javascript (*procesamientoAjax.js*) que se incluyó en la cabecera del documento. Cuando se ejecuta, se le envían los parámetros correspondientes para que pueda procesarlos (la *opcionCombo*, el valor seleccionado por el usuario y el *comboListado*, el combo

que se llenara de datos de acuerdo al resultado de la consulta a partir del valor seleccionado).
Acorde al *comboListado* que se pasa, se selecciona el nombre del archivo que luego procesara los datos en el servidor (el nombre del archivo se guarda en la variable *archivoPhp*). En el caso que el usuario, no escoja una marca determinada (el valor de la opción sea igual a cero), no se va al servidor, sino que se deja la pagina como el estado inicial. En el caso contrario, se crea el objeto XMLHttpRequest (dependiendo del navegador que el usuario disponga) y se le pasa al archivo PHP (es decir, el nombre que esta contenido en *archivoPhp*), utilizando el método GET, el valor de la opción seleccionada (en la primera instancia llama al archivo *procesarModelos.php*).

En *procesarModelos.php*, se toma el valor de la opción seleccionada desde la URL teniendo en cuenta su identificador (haciendo uso de la función $_GET de PHP), y se utiliza para armar la consulta que traerá de la base de datos, los modelos de los vehículos. Los resultados de esta consulta se llenaran en un combo de la misma manera que se explico con anterioridad (asignando un evento que dispara una función ante un cambio). Mientras *procesarModelos.php* se ejecuta, *procesamientoAjax.js* introduce una imagen de espera animada en el combo de *MarcaModeloTipo.php* que se va a listar. Cuando *procesarModelos.php* haya finalizado, este le devuelve la respuesta de la consulta por parte del servidor a *procesamientoAjax.js* y este último lo introduce en el combo que poco tiempo antes mostraba una imagen de espera.
Esta explicación es aplicable cuando el usuario selecciona un valor del combo de modelos, siendo participes de esta acción, *MarcaModeloTipo.php*, *procesamientoAjax.js* y *procesarTipos.php.*

Otra funcionalidad que destaca *MarcaModeloTipo.php* es la inclusión de una pagina que contiene efectos (arrastrar y soltar) producto de Javascript (*dragAndDrop.php*), la cual se puede visualizar al realizar click sobre el elemento *checkbox* quien llama a la función *mostrarDiv()* definida en la cabecera de la

pagina quien se encarga de ocultar o hacer visible el elemento *div* en el cual esta contenida la pagina *dragAndDrop.php*.
Esta pagina fue adaptada para poder introducir mayor atractivo y funcionalidad en esta aplicación ejemplo (ver Anexo A.3 - *dragAndDrop.php*). Las modificaciones a las cuales se sometió consisten en, la inclusión de imágenes en miniatura de una determinada marca de vehículos, reforma de los títulos y la adición de líneas de código (en la cabecera de la pagina `<script type="text/javascript" src="js/popUp.js"></script>`, y en el cuerpo de la misma, dentro de la definición del código Javascript `obj.onmousedown = function(){popUp(idOfDraggedItem)}`), para que una imagen tenga asociada una función teniendo en cuenta el gesto de mouse.

El propósito de *dragAndDrop.php* es permitir arrastrar una imagen (manteniendo presionado en botón del mouse) desde un área de la pagina (definida por un elemento div) hacia otra área distinta (elemento div con diferente valor de id), pudiendo soltar la imagen y que esta se posicione donde le sea requerido. Esta funcionalidad da lugar, a que se pueda arrastrar la imagen desde la izquierda hacia la derecha y viceversa. Con la excepción que, en el área derecha al momento de soltar la imagen, el usuario tiene la opción de volver a arrastrar la imagen si realiza un click, pero si realiza un segundo click se lanzará la función antes mencionada, la cual muestra un mensaje emergente (teniendo en cuenta el nombre del vehículo que se cliqueó). Esto sucede ya que al incluirse el archivo *popUp.js* en esta pagina, se ejecuta la función *popUp*, que esta definida en este ultimo, la que es responsable de llamar a la pagina *popUp.php*. Esta página tiene la función de tomar el valor de la opción seleccionada (el nombre del vehículo para esa imagen), para poder realizar la consulta a la base de datos, de modo que al momento de obtener un resultado de la misma, muestre la imagen del vehículo en tamaño normal, además de los detalles del mismo.

Volviendo a nuestra pagina de mayor importancia, podemos destacar la inclusión del archivo *timeout.js*, quien tiene como propósito, el monitoreo del uso de la

pagina. Esto es, mientras el usuario no pierda el enfoque de la pagina y/o interactúe con la misma a través del movimiento de su mouse, se reiniciará un temporizador para el uso de la aplicación, en el caso que el usuario perdiera la visualización de la pagina o no interactúe con la misma por medio de su mouse, el temporizador seguirá corriendo hasta alcanzar el limite dispuesto (15000 milisegundos, equivale a 15 segundos), emitirá un mensaje alertando la finalización de la sesión y se redireccionará a una pagina (*logout.php*) quien se encargara de destruir la sesión actual y posicionar la URL del navegador, en la pagina principal de logueo.

Estructura de base de datos

Partiendo de esta estructura de base de datos, se detalla el siguiente script:

```
-- phpMyAdmin SQL Dump
-- version 2.11.0
-- http://www.phpmyadmin.net
--
-- Servidor: localhost
-- Tiempo de generación: 04-02-2008 a las 12:46:27
-- Versión del servidor: 5.0.45
-- Versión de PHP: 5.2.4
SET SQL_MODE="NO_AUTO_VALUE_ON_ZERO";
--
-- Base de datos: `ajax`
--
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `detalles`
--
CREATE TABLE `detalles` (
  `id` int(2) unsigned NOT NULL auto_increment,
  `modelo` varchar(30) NOT NULL,
  `anio` int(4) unsigned NOT NULL,
  `tipo` varchar(50) NOT NULL,
  `precio` varchar(10) NOT NULL,
  `combustible` varchar(10) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=5 ;
--
-- Volcar la base de datos para la tabla `detalles`
--
```

```
INSERT INTO `detalles` (`id`, `modelo`, `anio`, `tipo`, `precio`,
`combustible`) VALUES
(1, 'Clio', 2000, 'RN PK AA 5 PUERTAS', '$ 31.800', 'DIESEL'),
(2, 'Logan', 2007, '1.6 CONFORT 4 PUERTAS', '$ 48.000', 'NAFTA'),
(3, 'Megane', 2004, 'RT TD AB  4 PUERTAS', '$ 37.800', 'DIESEL'),
(4, 'Twingo', 2003, 'AUTHENTIQUE 3 PUERTAS', '$ 28.500', 'NAFTA');
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `marcas`
--
CREATE TABLE `marcas` (
  `id` int(2) unsigned NOT NULL auto_increment,
  `marca` varchar(20) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=7 ;
--
-- Volcar la base de datos para la tabla `marcas`
--
INSERT INTO `marcas` (`id`, `marca`) VALUES
(1, 'Chevrolet'),
(2, 'Fiat'),
(3, 'Ford'),
(4, 'Peugeot'),
(5, 'Renault'),
(6, 'Volkswagen');
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `modelos`
--
CREATE TABLE `modelos` (
  `id` int(2) unsigned NOT NULL auto_increment,
  `idMarca` int(2) unsigned NOT NULL,
  `modelo` varchar(30) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=21 ;
--
-- Volcar la base de datos para la tabla `modelos`
--
INSERT INTO `modelos` (`id`, `idMarca`, `modelo`) VALUES
(1, 1, 'Astra'),
(2, 1, 'Corsa'),
(3, 1, 'Zafira'),
(4, 2, 'Duna'),
(5, 2, 'Palio'),
(6, 2, 'Uno'),
(7, 3, 'F-100'),
(8, 3, 'Fiesta'),
(9, 3, 'Ka'),
(10, 3, 'Mondeo'),
(11, 4, '206'),
(12, 4, '307'),
(13, 4, '504'),
(14, 5, 'Clio'),
(15, 5, 'Logan'),
(16, 5, 'Megane'),
(17, 5, 'Twingo'),
(18, 6, 'Bora'),
```

```
(19, 6, 'Gol'),
(20, 6, 'Senda');
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `tipos`
--
CREATE TABLE `tipos` (
  `id` int(2) unsigned NOT NULL auto_increment,
  `idModelo` int(2) unsigned NOT NULL,
  `tipo` varchar(50) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=52 ;
--
-- Volcar la base de datos para la tabla `tipos`
--
INSERT INTO `tipos` (`id`, `idModelo`, `tipo`) VALUES
(1, 1, '2.0 GL 4 PUERTAS'),
(2, 1, '2.0 GL 5 PUERTAS'),
(3, 2, '1.6 GL 3 PUERTAS'),
(4, 2, '1.6 GL 4 PUERTAS'),
(5, 2, '1.6 GL 5 PUERTAS'),
(6, 3, '2.0 GL 5 PUERTAS'),
(7, 3, '2.0 GLS 16V 5 PUERTAS'),
(8, 4, '1.4 S CONFORT 4 PUERTAS'),
(9, 4, '1.4 S EQUIPADO P/GNC 4 PUERTAS'),
(10, 4, '1.7 SD 4 PUERTAS'),
(11, 5, '1.3 FIRE 3 PUERTAS'),
(12, 5, '1.4 FIRE WAY 5 PUERTAS'),
(13, 5, '1.8 R FULL 3 PUERTAS'),
(14, 6, '1.3 FIRE LN 3 PUERTAS'),
(15, 6, '1.3 FIRE VAN CARGO 3 PUERTAS'),
(16, 7, '3.9 TDI XL 2 PUERTAS'),
(17, 7, '3.9 TDI XLT L/06 2 PUERTAS'),
(18, 7, '3.9 TDI XLT L/06 D/CAB 4X4 4 PUERTAS'),
(19, 8, '1.4 MAX ENERGY TDCI 4 PUERTAS'),
(20, 8, '1.6 FRESH 5 PUERTAS'),
(21, 9, '1.0 TATOO 3 PUERTAS'),
(22, 9, '1.0 TATOO PLUS AA 3 PUERTAS'),
(23, 10, '1.8 TITANIUM TDCI 4 PUERTAS'),
(24, 10, '2.2 ST TDCI 4 PUERTAS'),
(25, 10, '3.0 ST V6 4 PUERTAS'),
(26, 11, '1.4 X-LINE 3 PUERTAS'),
(27, 11, '1.6 XS L/N 3 PUERTAS'),
(28, 11, '2.0 HDI PREMIUM 5 PUERTAS'),
(29, 12, '1.6 XS 4 PUERTAS'),
(30, 12, '2.0 RWC HDI 5 PUERTAS'),
(31, 12, '2.0 CC 180 CV L/N 2 PUERTAS'),
(32, 13, 'GR II 4 PUERTAS'),
(33, 13, 'GR FULL 4 PUERTAS'),
(34, 14, 'RL 3 PUERTAS'),
(35, 14, 'RN PK2 AA 5 PUERTAS'),
(36, 14, 'RT PK2 DH 5 PUERTAS'),
(37, 15, '1.5 DCI LUXE 4 PUERTAS'),
(38, 15, '1.6 CONFORT 4 PUERTAS'),
(39, 16, 'CABRIO 115 HP 2 PUERTAS'),
(40, 16, 'COUPE 150 HP 2 PUERTAS'),
(41, 17, 'AUTHENTIQUE 3 PUERTAS'),
```

```
(42, 17, 'JOVEN 3 PUERTAS'),
(43, 17, 'ESPRESSION AA 3 PUERTAS'),
(44, 18, '1.8 T. HIGHLINE CUERO 4 PUERTAS'),
(45, 18, '1.9 TDI TRENDLINE 4 PUERTAS'),
(46, 18, '2.0 TRENDLINE 4 PUERTAS'),
(47, 19, '1.6 POWER LN AA DH 3 PUERTAS'),
(48, 19, '1.9 TRENDLINE LN 3 PUERTAS'),
(49, 19, '1.9 SD COUNTRY FORMAT 5 PUERTAS'),
(50, 20, '4 PUERTAS'),
(51, 20, 'CONFORT 4 PUERTAS');
-- --------------------------------------------------------
--
-- Estructura de tabla para la tabla `usuarios`
--
CREATE TABLE `usuarios` (
  `id` int(4) NOT NULL auto_increment,
  `usuario` varchar(15) NOT NULL default '',
  `password` varchar(32) NOT NULL default '',
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM  DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;
--
-- Volcar la base de datos para la tabla `usuarios`
--
INSERT INTO `usuarios` (`id`, `usuario`, `password`) VALUES
(1, 'santiago', md5('santy123'));
```

Los distintos ejemplos que se explicaran a lo largo del documento, utilizan una misma clase para poder interactuar con la base de datos antes mencionada, esta clase es *mysql.php*, la cual contiene funciones que permiten la conexión, selección de la base a utilizar, ejecución de consultas y desconexión con la base de datos.

mysql.php

Descripción:

Contiene funciones (tales como conectar, desconectar, seleccionar y realizar una consulta), que permiten a la aplicación interactuar con la base de datos.

```
<?php

class mysql {
 var $servidorMysql = "localhost";
 var $usuario = "root";
 var $contrasena = "";
 var $baseDatos = "ajax";
 var $conexion;
```

```
 var $abierto = false;

 function conectar(){
   $this->conexion = @mysql_connect($this->servidorMysql, $this->usuario,
                                    $this->contrasena) or die("NO ES
                                    POSIBLE LA CONEXION!");
   $this->abierto = true;
   return $this->conexion;
 }

 function desconectar(){
   if($this->abierto == true){
     @mysql_close($this->conexion);
   }
 }

 function select($db){
   mysql_select_db($db)or die("NO ES POSIBLE SELECCIONAR
                               LA BASE DE DATOS!");
 }

 function query($query){
  $this->conectar();
  $this->select($this->baseDatos);
  $results = @mysql_query($query);
  $this->desconectar();
  return $results;
 }
}

?>
```

6.1 Explicación de la interacción sincrónica de una aplicación Web tradicional.

Un ejemplo práctico del modelo clásico de una aplicación web, es el que ofrece *EjemploSincronico.php*, como se puede ver a continuación:

EjemploSincronico.php

Descripción:

Carga un combo desplegable con las marcas de los vehículos (que obtiene por medio de la inclusión de *mysql.php*). El botón "*Listar*" (a partir de la opción seleccionada), obtiene una lista de los distintos modelos de vehículos para una determinada marca.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Ejemplo Aplicación Sincronica</title>
</head>

<body>
<h2>Ejemplo Aplicación Sincronica</h2>
<hr>
<form action="EjemploSincronico.php" method="get">
 <table cellpadding="0" cellspacing="0" border="1px">
  <tr>
   <td width="100px">
    <strong>Marcas:</strong>
   </td>
   <td width="100px">
   <?php
    include ("mysql.php");

    $mysql = new mysql;
    $sql = "SELECT id, marca FROM marcas";
    $consulta = $mysql->query($sql);

    echo "<select name='marca'>";
    echo "<option value='0'>[SELECCIONAR]</option>";
    while($registro = mysql_fetch_row($consulta)){
      echo "<option value='".$registro[0]."'>".$registro[1]."</option>";
    }
    echo "</select>";
   ?>
   </td>
   <td><input type="submit" value="Listar"/></td>
  </tr>
 </table>
</form>

<br>

<table cellpadding="0" cellspacing="0" border="1px">
 <?php
 $mysql = new mysql;
 $opcionSeleccionada = $_GET["marca"];
 $sql = "SELECT marca, modelo FROM marcas mr, modelos md WHERE
       mr.id='$opcionSeleccionada' AND md.idMarca='$opcionSeleccionada'";
 $consulta = $mysql->query($sql);

 $flag = 1;
 while($registro = mysql_fetch_row($consulta)){
  if($flag == 1){
    echo "<tr><td width='100px'><strong>".$registro[0]."</strong>
         </td></tr><tr><td width='100px'>".$registro[1]."</td></tr>";
    $flag = 0;
  }else{
    echo "<tr><td width='100px'>".$registro[1]."</td></tr>";
  }
 }
```

```
 ?>
</table>
</html>
```

De acuerdo a como se explica en la figura 3, la comunicación entre el cliente y el servidor se da a partir de la actividad por parte del usuario, la cual involucra la transmisión de datos hacia el servidor para que sean procesados y el resultado del procesamiento es retornado nuevamente al cliente, quien depende de ésta respuesta para continuar con su tarea.

En la primera parte del código PHP, se incluye a la clase *mysql.php*, se crea un objeto del tipo *mysql* para poder realizar la consulta a la base de datos (solicitando las marcas de los vehículos), el resultado de la consulta se almacena en una variable que luego será mostrada en la pagina (en el combobox).
En la segunda parte, dependiendo de la opción seleccionada, se lleva a cabo una nueva consulta a la base de datos, la cual surge del disparo de un evento por parte del usuario (click en el botón "Listar"). El resultado proveniente de la base de datos, acorde a lo seleccionado, se dibuja en formato de tabla (lista de modelos), dando lugar a un recarga total de la información de la pagina.

Figura 8: Ejemplo Aplicación Sincrónica

6.2 Explicación de la interacción asincrónica de una aplicación AJAX.

Por parte del modelo AJAX de una aplicación web, el ejemplo practico esta dado por la comunicación entre el conjunto de 3 archivos *EjemploAsincronico.php*, *Ajax.js* y *listarTabla.php*, en mayor detalle:

EjemploAsincronico.php

Descripción:

Carga un combo desplegable con las marcas de los vehículos (que obtiene por medio de la inclusión de *mysql.php*). Al seleccionar una opción (cambiando el valor de carga predeterminado, [*SELECCIONAR*], por el de una marca), dispara un evento que obtiene una lista de los distintos modelos de vehículos para una determinada marca.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title>Ejemplo Aplicación Asincronica</title>
 <script type="text/javascript" src="Ajax.js"></script>
</head>

<body>
<h2>Ejemplo Aplicación Asincrónica</h2>
<hr>
<table cellpadding="0" cellspacing="0" border="1px">
 <tr>
  <td width="100px">
   <strong>Marcas:</strong>
  </td>
  <td width="100px">
   <?php
    include ("mysql.php");

    $mysql = new mysql;
    $sql = "SELECT id, marca FROM marcas";
    $consulta = $mysql->query($sql);

    echo "<select name='marca' id='marca'
            onChange='listarTabla(\"marca\")'>";
    echo "<option value='0'>[SELECCIONAR]</option>";
    while($registro = mysql_fetch_row($consulta)){
      echo "<option value='".$registro[0]."'>".$registro[1]."</option>";
    }
    echo "</select>";
   ?>
  </td>
 </tr>
</table>

<br>

<table id="lista" cellpadding="0" cellspacing="0" border="1px">
</table>
</html>
```

Ajax.js

Descripción:

Contiene 2 funciones; *crearObjXHR*, que crea un objeto AJAX; y *listarTabla*, que a partir de la marca (seleccionada en *EjemploAsincronico.php*) y del objeto creado, invoca al archivo que realiza la consulta a la base de datos, para poder mostrar el resultado como una lista.

```
function listarTabla(marca){
 indiceOpcion = document.getElementById(marca);
 texto = indiceOpcion.options[indiceOpcion.selectedIndex].text;
 valor = indiceOpcion.options[indiceOpcion.selectedIndex].value;
 listado = document.getElementById("lista");
 opcion = document.createElement("option");

 if(valor == 0){
  opcion.value = 0;
  opcion.innerHTML = "[SELECCIONAR]";
 }else{
  objAjax = crearObjXHR();
  objAjax.open("GET", "listarTabla.php?marca="+texto+"&id="+valor, true);
  objAjax.onreadystatechange = function(){
     if (objAjax.readyState == 4){
       if (objAjax.status == 200) {
         listado.innerHTML = objAjax.responseText;
       }
     }
  }
 objAjax.send(null);
 }
}

function crearObjXHR(){
 if (window.XMLHttpRequest){
   xhr = new XMLHttpRequest();
 }else if(window.ActiveXObject){
   xhr = new ActiveXObject("Microsoft.XMLHTTP");
 }
 return xhr;
}
```

listarTabla.php

Descripción:

Realiza una consulta a la base de datos, a partir del valor de la marca que obtiene de la URL (por medio del método _GET), solicitando los modelos de la misma. Luego recorre la consulta, mostrando los valores resultantes en una tabla que elabora dinámicamente.

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

 $textoMarca = $_GET["marca"];
 $valorMarca = $_GET["id"];
 $mysql = new mysql;
```

```
 $sql = "SELECT modelo FROM modelos WHERE idMarca='$valorMarca'";
 $consulta = $mysql->query($sql);

 echo "<tr><td width='100px'><strong>".$textoMarca."</strong></td></tr>";

 while($registro = mysql_fetch_row($consulta)){
   echo "<tr><td width='100px'>".$registro[0]."</td></tr>";
 }
?>
```

De acuerdo a como se explica en la figura 4, entre la comunicación del cliente y del servidor, existe una capa intermedia que es la del motor AJAX.
Partiendo de la actividad del usuario, se envían datos hacia el motor AJAX quien se encarga del pre-procesamiento de los mismos, luego son transmitidos al servidor para su posterior procesamiento.
El motor AJAX le envía al usuario una señal de espera hasta obtener el resultado proveniente del servidor, durante este periodo el usuario puede continuar utilizando la aplicación sin la necesidad de obtener una respuesta inmediata.

Al igual que en el ejemplo anterior, en la primera parte del código PHP, se incluye a la clase *mysql.php*, se crea un objeto del tipo *mysql* para poder realizar la consulta a la base de datos (solicitando las marcas de los vehículos), el resultado de la consulta se almacena en una variable que luego será mostrada en la pagina (en el combobox), pero en este caso cada elemento del combo tendrá asociado una función Javascript la cual se disparará cuando el usuario cambie de una elemento al otro.
Esta función (Ajax.js) crea un objeto AJAX (objeto XMLHttpRequest), el cual enviará una solicitud, dependiendo de la opción que haya sido seleccionada, hacia una función en PHP (listarTabla.php) que se conectara con base de datos por medio de una nueva consulta. El resultado proveniente de la base de datos, será interpretado por esta función y devuelto al objeto AJAX, quien a su vez lo entregará al navegador. En este caso, la asincronía da lugar a la recarga parcial de la información de la pagina.

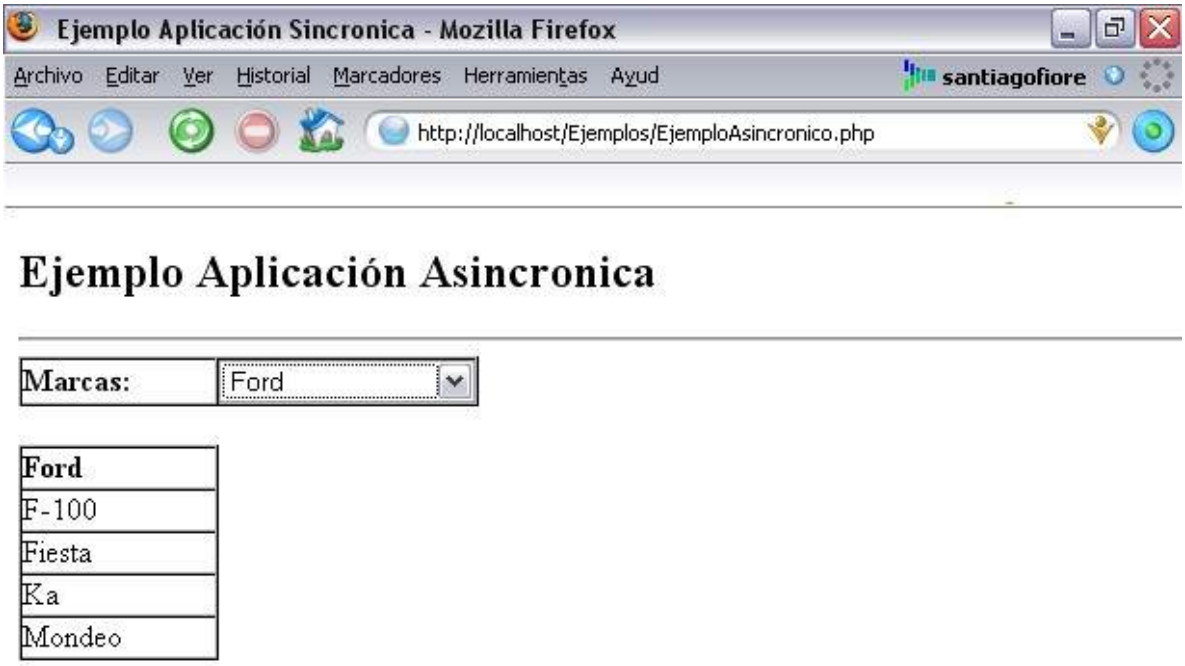


Figura 9: Ejemplo Aplicación Asincrónica

6.3 Archivos del ejemplo practico en tecnología AJAX.

estilos.css

```
body {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 11px;
color: #333333;
background-color:#E3F2F9;
}

h1 {
font-size:16px;
line-height: 1em;
}

h2 {
font-size:15px;
line-height: 1em;
font-weight: 900;
}
```

```
h3 {
font-size:14px;line-height: 1em;
font-weight: 900;
}

h4 {
font-size:13px;
line-height: 1em;
font-weight: 900;
}

h5 {
font-size:12px;
line-height: 1em;
font-weight: 900;
}

h6 {
font-size:11px;
line-height: 1em;
font-weight: 900;
}

td{
font-size: 12px;
background-color:#E3F2F9
}

th{
font-size: 12px;
}

a{
color: #CC6600;
}

div{
background-color:#E3F2F9
}

a:hover{
color: #659FDE;
}

a.visted{
color: #CC6600;
}

a.image, a.image:hover{
border:0;
}

li{
list-style: square;
}

ul{
```

```
list-style: square;
}
```

index.php

Descripción:

Solicita el nombre de usuario y contraseña, para poder validar estos valores con los que están en la base de datos.

```
<html>
<head>
 <title>Login de Usuarios</title>
 <link href="css/estilos.css" rel="stylesheet" type="text/css"
media="screen">
</head>

<body>
<table width="300" border="0" align="center" cellpadding="0"
cellspacing="1" bgcolor="#CCCCCC">
 <tr>
  <form name="login" method="post" action="login.php">
   <td>
    <table width="100%" border="0" cellpadding="3" cellspacing="1">
     <tr>
      <td colspan="2" align="center">
       <strong>Login de Usuarios</strong>
      </td>
     </tr>
     <tr>
      <td width="78">Usuario:</td>
      <td width="294">
       <input name="user" type="text">
      </td>
     </tr>
     <tr>
      <td>Password:</td>
      <td>
       <input name="pass" type="password">
      </td>
     </tr>
     <tr>
      <td colspan="2" align="center">
       <input name="Submit" type="submit" value="Login">
      </td>
     </tr>
    </table>
   </td>
  </form>
 </tr>
```

```
</table>
</body>
</html>
```

login.php

Descripción:

Obtiene los valores del formulario (por medio del método _POST), correspondientes al nombre de usuario y contraseña, y con ellos realiza una consulta a la base de datos para verificar su existencia en la misma. En caso de éxito, registra la sesión con los valores en cuestión y lo redirecciona a la pagina donde se encuentra la aplicación; en caso contrario, muestra un mensaje de alerta.

```
<html>
<head>
 <title>Login de Usuarios</title>
 <link href="css/estilos.css" rel="stylesheet" type="text/css"
media="screen">
</head>

<body>
 <?php
  include ("clases/mysql.php");

  $user = $_POST['user'];
  $pass = md5($_POST['pass']);

  $mysql = new mysql;
  $sql = "SELECT * FROM usuarios WHERE usuario='$user' and
          password='$pass'";
  $consulta = $mysql->query($sql);

  $count = mysql_num_rows($consulta);
  if($count == 1){
    session_register("user");
    session_register("pass");
    header("location:MarcaModeloTipo.php");
  }else{
    echo "<h1>USUARIO O PASSWORD INCORRECTO!</h1>";
  }
 ?>
</body>
</html>
```

MarcaModeloTipo.php

Descripción:

Inicia la sesión y verifica que el usuario este registrado en la misma, en caso contrario lo redirecciona hacia la pagina principal. Llena el combo de Marcas con las distintas marcas de vehículos que obtiene de la consulta a la base de datos, para que luego pueda ser utilizada. Contiene la función *mostrarDiv* que permite ocultar o visualizar un elemento donde esta embebida una página.

```
<?php
 session_start();
 if(!session_is_registered(user)){
  header("location:index.php");
 }
?>
<html>
<head>
 <title>Marca - Modelo - Tipo</title>
 <link href="css/estilos.css" rel="stylesheet" type="text/css"
media="screen">

 <script type="text/javascript" src="js/procesamientoAjax.js"></script>
 <script type="text/javascript" src="js/timeout.js"></script>
 <script>
  function mostrarDiv(){
  var obj = document.getElementById("divDragAndDrop");
  if(obj.style.visibility == "hidden"){
    obj.style.visibility = "visible";
  }else{
    obj.style.visibility = "hidden";
  }
  }
 </script>
</head>

<body>
<table cellpadding="0" cellspacing="0" border="1px">
 <tr>
  <td width="100px">
   <strong>Marcas:</strong>
  </td>
  <td width="100px">
   <div id="divMarca">

  <?php
   include ("clases/mysql.php");

   $mysql = new mysql;
```

```
   $sql = "SELECT id, marca FROM marcas";
   $consulta = $mysql->query($sql);

   echo "<select id='marca' name='marca' onChange='cargarCombo(\"marca\",
          \"modelo\")'>";
   echo "<option value='0'>[SELECCIONAR]</option>";
   while($registro = mysql_fetch_row($consulta)){
     echo "<option value='".$registro[0]."'>".$registro[1]."</option>";
   }
   echo "</select>";
  ?>

   </div>
  </td>

  <td width="100px">
   <strong>Modelos:</strong>
  </td>
  <td width="100px">
   <div id="divModelo">
    <select id="modelo" name="modelo" disabled="disabled">
      <option value="0">--------------------------</option>
    </select>
   </div>
  </td>

  <td width="100px">
   <strong>Tipos:</strong>
  </td>
  <td width="100px">
   <div id="divTipo">
    <select id="tipo" name="tipo" disabled="disabled">
      <option value="0">--------------------------</option>
    </select>
   </div>
  </td>
 </tr>
</table>
<br>
<table width="200" border="0">
 <tr>
  <td>
   <strong>
    <input id="checkbox" type="checkbox" onChange="mostrarDiv()">Drag
                                                                And Drop
   </strong>
  </td>
 </tr>
 <tr>
  <td>
   <div id="divDragAndDrop" style="visibility:hidden">
     <iframe src="dragAndDrop.php" height="560" width="630"></iframe>
   </div>
  </td>
 </tr>
</table>
```

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

procesamientoAjax.js

Descripción:

Contiene 3 funciones; *crearObjXHR*, que crea un objeto AJAX; *deshabilitarTipo*, que deshabilita el combo que no tiene información y *cargarCombo*, que a partir de una opción seleccionada del combo (seleccionada en *MarcaModeloTipo.php*) y del objeto creado, invoca al archivo que realiza la consulta a la base de datos, para poder mostrar el resultado en el combo correspondiente.

```
function cargarCombo(opcionCombo, comboListado){
 indiceOpcion = document.getElementById(opcionCombo);
 valorOpcion = indiceOpcion.options[indiceOpcion.selectedIndex].value;
 listado = document.getElementById(comboListado);
 opcion = document.createElement("option");

 if(comboListado == 'modelo'){
   archivoPhp =  "procesarModelos.php?id=";
   deshabilitarTipo();
 }else{
   archivoPhp =  "procesarTipos.php?id=";
 }

if(valorOpcion == 0){
   listado.length = 0;
   opcion.value = 0;
   opcion.innerHTML = "[SELECCIONAR]";
   listado.appendChild(opcion);
   listado.disabled = true;
 }else{
   objAjax = crearObjXHR();
   objAjax.open("GET", archivoPhp+valorOpcion, true);
   objAjax.onreadystatechange = function(){
     if (objAjax.readyState == 1){
       listado.length = 0;
       opcion.value = 0;
       opcion.innerHTML = '<img src="images/loading.gif"/>';
       listado.appendChild(opcion);
       listado.disabled = true;
     }
     if (objAjax.readyState == 4){
       listado.parentNode.innerHTML = objAjax.responseText;
     }
```

```
   }
   objAjax.send(null);
 }
}

function deshabilitarTipo(){
 tipo = document.getElementById('tipo');
 tipo.length = 0;
 tipo.options[0] = new Option("-------------------------", 0);
 tipo.disabled = true;
}


function crearObjXHR(){
 if (window.XMLHttpRequest){
   xhr = new XMLHttpRequest();
 }else if(window.ActiveXObject){
   xhr = new ActiveXObject("Microsoft.XMLHTTP");
 }
 return xhr;
}
```

procesarModelos.php

Descripción:

Realiza una consulta a la base de datos, a partir del valor de la marca que obtiene de la URL (por medio del método _GET), solicitando los modelos de la misma. Luego recorre la consulta, armando un combo con los valores (en código HTML) y lo entrega al archivo que lo solicitó.

```
<?php
 include ("clases/mysql.php");

 $opcionSeleccionada = $_GET["id"];
 $mysql = new mysql;
 $sql = "SELECT id, modelo FROM modelos WHERE
         idMarca='$opcionSeleccionada'";
 $consulta = $mysql->query($sql);

 echo "<select id='modelo' name='modelo'
        onchange='cargarCombo(\"modelo\", \"tipo\")'>";
 echo "<option value='0'>[SELECCIONAR]</option>";
 while($registro = mysql_fetch_row($consulta)){
   $registro[0] = htmlentities($registro[0]);
   echo "<option value='".$registro[0]."'>".$registro[1]."</option>";
 }
```

```
 echo "</select>";
?>
```

procesarTipos.php

Descripción:

Realiza una consulta a la base de datos, a partir del valor del modelo que obtiene de la URL (por medio del método _GET), solicitando los tipos del mismo. Luego recorre la consulta, armando un combo con los valores (en código HTML) y lo entrega al archivo que lo solicitó.

```
<?php
 include ("clases/mysql.php");

 $opcionSeleccionada = $_GET["id"];
 $mysql = new mysql;
 $sql = "SELECT tipo FROM tipos WHERE idModelo='$opcionSeleccionada'";
 $consulta = $mysql->query($sql);

 echo "<select id='tipo' name='tipo'>";
 echo "<option value='0'>[SELECCIONAR]</option>";
 while($registro = mysql_fetch_row($consulta)){
   $registro[0] = htmlentities($registro[0]);
   echo "<option value='".$registro[0]."'>".$registro[0]."</option>";
 }
 echo "</select>";
?>
```

dragAndDrop.php

Descripción:

Permite arrastrar y soltar (por medio de un click), las imágenes de los vehículos desde un contenedor al otro (contenedor izquierdo al derecho y viceversa). El contenedor derecho permite con un segundo click, abrir una ventana informativa acerca del vehículo.

Sección *Drag and Drop*, Demo 2 del ejemplo *“Custom drag and drop script”*.

Fuente: http://www.dhtmlgoodies.com/index.html?page=dragDrop

```
<html>
<head>
 <title>Drag And Drop</title>
 <link rel="stylesheet" href="css/demos.css" media="screen"
type="text/css">

 <script type="text/javascript" src="js/dragAndDrop.js"></script>
 <script type="text/javascript" src="js/popUp.js"></script>
</head>

<body>
<div id="mainContainer">
 <h2>Renault</h2>
  <div id="leftColumn">
   <p><b>Arrastrar</b></p>
    <div id="dropContent">
      <div id="Clio"><img src="images/thumbnails/Clio.jpg"></div>
      <div id="Logan"><img src="images/thumbnails/Logan.jpg"></div>
      <div id="Megane"><img src="images/thumbnails/Megane.jpg"></div>
      <div id="Twingo"><img src="images/thumbnails/Twingo.jpg"></div>
    </div>
  </div>
  <div id="rightColumn">
   <div id="dropBox" class="dropBox">
     <p><b>Soltar y PopUp (2do click)</b></p>
     <div id="dropContent2"></div>
   </div>
  </div>
  <div class="clear"></div>
</div>

<script type="text/javascript">
// Custom drop actions for <div id="dropBox"> and <div id="leftColumn">
 function dropItems(idOfDraggedItem,targetId,x,y){
  if(targetId=='dropBox'){    // Item dropped on <div id="dropBox">
    var obj = document.getElementById(idOfDraggedItem);
    if(obj.parentNode.id=='dropContent2'){
      obj.onmousedown = function(){popUp(idOfDraggedItem)}
      return;
    }
 document.getElementById('dropContent2').appendChild(obj);
// Appending  dragged element as child of target box
  }
  if(targetId=='leftColumn'){ // Item dropped on <div id="leftColumn">
     var obj = document.getElementById(idOfDraggedItem);
     if(obj.parentNode.id=='dropContent')return;
     document.getElementById('dropContent').appendChild(obj);
// Appending dragged element as child of target box
  }
 }

 function onDragFunction(cloneId,origId){
  self.status = 'Started dragging element with id ' + cloneId;
  var obj = document.getElementById(cloneId);
```

```
  obj.style.border='1px solid #F00';
 }

 var dragDropObj = new DHTMLgoodies_dragDrop();
// Make <div id="boxX"> dragable. slide item back into original position
after drop
 dragDropObj.addSource("Clio",true,true,true,false,'onDragFunction');
 dragDropObj.addSource("Logan",true,true,true,false,'onDragFunction');
 dragDropObj.addSource("Megane",true,true,true,false,'onDragFunction');
 dragDropObj.addSource("Twingo",true,true,true,false,'onDragFunction');

 dragDropObj.addTarget('dropBox','dropItems');
// Set <div id="dropBox"> as a drop target. Call function dropItems on
drop
 dragDropObj.addTarget('leftColumn','dropItems');
// Set <div id="leftColumn"> as a drop target. Call function dropItems on
drop
 dragDropObj.setSlide(true); //Activate sliding animation
 dragDropObj.init();
</script>
</body>
</html>
```

popUp.js

Descripción:

Contiene una función *popUp*, que permite desplegar una ventana, con la información del vehículo.

```
function popUp(id) {
 var prop="toolbar=no, status=no, menubar=no, scrollbars=no,
           resizable=no, width=475, height=400, top=200, left=300";
 window.open("popup.php?id="+id,"",prop);
}
```

popUp.php

Descripción:

Realiza una consulta a la base de datos, a partir del valor del modelo que obtiene de la URL (por medio del método _GET), solicitando los detalles del

mismo (como ser foto, modelo, año, tipo, precio y combustible). Luego arma una tabla con los valores (en código HTML) y lo entrega al archivo que lo solicitó.

```
<html>
<head>
 <title>PopUp</title>
 <link href="css/estilos.css" rel="stylesheet" type="text/css"
media="screen">
</head>

<body>
<table cellpadding="0" cellspacing="0" border="1px">
 <tr>
  <td colspan="2" width="100px">

  <?php
   include ("clases/mysql.php");

   $opcionSeleccionada = $_GET["id"];
   $mysql = new mysql;
   $sql = "SELECT modelo, anio, tipo, precio, combustible FROM detalles
           WHERE modelo = '$opcionSeleccionada'";
   $consulta = $mysql->query($sql);

   $registro = mysql_fetch_row($consulta);

   echo "<img src='images/pictures/".$registro[0].".jpg'>";
   echo " </td>";
   echo "</tr>";

   echo "<tr>";
   echo " <td ><strong>Modelo:</strong></td>";
   echo " <td>".$registro[0]."</td>";
   echo "</tr>";

   echo "<tr>";
   echo " <td ><strong>Año:</strong></td>";
   echo " <td>".$registro[1]."</td>";
   echo "</tr>";

   echo "<tr>";
   echo " <td ><strong>Tipo:</strong></td>";
   echo " <td>".$registro[2]."</td>";
   echo "</tr>";

   echo "<tr>";
   echo " <td ><strong>Precio:</strong></td>";
   echo " <td>".$registro[3]."</td>";
   echo "</tr>";

   echo "<tr>";
   echo " <td ><strong>Combustible:</strong></td>";
   echo " <td>".$registro[4]."</td>";
   echo "</tr>";
?>
```

```
</table>
</body>
</html>
```

timeout.js

Descripción:

Contiene 3 funciones; una funciona asignada al evento del mouse (ya sea que se ejecuta por el movimiento del puntero del mouse o porque el puntero pierde el foco de la pagina), que verifica el tiempo que pasa sin que el mouse reciba un estimulo, de manera de borrar y reinicial el tiempo; *setearTimeOut*, establece un tiempo para poder ejecutar una función especifica; *logOut*, produce un mensaje de alerta y redirecciona hacia otra pagina.

```
var timeOut = null;

window.onblur = window.onmousemove = function() {
 if(timeOut) clearTimeout(timeOut);
 setearTimeOut();
}

function setearTimeOut() {
 timeOut = setTimeout("logOut()", 15000);
}

function logOut() {
 alert("Fin de la Sesión por Inactividad.");
 document.location.href = "logout.php";
}
```

logout.php

Descripción:

Redirecciona la hacia la pagina principal y destruye la sesión en la que se encuentra el usuario.

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

CONCLUSIÓN.

Al momento de poner manos a la obra en una aplicación web, se deben analizar las distintas tecnologías con las que uno dispone, si bien a lo largo del texto se da una breve descripción de las mismas, optando por dar mayor importancia a AJAX y haciendo un estudio mas profundo de ella, esto no debe entenderse como que las demás tecnologías son de menor importancia, ya que el uso de cada una de ellas dependerá de la aplicación final que se quiera obtener. Así mismo las distintas tecnologías pueden interactuar complementándose unas a otras.

La tecnología AJAX permite que una aplicación sea dinámica, que cargue contenidos parciales de una pagina, que brinde una mayor interacción del usuario final con la aplicación sin la necesidad de esperar por una respuesta del lado del servidor. Otros de los beneficios que se puede mencionar es la simplicidad en la implementación de las herramientas desarrolladas en Javascript (librerías y frameworks), las cuales ofrecen al usuario un gran atractivo visual y facilidad de uso.

En el proceso de elaboración de una aplicación AJAX, pueden surgir distintos inconvenientes, para la solución de los mismos se pueden utilizar los patrones de diseño, la no utilización de ellos como el uso en exceso puede producir contratiempos.
Si no se han tomado recaudos al momento de diseñar la seguridad de la aplicación web, puede darse lugar a ataques provenientes de piratas informáticos o códigos malintencionados.
El ejemplo final muestra el funcionamiento básico que ofrece la tecnología AJAX, así como también el funcionamiento de distintas herramientas, y soluciones comunes ante la presencia de algún percance.

GLOSARIO.

Actionscript: es un lenguaje de programación orientado a objetos, utilizado en especial en aplicaciones web animadas realizadas en el entorno Flash [GLSR01].

API (Application Programming Interface) o Interfaz de Programación de Aplicaciones: representa una interfaz de comunicación entre componentes software. Se trata del conjunto de llamadas a ciertas bibliotecas que ofrecen acceso a ciertos servicios desde los procesos y representa un método para conseguir abstracción en la programación [GLSR02].

Applet: es un componente de software que se ejecuta dentro del contexto de otro programa, por ejemplo un navegador web; a diferencia de un programa, un applet no puede correr de manera independiente, ofrece información gráfica y a veces interactúa con el usuario [GLSR03].

ASP: es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente [GLSR04].

ATOM: formato XML similar a RSS, nacido para resolver la confusión creada por la existencia de estándares similares para sindicación web (RSS y RDF) y crear una API y un formato de sindicación web más flexibles [GLSR05].

banner (pancartas): formato publicitario en Internet, la cual consiste en incluir una pieza publicitaria dentro de una página web [GLSR06].

blog: sitio web periódicamente actualizado que recopila cronológicamente textos o artículos de uno o varios autores, apareciendo primero el más reciente, donde el autor conserva siempre la libertad de dejar publicado lo que crea pertinente [GLSR07].

buffer (de datos): es una ubicación de la memoria en una computadora reservada para el almacenamiento temporal de información, donde espera para poder ser procesada [GLSR08].

bytecodes: es un código intermedio más abstracto que el código máquina. En detalle, es un fichero binario que contiene código máquina producido por el compilador [GLSR09].

cache: es una porción de memoria añadida a un disco con la utilidad de almacenar los datos recientemente leídos y por lo tanto agilizar la carga de los mismos en caso que estos vuelvan a ser solicitados [GLSR10].

CSS (Cascading Style Sheets) o Hojas de Estilo en Cascada: son un lenguaje formal usado para definir la presentación de un documento estructurado escrito en HTML o XML y por lo tanto en XHTML. La idea que se encuentra detrás del

desarrollo de CSS es separar la estructura de un documento de su presentación [GLSR11].

CMS (Content Management System) o Sistema de Gestión de Contenidos: permite la creación y administración de contenidos principalmente en páginas web. Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio. El sistema permite manejar de manera independiente el contenido y el diseño [GLSR12].

Cookie: fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas [GLSR13].

DOM (Document Object Model) o Modelo de Objetos de Documento: forma de representar los elementos de un documento estructurado (tal como una página web HTML o un documento XML) como objetos que tienen sus propios métodos y propiedades. DOM es una API para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos con lenguajes como Javascript [GLSR14].

feedback: retroalimentación.

GET: método de peticiones HTTP, los valores se pasan por la URL.

gecko: es un motor de interpretación de código abierto diseñado para soportar los estándares abiertos de Internet [GLSR15].

HTML (Hypertext Markup Language) o Lenguaje de Marcas de Hipertexto:
es un lenguaje de etiquetas diseñado para describir la estructura de texto en un documento y complementar ese texto con formularios interactivos, imágenes

embebidas y otros objetos; de manera de presentarlo en forma de hipertexto, que es el formato estándar de las páginas web [GLSR16].

IDE (Integrated Development Environment) o Entorno de Desarrollo Integrado: aplicación de software que proporciona funciones completas a programadores para el desarrollo de software. Un IDE normalmente consta de un editor de código fuente, un compilador y/o intérprete, herramientas de automatización de generación y un depurador. Y en ocasiones, un sistema de control de versiones y diversas herramientas están integradas para simplificar la construcción de una interfaz gráfica de usuario [GLSR17].

Java: lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 1990. Toma mucha de su sintaxis de lenguajes como C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros [GLSR18].

JCC (Javascript Client Communication) o Comunicación de Cliente Javascript: hace referencia a las técnicas de programación que, utilizando objetos JSI (Interfaces Compartidas JavaScript, del ingles JavaScript Shared Interfaces) en el navegador (en el lado cliente y no en el servidor), facilitan la integración en la misma página web de aplicaciones y servicios a priori independientes [GLSR19].

JDK (Java Development Kit) o Kit de Desarrollo Java: compilador y conjunto de herramientas de desarrollo para la creación de programas independientes y applets en lenguaje Java [GLSR20].

JRE (Java Runtime Environment) o Entorno de Ejecución Java: proporciona únicamente un subconjunto del lenguaje de programación Java sólo para ejecución. El usuario final normalmente utiliza JRE en paquetes y añadidos [GLSR21].

JUnit: es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera [GLSR22].

kbps: (velocidad en la que viajan los datos por la red) kilobits por segundos, 1 kbps equivale a 1000 bits por segundo.

meme: unidad teórica de información cultural para su transmisión de un individuo a otro o de una mente a otra [GLSR23].

PERL (Practical Extraction and Report Language) o Lenguaje Práctico para la Extracción e Informe: lenguaje de propósito general originalmente desarrollado para la manipulación de texto y que ahora es utilizado para un amplio rango de tareas incluyendo administración de sistemas, desarrollo web, programación en red, desarrollo de GUI y más [GLSR24].

PHP (Hypertext Pre-processor) o Preprocesador de Hipertexto: lenguaje de programación usado normalmente para la creación de páginas web dinámicas [GLSR25].

plugin (plug-in, addon, add-on) o complemento: aplicación que interactúa con otra aplicación para aportarle una función o utilidad específica. Se utilizan como una forma de expandir programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes ni complicar el desarrollo del programa principal [GLSR26].

popups (ventanas emergentes): formato publicitario en Internet, la cual consiste en incrementar el tráfico en la web. Esto tiene lugar, cuando ciertos sitios web abren una nueva ventana del navegador web para mostrar la publicidad [GLSR27].

POST: método de peticiones HTTP, los valores se pasan por los formularios.

Python: es un lenguaje de programación dinámico orientado a objetos utilizado para muchos tipos de desarrollo de software. Ofrece compatibilidad para la integración con otros lenguajes y herramientas, además de una amplia biblioteca estándar [GLSR28].

refactorización: técnica de la ingeniería de software para reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo [GLSR29].

REST (Representational State Transfer) o Transferencia de Estado Representacional: es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web, describe cualquier interfaz web simple que utiliza XML y HTTP [GLSR30].

RSS (Really Simple Syndication) o Sindicación Realmente Simple: formato sencillo de datos que es utilizado para redifundir contenidos a suscriptores de un sitio web. Permite distribuir contenido sin necesidad de un navegador, utilizando un software diseñado para leer estos contenidos RSS [GLSR31].

scripting: también conocido como lenguaje de script o lenguaje interpretado a un lenguaje de programación que fue diseñado para ser ejecutado por medio de un intérprete [GLSR32].

servlets (Java servlet): programas escritos en Java que se ejecutan del lado del servidor, ya sea en el contexto de un contenedor web o de un servidor de aplicaciones [GLSR33].

sindicación: redifusión de contenidos informativos.

tabla hash: estructura de datos que asocia claves con valores. La operación principal que soporta de manera eficiente es la búsqueda: permite el acceso a los elementos almacenados a partir de una clave generada [GLSR34].

toolkit: kit de herramientas.

URL (Uniform Resource Locutor) o Localizador Uniforme de Recurso: es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización [GLSR35].

versión beta: versión de prueba.

W3C (World Wide Web Consortium): El World Wide Web Consortium, es un consorcio internacional que produce estándares para la World Wide Web (HTML, CSS, XML, DOM, etc) [GLSR36].

servicios Web (Web services): es una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, en redes de ordenadores como Internet [GLSR37].

widget es un componente software visible y personalizable. Visible porque está pensado para ser usado en los interfaces gráficos de los programas, y personalizable porque el programador puede cambiar muchas de sus propiedades [GLSR38].

wiki: es un sitio web colaborativo que puede ser editado por varios usuarios. Los usuarios de una wiki pueden así crear, modificar, borrar el contenido de una página web, de forma interactiva, fácil y rápida; dichas facilidades hacen de la wiki una herramienta efectiva para la escritura colaborativa [GLSR39].

XHTML (eXtensible Hypertext Markup Language) o Lenguaje Extensible de Marcas de Hipertexto: es el lenguaje de etiquetas pensado para sustituir a HTML como estándar para las páginas web. XHTML es la versión XML de HTML, por lo que básicamente tiene, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es lograr una web semántica, donde la información, y la forma de presentarla estén claramente separadas.
XHTML transmite la información que contiene un documento, dejando para las hojas de estilo y Javascript el aspecto y diseño en distintos medios [GLSR40].

XPath (XML Path Language): es un lenguaje que permite construir expresiones que recorren y procesan un documento XML, además de buscar y seleccionar teniendo en cuenta la estructura jerárquica del mismo. Su propósito es navegar a través de los nodos y atributo existentes en los documentos XML [GLSR41].

BIBLIOGRAFÍA.

[Garitano] La nueva interacción en Internet, Web 2.0
Autor: Bertol Belloso Garitano (aka newton)
Bajo licencia: Reconocimiento-NoComercial- CompartirIgual 2.5 España
Fuente: http://creativecommons.org/licenses/by-sa/2.5/es/
Formato del documento: PDF

[Crane-Pascarello-James06]
AJAX In Action
Autores: Dave Crane, Eric Pascarello y Darren James
Editorial: Manning Publications Co.
Fecha: 2006
Formato del documento: PDF

[Asleson-Schutta06]

Foundations of AJAX

Autores: Ryan Asleson y Nathaniel T. Schutta

Editorial: Apress.

Fecha: 2006

Formato del documento: PDF

[Abrams- Feizabadi98]

World Wide Web - Beyond the Basics

Autor: Marc Abrams y Shahrooz Feizabadi

Editorial: Prentice Hall.

Fecha: 1998.

Fuente: http://ei.cs.vt.edu/~wwwbtb/book/

Formato del documento: HTML

[W3C99] HTML 4.01 Specification - W3C Recommendation

Fecha: 24 de Diciembre del 1999

Fuente: http://www.w3.org/TR/html401/

Formato del documento: HTML

[Gehtland-Galbraith-Almaer05]

A Web 2.0 Primer Pragmatic AJAX

Autores: Justin Gehtland, Ben Galbraith y Dion Almaer

Editorial: The Pragmatic Bookshelf

Fecha: 26 de Octubre del 2005

Formato del documento: PDF

[Zakas-McPeak-Fawcett06]

Professional AJAX

Autores: Nicholas C. Zakas, Jeremy McPeak y Joe Fawcett

Editorial: Wiley Publishing, Inc.

Fecha: 2006
Formato del documento: PDF

[O'Reilly06] What Is Web 2.0. Design Patterns and Business Models for the Next Generation of Software.
Autor: Tim O'Reilly, Presidente y CEO de O'Reilly Media, Inc
Fecha: 23 de Febrero del 2006
Fuente:http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
Formato del documento: HTML

[Bray06] What Matters.
Autor: Tim Bray, Director de Tecnologias Webs de Sun Microsystems
Fecha: 12 de Mayo del 2006
Fuente:
http://www.tbray.org/ongoing/When/200x/2006/05/12/Web-0.3
Formato del documento: HTML

[Hinchcliffe06] A round of Web 2.0 reductionism.
Autor: Dion Hinchcliffe
Fecha: 15 de Mayo del 2006
Fuente: http://blogs.zdnet.com/Hinchcliffe/?p=41
Formato del documento: HTML

[O'Reilly-UB06] My Commencement Speech at SIMS.
(Discurso en la Universidad de Berkeley)
Autor: Tim O'Reilly, Presidente y CEO de O'Reilly Media, Inc
Fecha: 14 de Mayo del 2006
Fuente: http://radar.oreilly.com/archives/2006/05/my-commencement-speech-at-sims.html

Formato del documento: HTML

[Moral06] Qué debe tener una web 2.0
Autor: José Antonio del Moral
Fecha: 10 de Septiembre del 2006
Fuente:http://blogs.alianzo.com/redessociales/2006/09/10/que-debe-tener-una-web-2-0/
Formato del documento: HTML

[Giles05] Internet encyclopaedias go head to head
Autor: Jim Giles
Fecha: 14 de Diciembre del 2005
Fuente:http://www.nature.com/nature/journal/v438/n7070/full/438900a.html
Formato del documento: HTML

[Carr05] The amorality of Web 2.0
Autor: Nicholas Carr
Fecha: 3 de Octubre del 2005
Fuente:http://www.roughtype.com/archives/2005/10/the_amorality_o.php
Formato del documento: HTML

[Heilmann06] Beginning JavaScript with DOM Scripting and AJAX - From Novice to Professional
Autor: Christian Heilmann
Editorial: Apress.
Fecha: 2006
Formato del documento: PDF

[Goodman01] JavaScript Bible, Gold Edition

Autor: Danny Goodman
Editorial: Hungry Minds, Inc.
Fecha: 2001
Formato del documento: PDF

[Hunter-Watt-Rafter-Duckett-Ayers-Chase-Fawcett-Gaven-Patterson04]
Beginning XML, 3rd Edition
Autores: David Hunter, Andrew Watt, Jeff Rafter, Jon Duckett, Danny Ayers, Nicholas Chase, Joe Fawcett, Tom Gaven y Bill Patterson
Editorial: Wiley Publishing, Inc.
Fecha: 2004
Formato del documento: PDF

[Jacobs06] Beginning XML with DOM and AJAX - From Novice to Professional
Autor: Sas Jacobs
Editorial: Apress.
Fecha: 2006
Formato del documento: PDF

[Shin-Ajax07] AJAX Basics
18-week "Free" AJAX Programming (with Passion!) Online Course
Autor: Sang Shin, sang.shin@sun.com
Fecha: 12 de Febrero del 2007
Bajo licencia: Sun Microsystems
Fuente:http://www.javapassion.com/ajax/AJAXBasics_speakernoted.pdf
Formato del documento: PDF

[Shin-Dojo07] Introduction to Dojo Toolkit
18-week "Free" AJAX Programming (with Passion!) Online Course
Autor: Sang Shin, sang.shin@sun.com
Fecha: 12 de Febrero del 2007
Bajo licencia: Sun Microsystems
Fuente:http://www.javapassion.com/ajax/DojoToolkit_speakernoted.pdf
Formato del documento: PDF

[Shin-DWR07] Introduction to DWR (Direct Web Remoting)
18-week "Free" AJAX Programming (with Passion!) Online Course
Autor: Sang Shin, sang.shin@sun.com
Fecha: 12 de Febrero del 2007
Bajo licencia: Sun Microsystems
Fuente:
http://www.javapassion.com/ajax/DWR_speakernoted.pdf
Formato del documento: PDF

[Shin-JSON07] Introduction to JSON (JavaScript Object Notation)
18-week "Free" AJAX Programming (with Passion!) Online Course
Autor: Sang Shin, sang.shin@sun.com
Fecha: 12 de Febrero del 2007
Bajo licencia: Sun Microsystems
Fuente:
http://www.javapassion.com/ajax/JSON_speakernoted.pdf
Formato del documento: PDF

[Shin-GWT07] Google Web Toolkit (GWT) Basics

18-week "Free" AJAX Programming (with Passion!) Online Course
Autor: Sang Shin, sang.shin@sun.com
Fecha: 12 de Febrero del 2007
Bajo licencia: Sun Microsystems
Fuente: http://www.javapassion.com/ajax/GWT_speakernoted.pdf
Formato del documento: PDF

[Garrett05] AJAX: A New Approach to Web
Autor: Jesse James Garrett
Fecha: 18 de Febrero del 2005
Fuente:http://www.adaptivepath.com/publications/essays/archives/000385.php
Formato del documento: HTML

[Stewart07] User Experience, Rich Internet Applications and the future of software
Autor: Ryan Stewart
Categorias: Rich Internet Applications, Experience, Rich Media, Design, Designer Workflow
Fecha: 9 de Febrero del 2007
Fuente: http://blogs.zdnet.com/Stewart/?p=256
Formato del documento: HTML

[Cluts98] DHTML? Applets? Controls? Which To Use?
Autor: Nancy Cluts - Ingeniera de tecnología de desarrolladores - Microsoft Corporation
Fecha: 13 de Febrero del 1998
Fuente:http://www.microsoft.com/technet/community/events/dhmtlvsc.mspx

Formato del documento: HTML

[Fain07] Rich Internet Applications - State of the Union
What's your technology choice for implementing RIA?
Autor: Yakov Fain
Fecha: 13 Febrero del 2007
Fuente: http://java.sys-con.com/read/336933.htm
Formato del documento: HTML

[O'Rourke04] A Look at Rich Internet Applications
Autor: Cameron O'Rourke, cameron.orourke@oracle.com
Fecha: Julio / Agosto 2004
Fuente: http://www.oracle.com/technology/oramag/oracle/04-jul/o44dev_trends.html
Formato del documento: HTML

[Grosso05] Laszlo: An Open Source Framework for Rich Internet Applications
Autor: William Grosso
Fecha: 22 de Marzo del 2005
Fuente:http://today.java.net/pub/a/today/2005/03/22/laszlo.html
Formato del documento: HTML

[Gadge06] Technology options for Rich Internet Applications
Autor: Vaibhav V. Gadge, vaigadge@in.ibm.com, Software Engineer, IBM
Fecha: 25 de Julio 2006
Fuente: http://www-128.ibm.com/developerworks/library/wa-richiapp/
Formato del documento: HTML

[Srinivas01] Java Web Start to the rescue
Autor: Raghavan N. Srinivas, rags@sun.com
Fecha: 07 de Junio del 2001
Fuente: http://www.javaworld.com/javaworld/jw-07-2001/jw-0706-webstart.html?page=2
Formato del documento: HTML

[Domenig05] Rich Internet Applications and AJAX - Selecting the best product
Autor: Marc Domenig
Fecha: 2005
Fuente: http://www.javalobby.org/articles/ajax-ria-overview/
Formato del documento: HTML

[Eckel07] Computing Thoughts Hybridizing Java
Autor: Bruce Eckel
Fecha: 30 de Enero del 2007
Fuente:http://www.artima.com/weblogs/viewpost.jsp?thread=193593
Formato del documento: HTML

[Prototype] Fuente: http://www.prototypejs.org/
Autor: Sam Stephenson
Formato del documento: HTML

[Scriptaculous] Fuente: http://script.aculo.us/
Autor: Thomas Fuchs
Formato del documento: HTML

[Dojo] Fuente: http://dojotoolkit.org/
Autor: Alex Russell

Formato del documento: HTML

[DWR] Fuente: http://getahead.org/dwr/
Formato del documento: HTML

[Constable06] Simple jQuery Examples
Autor: Mark Constable
Año: 2006
Fuente: http://markc.renta.net/jquery/
Formato del documento: HTML

[Yahoo!] Fuente: http://developer.yahoo.com/yui/
Formato del documento: HTML

[Rico] Fuente: http://openrico.org/
Autores: Bill Scott y Darren James
Formato del documento: HTML

[Sajax] Fuente: www.modernmethod.com/sajax
Formato del documento: HTML

[JSON] Fuente: http://json.org/
Formato del documento: HTML

[JSON-RPC] Fuente: http://json-rpc.org/
Formato del documento: HTML

[GWT] Fuente: http://code.google.com/webtoolkit
Formato del documento: HTML

[Mahemoff06] AJAX Design Patterns

Autor: Michael Mahemoff
Editorial: O'Reilly.
Fecha: Junio 2006
Formato del documento: CHM y HTML
Fuente Web (Wiki): http://ajaxpatterns.org/

[OWASP-XSS] Cross-site-scripting
Proyecto Abierto de Seguridad de Aplicaciones Web
(The Open Web Application Security Project - OWASP)
Fuente: http://www.owasp.org/index.php/Cross-site-scripting
Formato del documento: HTML

[OWASP-XSRF] Cross-Site Request Forgery
Proyecto Abierto de Seguridad de Aplicaciones Web
(The Open Web Application Security Project - OWASP)
Fuente: http://www.owasp.org/index.php/XSRF
Formato del documento: HTML

[OWASP-XPATH] XPATH Injection
Proyecto Abierto de Seguridad de Aplicaciones Web
(The Open Web Application Security Project - OWASP)
Fuente: http://www.owasp.org/index.php/XPATH_Injection
Formato del documento: HTML

[Hayre-Kelath06] AJAX Security Basics
Autor: Jaswinder Hayre y Jayasankar Kelath
Fecha: 19 de Junio del 2006
Fuente: http://www.securityfocus.com/infocus/1868
Formato del documento: HTML

[Hoffman06] AJAX Security Dangers

Autor: Billy Hoffman
Fecha: 2006
Fuente: SPI Dynamics, Inc.
www.spidynamics.com/assets/documents/AJAXdangers.pdf
Formato del documento: PDF

[Shah10/06] Top 10 Web 2.0 Attack Vectors
Fecha: 9 de Octubre del 2006
Autor: Shreeraj Shah
Fuente: http://www.net-security.org/article.php?id=949
Formato del documento: HTML

[Shah11/06] Top 10 AJAX Security Holes and Driving Factors
Fecha: 10 de Noviembre del 2006
Autor: Shreeraj Shah
Fuente: http://www.net-security.org/article.php?id=956
Formato del documento: HTML

[Wiki01] http://es.wikipedia.org/wiki/Navegador_web

[Wiki02] http://en.wikipedia.org/wiki/Rich_Internet_application

[Wiki03] http://en.wikipedia.org/wiki/OpenLaszlo

[Wiki04] http://es.wikipedia.org/wiki/Web_2.0

[Wiki05] http://es.wikipedia.org/wiki/Biblioteca_%28programaci%C3%B3n%29

[Wiki06] http://es.wikipedia.org/wiki/Framework

[Wiki07] http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o

[FIG1]:

http://www.oracle.com/technology/oramag/oracle/04-jul/o44dev_trends.html

[FIG2], [FIG3], [FIG4] y [FIG5]:

http://www.adaptivepath.com/publications/essays/archives/000385.php

[FIG6]:

http://sociedaddelainformacion.telefonica.es/jsp/articulos/detalle.jsp?elem=2146

[FIG7]:

http://code.google.com/webtoolkit/overview.html

[TAB1]:

http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html

[GLSR01]: http://es.wikipedia.org/wiki/ActionScript

[GLSR02]: http://es.wikipedia.org/wiki/Application_Programming_Interface

[GLSR03]: http://es.wikipedia.org/wiki/Applet

[GLSR04]: http://es.wikipedia.org/wiki/Active_Server_Pages

[GLSR05]: http://es.wikipedia.org/wiki/Atom

[GLSR06]: http://es.wikipedia.org/wiki/Banner

[GLSR07]: http://es.wikipedia.org/wiki/Blog

[GLSR08]: http://es.wikipedia.org/wiki/Buffer

[GLSR09]: http://es.wikipedia.org/wiki/Bytecode

[GLSR10]: http://es.wikipedia.org/wiki/Cach%C3%A9_de_disco

[GLSR11]: http://en.wikipedia.org/wiki/Cascading_Style_Sheets

[GLSR12]: http://es.wikipedia.org/wiki/Sistema_de_gesti%C3%B3n_de_contenido

[GLSR13]: http://es.wikipedia.org/wiki/Cookie

[GLSR14]: http://es.wikipedia.org/wiki/Document_Object_Model

[GLSR15]: http://en.wikipedia.org/wiki/Gecko_(layout_engine)

[GLSR16]: http://en.wikipedia.org/wiki/HTML

[GLSR17]: http://en.wikipedia.org/wiki/Integrated_development_environment

[GLSR18]: http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java

[GLSR19]: http://es.wikipedia.org/wiki/JCC

[GLSR20]: http://es.wikipedia.org/wiki/JDK

[GLSR21]: http://es.wikipedia.org/wiki/JRE

[GLSR22]: http://es.wikipedia.org/wiki/JUnit

[GLSR23]: http://es.wikipedia.org/wiki/Meme

[GLSR24]: http://perldoc.perl.org/perlintro.html

[GLSR25]: http://en.wikipedia.org/wiki/PHP

[GLSR26]: http://en.wikipedia.org/wiki/Plugin

[GLSR27]: http://en.wikipedia.org/wiki/Pop-up_ad

[GLSR28]: http://www.python.org/

[GLSR29]: http://es.wikipedia.org/wiki/Refactorizaci%C3%B3n

[GLSR30]: http://en.wikipedia.org/wiki/Representational_State_Transfer

[GLSR31]: http://es.wikipedia.org/wiki/RSS

[GLSR32]: http://es.wikipedia.org/wiki/Lenguaje_interpretado

[GLSR33]: http://es.wikipedia.org/wiki/Java_Servlet

[GLSR34]: http://es.wikipedia.org/wiki/Tabla_hash

[GLSR35]: http://es.wikipedia.org/wiki/URL

[GLSR36]: http://es.wikipedia.org/wiki/World_Wide_Web_Consortium

[GLSR37]: http://es.wikipedia.org/wiki/Servicios_Web

[GLSR38]: http://libros.es.gnome.org/librognome/librognome/librognome/c4389.html

[GLSR39]: http://es.wikipedia.org/wiki/Wiki

[GLSR40]: http://es.wikipedia.org/wiki/XHTML

[GLSR41]: http://es.wikipedia.org/wiki/Xpath