



Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB

Luis Adolfo Oxlej Mangandi

Asesorado por el Ing. Calixto Raúl Monzón Pérez

Guatemala, septiembre de 2008

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA



FACULTAD DE INGENIERÍA

**FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO
ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB**

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA

POR:

LUIS ADOLFO OXLAJ MANGANDI

ASESORADO POR EL ING. CALIXTO RAÚL MONZÓN PÉREZ

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, SEPTIEMBRE DE 2008.

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA



NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Inga. Glenda Patricia García Soria
VOCAL II	Inga. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Angel Dávila Calderón
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Marlon Antonio Pérez Turk
EXAMINADORA	Inga. Virginia Victoria Tala Ayerdi
EXAMINADOR	Ing. Victor Hugo de León Barrios
SECRETARIA	Inga. Marcia Ivónne Véliz Vargas

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

**FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO
ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB,**

tema que me fuera asignado por la Dirección de la Escuela de Sistemas y Sistemas, con fecha de enero de 2007.



LUIS ADOLFO OXLAJ MANGANDI



**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS**

Guatemala, 26 de mayo de 2008

Señores
Comisión de Revisión de Trabajos de Graduación
Carrera de Ciencias y Sistemas
Facultad de Ingeniería
Universidad de San Carlos de Guatemala
Guatemala, Ciudad

Respetables Señores:

El motivo de la presente es informarles que como asesor del estudiante **LUIS ADOLFO OXLAJ MANGANDI** he procedido a revisar el trabajo de graduación titulado "**FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB**", y a mi criterio el mismo se encuentra concluido.

He tenido reuniones periódicas con el estudiante y luego de haber revisado cuidadosamente el trabajo, considero que cumple con los requisitos de calidad y profesionalismo que deben caracterizar a un futuro profesional de Informática.

Sin otro particular me suscribo de ustedes,

Atentamente,


Ing. Calixto Raul Monzón Pérez

Calixto Raul Monzón Pérez
INGENIERO EN CIENCIAS Y SISTEMAS
MASTER EN ADMINISTRACION DE EMPRESAS
COLEGIADO 3656



Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería en Ciencias y Sistemas

Guatemala, 14 de Julio de 2008

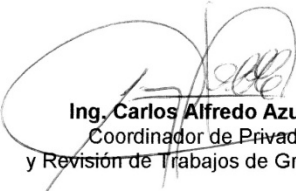
Ingeniero
Marlon Antonio Pérez Turk
Director de la Escuela de Ingeniería
En Ciencias y Sistemas

Respetable Ingeniero Pérez:

Por este medio hago de su conocimiento que he revisado el trabajo de graduación del estudiante **LUIS ADOLFO OXLAJ MANGANDI**, titulado: **“FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB”**, y a mi criterio el mismo cumple con los objetivos propuestos para su desarrollo, según el protocolo.

Al agradecer su atención a la presente, aprovecho la oportunidad para suscribirme,

Atentamente,


Ing. Carlos Alfredo Azurdia
Coordinador de Privados
y Revisión de Trabajos de Graduación



E
S
C
U
L
A
D
E

C
I
E
N
C
I
A
S

Y

S
I
S
T
E
M
A
S

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA


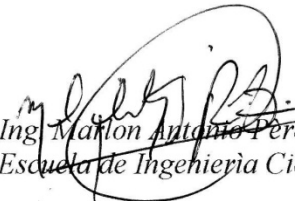


FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
TEL: 24767644

*El Director de la Escuela de Ingeniería en Ciencias y Sistemas de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer el dictamen del asesor con el visto bueno del revisor y del Licenciado en Letras, de trabajo de graduación titulado **“FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB”**, presentado por el estudiante **LUIS ADOLFO OXLAJ MANGANDI**, aprueba el presente trabajo y solicita la autorización del mismo.*

“ID Y ENSEÑAD A TODOS”

Ing. Marlon Antonio Pérez Yurk
Director, Escuela de Ingeniería Ciencias y Sistemas



Guatemala, 01 de septiembre 2008

Universidad de San Carlos
de Guatemala



Facultad de Ingeniería
Decanato

Ref. DTG.293.08

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería en Ciencias y Sistemas, al trabajo de graduación titulado: **FUERZAS Y DEBILIDADES DE AJAX COMO UN NUEVO ENFOQUE PARA EL DESARROLLO DE APLICACIONES WEB**, presentado por el estudiante universitario **Luis Adolfo Oxlej Mangandi**, autoriza la impresión del mismo.

IMPRÍMASE.

A handwritten signature in black ink, consisting of a large loop at the top and several vertical strokes below.

Ing. Murphy Olympo Paiz Recinos
DECANO



Guatemala, Septiembre de 2008

/cc
c.c. archivo.

ACTO QUE DEDICO A:

Dios

Por darme la vida, fuerzas, perseverancia, paciencia y oportunidades para alcanzar esta meta.

Mis padres

Luis Bartolo Oxlaj Matul y Aura Gracia Mangandi Ortiz, por brindarme su apoyo incondicional en todo momento.

Mis tíos

Patricia, Antonio, Josué, Isaac, por sus consejos y apoyo.

Mi asesor

Ing. Calixto Monzón Pérez, por compartir su experiencia, por asesorar este trabajo de graduación.

Mis amigos

Por su amistad y todos esos momentos compartidos a lo largo de la carrera.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES.....	VII
GLOSARIO	XI
RESUMEN.....	XXIII
OBJETIVOS.....	XXV
INTRODUCCIÓN	XXVII
1. DEFINICIÓN DE AJAX	1
1.1. ¿Qué es AJAX?	1
1.2. Diferencias de AJAX con las aplicaciones <i>Web</i> tradicionales.....	2
1.3. Modelo tradicional de una aplicación <i>Web</i>	3
1.4. Modelo AJAX de una aplicación <i>Web</i>	5
1.5. Exploradores que soportan AJAX.....	7
2. TECNOLOGÍAS QUE HACEN POSIBLE AJAX	9
2.1. Tecnologías detrás de AJAX	9
2.2. Javascript.....	9
2.2.1. ¿Qué es Javascript?	9
2.2.2. Variables	10
2.2.3. Tipos de Datos	11
2.2.4. Operadores	11
2.2.5. Sentencias	12
2.2.6. Funciones	12
2.2.7. Objetos	13
2.3. <i>Cascading Style Sheets</i> (Hojas de estilo en cascada).....	13
2.3.1. ¿Qué son las hojas de estilo?	13
2.3.2. Definición de las hojas de estilo	14

2.3.3.	Propiedades de las hojas de estilo	15
2.4.	<i>Document Object Model</i> (Modelo de objetos de documento)	17
2.4.1.	¿Qué es DOM?.....	17
2.4.2.	La interfaz <i>document</i>	18
2.4.3.	Interfaz <i>node</i>	19
2.4.4.	Interfaz <i>element</i>	20
2.5.	¿Qué es XML?	21
2.5.1.	Declaración de XML.....	21
2.5.2.	Etiquetas.....	22
2.5.3.	Atributos.....	22
2.5.4.	Elementos y sub-elementos.....	23
2.5.5.	DTD (<i>Document Type Definition</i>)	23
2.5.5.1.	Elementos	24
2.5.5.2.	Sub-elementos	24
2.5.5.3.	Número de sub-elementos	25
2.5.5.4.	Elementos vacíos	25
2.6.	JSON (JavaScript Object Notation)	25
2.6.1.	¿Qué es JSON?.....	25
2.6.2.	Objeto.....	26
2.6.3.	<i>Array</i>	26
2.6.4.	Valores	26
2.6.5.	Cadena.....	27
2.6.6.	JSON alternativa de XML.....	28
2.7.	El objeto <i>XMLHttpRequest</i>	28
2.7.1.	Antecedentes históricos.....	28
2.7.2.	El objeto <i>XMLHttpRequest</i>	29
2.7.3.	Propiedades	30
2.7.4.	Métodos.....	30
2.7.5.	Creación de la instancia	31

3.	AJAX UN NUEVO ENFOQUE PARA APLICACIONES WEB	33
3.1.	Principios de AJAX	33
3.2.	Arquitectura AJAX	35
3.3.	El patrón MVC.....	36
3.4.	Modelo de eventos	39
3.5.	Tecnologías alternativas.....	41
3.5.1.	Flash.....	42
3.5.2.	Java Applets	43
3.5.3.	XAML.....	44
3.5.4.	XUL	46
3.5.5.	Comparación de tecnologías alternativas y AJAX	47
3.5.6.	Eligiendo la mejor opción.....	48
3.6.	Ejemplos de Aplicaciones AJAX.....	51
3.6.1.	Google maps	51
3.6.2.	Gmail.....	52
4.	FRAMEWORKS PARA LOS LENGUAJES WEB MÁS COMUNES.....	55
4.1.	Clasificación de los <i>frameworks</i>	55
4.2.	Java	56
4.3.	PHP	57
4.4.	.Net.....	59
4.5.	Javascript.....	61
5.	CONSIDERACIONES DE SEGURIDAD EN AJAX	63
5.1.	Seguridad en AJAX	63
5.2.	Secuencias de comandos entre sitios	65
5.3.	Inyecciones SQL.....	67
5.4.	Protección de datos confidenciales	69
5.5.	Javascript y seguridad del explorador.....	70

6.	VENTAJAS Y DESVENTAJAS DE AJAX	73
6.1.	Ventajas.....	73
6.1.1.	Rendimiento	73
6.1.2.	Mejores Interfaces de usuario	74
6.1.3.	Mejor Interacción.....	75
6.1.4.	Portabilidad	76
6.2.	Desventajas	76
6.2.1.	Usabilidad	76
6.2.2.	Accesibilidad.....	77
6.2.3.	Problemas más comunes	78
7.	MEJORES PRÁCTICAS PARA EL USO DE AJAX.....	81
7.1.	Mejores prácticas utilizando Javascript	81
7.2.	Patrones de diseño.....	84
7.2.1.	El patrón fachada	85
7.2.2.	El patrón adaptador	86
7.2.3.	El patrón observador	87
7.2.4.	Patrones de diseño AJAX	88
7.3.	Optimización de código	89
7.3.1.	Optimizar los ciclos	89
7.3.2.	Agregar nodos DOM.....	90
7.3.3.	Minimizar la notación de punto	91
7.4.	Cuándo utilizar AJAX.....	91
7.5.	Manejo de errores.....	92
8.	EJEMPLO UTILIZANDO AJAX.....	95
8.1.	Descripción	95
8.2.	Solución.....	96
8.2.1.	Requisitos.....	96
8.2.2.	Descripción	96

8.2.3.	Opciones del estudiante	97
8.2.3.1.	Cursos asignados	98
8.2.3.2.	Asignación cursos	100
8.2.4.	Opciones de administrador o catedrático	100
8.2.4.1.	Cursos	101
8.2.4.2.	Agregar sección	103
8.2.4.3.	Crear <i>links</i>	103
8.2.4.4.	Nuevo ciclo	104
8.3.	Beneficios de AJAX	105
8.3.1.	Gestiones de la opción catedrático	105
8.3.2.	Gestiones de la opción estudiante	107
	CONCLUSIONES.....	109
	RECOMENDACIONES.....	113
	REFERENCIAS	117
	BIBLIOGRAFÍA.....	119
	APENDICES.....	117
	ANEXOS	134

ÍNDICE DE ILUSTRACIONES

FIGURAS

1	Modelo tradicional de una aplicación <i>Web</i>	4
2	Modelo AJAX de una aplicación <i>Web</i>	6
3	Estructura de JSON	27
4	Arquitectura <i>Web</i> y arquitectura AJAX	36
5	Funcionamiento general del patrón MVC	37
6	Ejemplo de una aplicación Flash	43
7	Ejemplo de un Java Applet	44
8	Ejemplo de una aplicación utilizando XAML	45
9	Ejemplo de una aplicación <i>Web</i> basada en XUL	47
10	Árbol de decisión entre tecnologías	50
11	Google Maps	52
12	Gmail	53
13	Estructura del patrón fachada	85
14	Estructura del patrón adaptador	86
15	Estructura del patrón observador	87
16	Esquema de un patrón de diseño AJAX	88
17	Pantalla de autenticación	97
18	Opciones para los estudiantes	98
19	Búsqueda por nombre curso	99
20	Búsqueda en árboles jerárquicos	99
21	Detalle de información de un curso	99
22	Asignación de cursos	100

23	Agregar cursos	101
24	Agregar curso a carrera	102
25	Agregar curso a ciclo actual	102
26	Agregar sección a curso	103
27	Agregar links a sección	104
28	Nuevo ciclo	104
29	Gestiones de la opción catedrático	107
30	Gestiones de la opción estudiante	108
31	Menú agregar referencia en Visual Studio 2005	127
32	Agregar la referencia en Visual Studio 2005	128
33	Diagrama del modelo de datos del ejemplo AJAX	133

TABLAS

I	Tipos de datos en Javascript	11
II	Operadores en Javascript	11
III	Sentencias de control y bucle en javascript	12
IV	Funciones de uso común en Javascript	12
V	Objetos de uso común en Javascript	13
VI	Tipos de unidades permitidas en CSS	15
VII	Propiedades de fuente comunes en CSS	15
VIII	Propiedades de texto comunes en CSS	16
IX	Propiedades de márgenes y <i>padding</i> comunes en CSS	16
X	Propiedades de colores y fondos comunes en CSS	17
XI	Métodos comunes de la interfaz <i>document</i> (DOM)	19
XII	Atributos comunes de la interfaz <i>node</i> (DOM)	19
XIII	Métodos comunes de la interfaz <i>node</i> (DOM)	20

XIV	Métodos comunes de la interfaz <i>element</i> (DOM)	20
XV	Símbolos para definir la cardinalidad de sub elementos en XML	25
XVI	Propiedades del objeto <i>XMLHttpRequest</i>	30
XVII	Métodos del objeto <i>XMLHttpRequest</i>	30
XVIII	Propiedades para el manejo de eventos en el explorador	40
XIX	Comparación de tecnologías alternativas y AJAX	48
XX	<i>Frameworks</i> de AJAX para Java	56
XXI	<i>Frameworks</i> de AJAX para PHP	58
XXII	<i>Frameworks</i> de AJAX para .Net	60
XXIII	<i>Frameworks</i> de AJAX para Javascript	61
XXIV	Tabla comparativa de la opción catedrático	106
XXV	Tabla comparativa de la opción estudiante	108

GLOSARIO

<i>.Net Framework</i>	Es una infraestructura que reúne un conjunto de lenguajes y servicios que simplifican el desarrollo de aplicaciones. Los principales componentes de este entorno son: lenguaje de compilación, biblioteca de clase de .Net, CLR (<i>Common Language Runtime</i>) y el sistema de tipos universal denominado <i>Common Type System</i> (CTS).
3GL	Siglas de <i>Third Generation Language</i> , lenguaje de tercera generación. Son lenguajes de programación de alta nivel para construir aplicaciones. El programador especifica qué tiene hacer la computadora y cómo debe hacerlo. Ejemplos de estos lenguajes son: C, Pascal y Fortran.
Accesibilidad	Característica de diseño de una aplicación <i>Web</i> que permite el acceso a la información sin limitación alguna.
<i>ActiveX</i>	Conjunto de tecnologías desarrolladas por Microsoft, que permite el intercambio de información entre diferentes aplicaciones.
Ancho de banda	Cantidad de datos que pueden transmitirse en un tiempo determinado por un canal de transmisión. El ancho de banda se expresa en <i>bits</i> por segundo (bps).

API	Siglas de <i>Application Program Interface</i> , interface de programación de aplicaciones. Es un conjunto de librerías, protocolos y herramientas que ayudan al programador a construir aplicaciones.
Aplicación	Programa de computadora para realizar una tarea específica. Entre las aplicaciones informáticas se encuentran: programas contables y programas para la administración de una planilla.
Asíncrono	Es la capacidad de manejar procesos independientemente de otros procesos. Esto quiere decir que un proceso A puede ejecutar otro proceso B. Entonces el proceso A no necesariamente espera a que se complete el proceso B para continuar.
Bookmark	Funcionalidad de los exploradores que permiten guardar la dirección URL de una página para visitar de nuevo esa misma página en otro momento.
Cache	Se refiere al área de memoria donde se almacena información que se accede continuamente. Es una técnica que se utiliza para mejorar el rendimiento de las aplicaciones que acceden continuamente a un dispositivo de menor rendimiento que la memoria del computador.
Chat	Comunicación en tiempo real entre dos usuarios. La comunicación puede hacerse desde una computadora u otro dispositivo que permita escribir y leer los mensajes.

Clase	Es una plantilla para objetos. Las clases encapsulan todas las características comunes de un conjunto particular de objetos.
<i>ComboBox</i>	Es un control de selección que despliega la actual selección y que provee una lista de posibles selecciones.
<i>Cookie</i>	Es un archivo de texto donde el explorador guarda información que posteriormente se usará para algún fin específico.
CRM	Siglas de <i>Customer Relationship Management</i> , administración de la relación con el cliente. Cubre todos los aspectos de interacción entre una compañía y sus clientes. Sean estos relacionados por la venta de un producto o servicio.
CSS	Siglas de <i>Cascading Style Sheets</i> , hojas de estilo en cascada. En un estándar para definir la presentación de documentos HTML y XHTML. Permite separar la estructura del documento y su presentación
<i>Datagrid</i>	Es un control para mostrar datos en forma tabular. Permite opciones como ordenamiento de columnas, manejo de eventos, formato, etcétera.
DOM	Siglas de <i>Document Object Model</i> , modelo de objetos de documento. Es un API para manipular la estructura de los documentos HTML y XML. El DOM define el documento como una estructura de árbol que puede ser modificada dinámicamente.

Etiqueta	En HTML son códigos de formato < > que indican al navegador como mostrar la información
Evento	Es una acción generada por las acciones de un usuario cuando interactúa con una aplicación, o por el cambio de un estado dentro de la aplicación.
Explorador	Software que se encarga de interpretar los distintos documentos que se publican en Internet. Permite al usuario visualizar, acceder y modificar la información que se publica.
Frame	Es una opción del HTML, que permite dividir una página <i>Web</i> en distintos espacios. Cada <i>frame</i> puede tener contenido distinto a los demás.
Framework	Es una estructura de soporte definida que sirve como base para el desarrollo de otra aplicación. Puede incluir el soporte de programas, librerías y documentación del mismo.
Hacker	Es una persona con altos conocimientos de informática, que obtiene acceso no autorizado a sistemas de información.
Hardware	Se refiere a los componentes físicos de la computadora como el procesador, disco duro, la memoria y los demás componentes que forman la computadora.
Hipervínculos	Es una referencia a un lugar específico de un documento o bien a otro documento distinto. Se representan por una palabra o imagen, al darles clic sobre ellos se abre el documento referenciado.

HTML	Siglas de <i>HyperText Markup Language</i> , lenguaje de marcas de hipertexto. Es el formato estándar de los documentos que se publican en el Internet. Este tipo de documento está formado por etiquetas que definen como se va a mostrar la información que interpreta el explorador.
HTTP	Siglas de <i>HyperText Transfer Protocol</i> , protocolo de transferencia de hipertexto. Es utilizado por Internet. Este protocolo define como los mensajes son estructurados y transmitidos y que acciones deben tomar los servidores y los exploradores en respuesta a determinados comandos.
IDE	Siglas de <i>Integrated Development Environment</i> , entorno de desarrollo integrado. Comúnmente incluye las siguientes herramientas: un diseñador de interfaces de usuario, editor de código, un compilador o un intérprete y un depurador. Ejemplos: Visual Studio, JBuilder y Frontpage.
Implementar	Implantar, poner en marcha o ejecutar una instalación informática.
Incompatibilidad	Se refiere a determinada funcionalidad y características que no son comunes en diferentes entornos.
Interactividad	Es la capacidad de intercambio y dialogo entre usuarios y ordenadores

Interfaz de usuario	Permiten al usuario comunicarse con el programa o aplicación. Existen varios tipos de estas interfaces como las basadas en comandos, menús y gráficos.
Internet	Es un conjunto de redes a nivel mundial que se conectan a través de distintas vías de comunicación entre las que destacan la fibra óptica, las ondas de radio y líneas telefónicas. Utiliza el protocolo TCP/IP para comunicarse con los distintos dispositivos.
ISO 8859-1	Es una norma que define la codificación del alfabeto latino y caracteres especiales.
Java	Lenguaje orientado a objetos desarrollado por Sun Microsystems. Este lenguaje utiliza un intérprete que es específico de la plataforma donde se ejecuta. Los programas escritos en Java, una vez compilados pueden ejecutarse en cualquier plataforma que tenga instalado el intérprete o máquina virtual.
Javascript	Es un lenguaje <i>script</i> interpretado, con sintaxis similar a Java pero simplificado. Los programas Javascript se incluyen en la misma página o se escriben en archivos separados, para incluirlos más adelante como una referencia en la página HTML.
Modelo	Representación simplificada de una parte de la realidad y sus elementos relacionados.

Multimedia	Se refiere a la forma de presentar información que incluye elementos de texto, sonido, imágenes, animación y video.
Multiplataforma	Es la capacidad de funcionar de forma similar en diferentes sistemas operativos o plataformas.
Objeto	Un objeto es una instancia de una clase. Pueden representar cosas reales o conceptuales. Los objetos cuentan con propiedades y acciones.
<i>Open source</i>	Código abierto se refiere a que es permitido ver, modificar y distribuir el código fuente de una aplicación o parte del mismo.
Orientado a objetos	Es una paradigma que considera a los sistemas como colecciones de objetos capaces de interactuar, que trabajan conjuntamente para realizar una tarea.
<i>Padding</i>	Es la cantidad de espacio entre el borde y el contenido de un elemento. Propiedad de presentación en CSS y HTML.
PHP	Es un lenguaje de programación del lado servidor, rápido, gratuito e independiente de plataforma. Este lenguaje provee bastante documentación y una amplia variedad de librerías.
Plataforma	Se refiere al <i>Software</i> o <i>Hardware</i> para un sistema. Define un estándar dentro del cual el sistema puede ser construido.

<i>Plug-in</i>	Es un programa que interactúa con otro para agregarle otra función o utilidad específica. Comúnmente los <i>plug-ins</i> proveen soporte de audio, video, animación y gráficas.
Renderizar	Es la acción de asignar y calcular todas las propiedades de un objeto antes de mostrarlo en pantalla. Muy utilizado para la generación de gráficas y animaciones.
RIA	Siglas de <i>Rich Internet Applications</i> , aplicaciones ricas de Internet. Son aplicaciones <i>Web</i> accesibles desde un explorador con la funcionalidad de una aplicación de escritorio. Emulando sus interfaces de usuario y comportamiento.
<i>Script</i>	Programa interpretado que se utiliza para llevar a cabo tareas secuenciales.
SDK	Siglas de <i>Software Development Kit</i> , Herramienta de desarrollo de aplicaciones. Es un paquete de programación que ayuda al programador a desarrollar aplicaciones para una plataforma específica.
Servicio <i>Web</i>	Es un modelo basado en estándares que permite conectar aplicaciones a través de Internet. La comunicación es en XML. No están atados a ningún sistema operativo o lenguaje de programación.
Servidor	Es una computadora conectada a una red que generalmente tiene más recursos que los otros nodos de la red y que pone a disposición sus recursos y servicios a los miembros de la red.

SGML	Siglas de <i>Standard Generalized Markup Language</i> , lenguaje estándar de marcado de documentos. Es un estándar de descripción de página independiente de dispositivo, lo que permite adaptar como se mostrará el documento de acuerdo al tamaño de la pantalla donde se muestra
Síncrono	Se refiere cuando un proceso depende de otro proceso. Es decir el proceso se desarrolla con correspondencia temporal con otro proceso.
Software	Programas de computadora que procesa el <i>Hardware</i> , para realizar una tarea. Los tipos primarios son los sistemas operativos y las aplicaciones. Los sistemas operativos se encargan de la administración de los recursos de la computadora. Las aplicaciones son programas que se ejecutan sobre un sistema operativo que tienen un fin específico como los procesadores de texto.
SQL	Siglas de <i>Structured Query Language</i> , lenguaje de consulta estructurado. Es un lenguaje de origen matemático que es utilizado en las bases de datos para realizar operaciones sobre los datos que almacenan. Entre las operaciones que realiza el SQL se encuentran la actualización, eliminación y consulta de información, y la creación e eliminación de tablas.
TreeView	Es un control que permite crear interfaces de usuario que representan datos jerárquicos.

UI	Siglas de <i>User Interface</i> , interfaz de usuario. Permiten al usuario comunicarse con el programa o aplicación. Existen varios tipos de estas interfaces como las basadas en comandos, menús y gráficos.
Unicode	Es una norma de codificación de caracteres. Su objetivo es asignar un identificador y nombre único a cada posible carácter de cada posible lenguaje
URL	Siglas de <i>Uniform Resource Locator</i> , localizador universal de recursos. Define la dirección global de los documentos y recursos del Internet
Usabilidad	Es una medida del nivel acerca de lo fácil, rápido y agradable que es utilizar un determinado producto o servicio.
Variable	Es una estructura de datos que puede adquirir o retorna un valor dado. Mantienen los datos hasta que el programa termine o se le asigne otro valor
Vbscript	Siglas de <i>Visual Basic Scripting</i> . Es un lenguaje <i>script</i> desarrollado por Microsoft soportador por Internet Explorer. Está basado en Visual Basic pero más simple.
W3C	Son las siglas de <i>World Wide Web Consortium</i> , se encarga de realizar estándares para la Web. Antes de que pase a ser una recomendación pasa por los siguientes estados: Borrador de trabajo, última convocatoria, recomendación candidata, propuesta de recomendación y recomendación del consorcio.

Web cam

Es una cámara digital pequeña que se conecta a una computadora, la cual permite capturar imágenes.

XML

Siglas de *eXtensible Markup Language*, lenguaje de marcas extensible. Es un lenguaje de marcas que permite definir otros lenguajes, además de ser un estándar para el intercambio de datos entre distintas plataformas.

RESUMEN

AJAX es un modelo utilizado para sitios *Web* que permite simular una aplicación de escritorio. La forma de hacerlo es a través de un objeto del explorador que permite la comunicación asíncrona con el servidor. Este objeto, el *XMLHttpRequest* junto con las hojas de estilo, XML y DOM forman en conjunto AJAX.

Este enfoque tiene una diferencia al compararlo con una aplicación *Web* tradicional. El motor AJAX que actúa como intermediario entre el explorador y el servidor *Web*, tiene como objetivo principal gestionar los eventos generados por el usuario y la aplicación ya sean estos de comunicación o cambios en la interfaz gráfica.

Además de AJAX existen otras opciones como Flash y los Java Applets. Estas opciones son similares, pero cada una con características especiales. Flash suele ser la mejor opción cuando se necesitan realizar algún tipo de animación, o cuando se necesita acceder a un dispositivo como cámaras y micrófonos. Java al ser un lenguaje robusto orientado a objetos, éste permite un desarrollo más formal, además de contar con muchas herramientas de desarrollo y depuración disponibles en el mercado. Lo atractivo de AJAX es que no hay que instalar ningún *plug-in*, solo con tener activo Javascript y disponer de un explorador lo suficientemente moderno la aplicación puede funcionar. La mejor opción depende del tipo de aplicación *Web* y de los requerimientos del mismo.

Una aplicación AJAX puede ser desarrollado desde cero, es decir usando directamente el objeto *XMLHttpRequest* y manipulando la interfaz gráfica de la página *Web*, o usando algún tipo de *framework* del lado del cliente o del servidor, que provee abstracciones de alto nivel para desarrollar la aplicación. La ventaja de esto es que el tiempo de desarrollo tiende a disminuir, pero también hay una gran posibilidad de aumentar la cantidad de código resultante, aumentando el tiempo de carga inicial.

La seguridad es importante al momento de diseñar la aplicación AJAX, ya que es tan vulnerable como las aplicaciones *Web* tradicionales a las secuencias de comando entre sitios y las inyecciones SQL. La validación de todo tipo entrada de datos en el lado del cliente y del servidor pueden prevenir los efectos de estos ataques.

AJAX al igual que cualquier tecnología tiene desventajas y ventajas. Entre las ventajas más importantes se encuentran la rapidez con la que puede reaccionar la interfaz gráfica, la interactividad y el consumo de ancho de banda. Entre las desventajas se encuentran la falta de herramientas adecuadas para el desarrollo, la accesibilidad y algunos aspectos de usabilidad como el problema con botón de atrás y el uso de *bookmarks*.

La optimización de código y el uso de patrones de diseño en el desarrollo de la aplicación AJAX, ayudan a disminuir los errores y a generar una aplicación más robusta y rápida.

En el capítulo final se presenta un ejemplo práctico de AJAX. La aplicación es una versión reducida de la Universidad Virtual de la Escuela de Ciencias y Sistemas. El ejemplo utiliza *frameworks open source* para generar la aplicación AJAX sin necesidad de programar intensivamente Javascript.

Se utiliza este ejemplo para demostrar los beneficios de AJAX, ya que se realiza una comparación de rendimiento con AJAX y sin AJAX. Al utilizar AJAX disminuye la cantidad de datos transmitidos, el tiempo necesario para completar la tarea y aumenta la interactividad.

OBJETIVOS

General

Proveer un documento donde se presenta en forma objetiva los aspectos a considerar cuando se desarrolla una aplicación AJAX, con el fin de decidir si es viable aplicar este enfoque a un proyecto dado.

Específicos

1. Conocer qué es AJAX.
2. Describir las tecnologías detrás de AJAX.
3. Conocer la arquitectura de AJAX.
4. Conocer los distintos *frameworks* para los lenguajes *Web* más comunes, que actualmente soportan AJAX.
5. Conocer las ventajas y desventajas del uso de AJAX.
6. Conocer las mejores prácticas para utilizar correctamente AJAX, para desarrollo de aplicaciones *Web*.
7. Realizar un ejemplo de una pequeña aplicación *Web* utilizando AJAX en .NET.

INTRODUCCIÓN

AJAX son las siglas *para Asynchronous Javascript y XML*, y no es una nueva tecnología, sino la combinación de varias tecnologías: Javascript como lenguaje de programación, XML para el intercambio de datos, DOM para el manejo de los objetos de la página y un modelo asíncrono de comunicación para el intercambio de datos.

La idea principal de AJAX es cargar y renderizar una página, esto se logra con base a *scripts* que se comunican de manera asíncrona con el servidor para traer los datos y luego cambiar partes de esta sin tener que cargar de nuevo la página.

Entre las empresas más grandes que utilizan AJAX se encuentra Google, Yahoo, Amazon y Microsoft. Las aplicaciones van desde completar algunos campos automáticamente hasta una suite de oficina con procesador de texto y hojas electrónicas.

Entre los beneficios de AJAX encontramos: rapidez de respuesta, interfaces de usuario gráficamente ricas, menor consumo de ancho de banda e interactividad. Pero no todo es perfecto con AJAX, porque presenta grandes problemas si no se utiliza correctamente.

La razón para realizar este trabajo es mostrar todos los aspectos positivos y negativos de esta tendencia. Conocer en qué situaciones es más recomendable y cuándo otra tecnología es mejor. Asimismo presenta los principios, tecnologías y aspectos que se deben tomar en cuenta si la opción AJAX es la elegida.

1. DEFINICIÓN DE AJAX

1.1. ¿Qué es AJAX?

Hace algún tiempo para realizar una aplicación *Web* que tuviera la funcionalidad de una aplicación de escritorio, había que realizarla en Flash, Java o algún otro producto comercial. Esta solución tenía el inconveniente de tener que instalar un *plug-in* o en el caso de los *applets* la máquina virtual de Java.

AJAX viene a ser otra opción y son las siglas *Asynchronous Javascript y XML*. Pero no es una nueva tecnología, sino la combinación de varias tecnologías y estándares que llevan muchos años usándose. Garret (2005) enumera las tecnologías que lo forman:

- CSS: Su función es manipular la presentación de las páginas *Web*, sin tener que cambiar el código HTML.
- DOM: Su función es la manipulación dinámica e interacción de los elementos HTML y XML.
- XML: Su función es proveer un estándar para el intercambio de datos.
- El objeto *XMLHttpRequest*: Su función es la proveer un mecanismo de comunicación síncrona y asíncrona con el servidor.
- Javascript: Su función es proveer un lenguaje de programación que se encarga de unir todas estas tecnologías y estándares.

Google es posiblemente la empresa que le dio el empuje necesario a AJAX para que se convirtiera en un enfoque que muchas empresas están siguiendo.

En la mayoría de artículos y libros no faltan Google Suggest, Google Maps y Gmail como ejemplos típicos de AJAX. Todos estos ejemplos se estaban desarrollando antes de que este enfoque tuviera un nombre.

1.2. Diferencias de AJAX con las aplicaciones *Web* tradicionales

Si se compara una aplicación *Web* tradicional con una basada en AJAX, se descubren muchas diferencias. Estas diferencias se refieren al modo de interactuar con el usuario y el servidor *Web*, comportamiento de las interfaces gráficas y el consumo de ancho de banda.

Por ejemplo una aplicación *Web* que funcione con el esquema tradicional, retornará una página HTML cada vez que se pulse sobre un link o botón. Mientras una basada en AJAX, el cambio será a nivel de segmento de la página. Esta diferencia trae beneficios como:

- Respuesta más rápida.
- Menor tráfico de datos.
- Mayor interactividad y una mejor experiencia al usuario.

Otra diferencia es la capacidad de simular una aplicación de escritorio. La forma de realizarlo es por medio del objeto *XMLHttpRequest*. Este objeto puede comunicarse con el servidor de forma asíncrona. Por ejemplo, si seleccionamos un departamento de Guatemala en un *ComboBox*, otro *ComboBox* se llenaría con los municipios de ese departamento. Esto se realizaría sin necesidad de actualizar la página por otra, solamente se solicitarían los datos relacionados con el departamento seleccionado.

Pero no todo en AJAX es perfecto. El simple hecho de estar basado en Javascript trae problemas de accesibilidad, porque no todos tienen activado Javascript en el explorador. También se encuentran los exploradores que solo muestran texto, estos no tiene soporte para Javascript y no podrán visualizar la aplicación AJAX.

Además de estos problemas también se encuentran los de usabilidad. Un usuario acostumbrado al funcionamiento de una aplicación *Web* tradicional, se puede confundir si la aplicación AJAX no muestra las señales apropiadas cuando cambia de un estado a otro. En el ejemplo anterior, si el escenario presenta una comunicación lenta con el servidor y una cantidad considerable de datos, el usuario no tendría idea de lo que esta pasando, si la pagina no muestra ninguna señal de comunicación o proceso con el servidor *Web*. Por eso es importante diseñar la aplicación AJAX teniendo en cuenta estos detalles.

1.3. Modelo tradicional de una aplicación *Web*

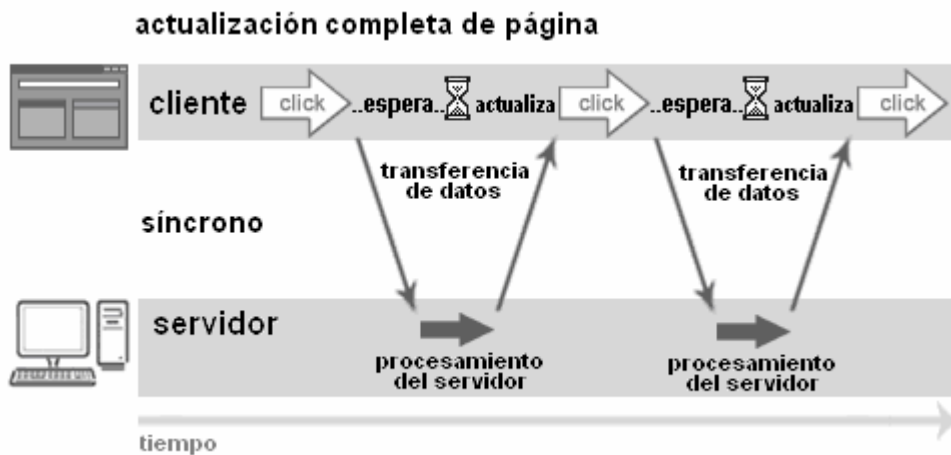
Para tener idea de las diferencias entre el modelo tradicional y el modelo AJAX, se resume de manera general el funcionamiento del modelo tradicional.

1. El usuario ingresa una dirección de un sitio en la barra de direcciones del explorador.
2. El explorador envía una serie de peticiones HTTP al servidor *Web*.
3. El servidor *Web* verifica la petición y de ser necesario realiza algún proceso con la información enviada.
4. El servidor envía el resultado de la petición como documentos HTML, CSS, Javascript e imágenes.

5. Si el usuario quiere navegar por el sitio *Web* debe hacer clic a los hipervínculos o botones que se encuentran en la página. De esta manera se repite el paso dos, enviando de nuevo todas las peticiones necesarias para obtener el contenido que se le mostrara al usuario.

En la figura de abajo se muestra claramente el proceso mencionado anteriormente. Cada vez que se hace una petición hay que esperar, para que se refresque la página.

Figura 1. Modelo tradicional de una aplicación *Web*



Fuente: Coach K. Wei. **AJAX: Asynchronous Java + XML?**

<http://www.developer.com/design/article.php/3526681>. (05/01/2008)

Pero es común que solo una parte de la página HTML cambie de contenido. En un escenario donde el ancho de banda sea reducido, por ejemplo conexión vía teléfono, el rendimiento del sitio *Web* será muy pobre, y el usuario tendrá que esperar mucho tiempo para ver el resultado de sus acciones.

Una alternativa para solucionar este problema es el uso de marcos. Un marco o *frame* es una etiqueta del lenguaje HTML que permiten dividir el espacio visual del explorador. En cada división puede haber una página HTML diferente. Por ejemplo, si dividimos el espacio en dos espacios verticales, el menú podría estar en el espacio superior y el contenido en el espacio inferior. Entonces el escenario anterior puede mejorar en rendimiento porque disminuiría la cantidad de información que el servidor envía al explorador del usuario.

1.4. Modelo AJAX de una aplicación Web

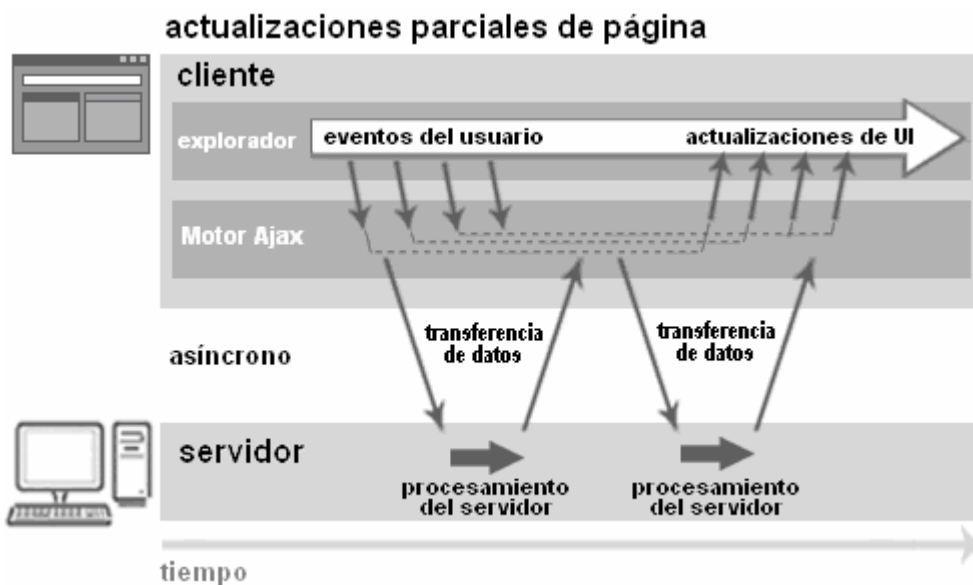
Este modelo es semejante al anterior, pero tiene una diferencia fundamental. Una vez cargada la aplicación AJAX, el motor AJAX se encarga de hacer peticiones al servidor en forma asíncrona.

Con esta característica el usuario puede seguir trabajando, y una vez que recibe los datos, el manejador de eventos se encarga de hacer los cambios a la página por medio de DOM. Para tener una mejor idea del proceso, se enumeran los pasos más importantes:

1. El usuario ingresa una dirección de un sitio en la barra de direcciones del explorador.
2. El explorador envía una serie de peticiones HTTP al servidor Web.
3. El servidor *Web* envía en respuesta de la petición documentos HTML, CSS, Javascript e imágenes.
4. Una vez que todos los archivos están cargados, el motor AJAX está listo para comunicarse con el servidor.
5. Ahora el usuario puede interactuar con la aplicación *Web*, y cada vez que se necesite de los datos del servidor, ocurre un evento.

6. Cuando ocurra el evento, el motor AJAX envía la petición al servidor. El servidor realiza todos los procesos necesarios y retorna la respuesta. En cuanto llegue la respuesta del servidor *Web*, el manejador de eventos lo detecta, y realiza por medio de DOM los cambios que fueran necesarios.

Figura 2. Modelo AJAX de una aplicación *Web*



Fuente: Coach K. Wei. **AJAX: Asynchronous Java + XML?**

<http://www.developer.com/design/article.php/3526681>. (05/01/2008)

El motor AJAX está escrito en Javascript, y utiliza al objeto *XMLHttpRequest* para intercambiar datos con el servidor en forma asíncrona. Otra función es la de hacer los cambios necesarios a la página. Cargar una página entera requiere de más ancho de banda que transmitir solo los datos necesarios. Estos cambios permiten una respuesta más rápida, menor uso de ancho de banda y una mejor experiencia al usuario.

1.5. Exploradores que soportan AJAX

Existen una gran cantidad de exploradores en el mercado, algunos disponibles solo para determinadas plataformas como Internet Explorer y otras multiplataformas como Mozilla Firefox. La mayoría de exploradores en sus nuevas versiones, soportan las aplicaciones AJAX, aunque existe la posibilidad de que alguna característica funcione solo en determinados exploradores.

La siguiente lista muestra los exploradores que soportan AJAX:

- Exploradores basados en *Gecko* como: Mozilla, Mozilla Firefox, SeaMonkey, Camino, Flock, Epiphany, Galeon y Netscape versión 7.1 y posteriores.
- Microsoft Internet Explorer desde la versión 5.0.
- Exploradores que implementan el API KHTML desde la versión 3.2, como Konqueror desde la versión 3.2 y Apple Safari desde la versión 1.2.
- Opera desde la versión 8.0.

Las versiones de los exploradores anteriores a las de la lista y los basados en texto no soportan el objeto *XMLHttpRequest* y en el caso de los basados en texto no soportan Javascript.

2. TECNOLOGÍAS QUE HACEN POSIBLE AJAX

2.1. Tecnologías detrás de AJAX

En el capítulo uno se mencionan las tecnologías que en conjunto forman AJAX. Todas estas tecnologías: Javascript, CSS, DOM, XML y el objeto *XMLHttpRequest* tienen una gran cantidad de aspectos los cuales se escapan del objetivo de este trabajo.

La finalidad de este capítulo es presentar en una forma resumida los aspectos más importantes de cada una de dichas tecnologías. También se presenta JSON como alternativa a XML, ya que el objeto *XMLHttpRequest* no necesariamente utiliza exclusivamente XML para la comunicación con el servidor, sino que tiene la flexibilidad de poder utilizar otras alternativas como JSON.

2.2. Javascript

2.2.1. ¿Qué es Javascript?

Javascript es un lenguaje de programación *script*, desarrollado por Netscape, cuyo propósito principal era la de modificar las etiquetas HTML y hacer tareas de validación. El código que se escribe de este lenguaje es interpretado por el explorador. Además Javascript no puede crear programas independientes, necesita estar incluido dentro de una página HTML para que pueda ser utilizado. Una de las ventajas de este lenguaje es que es fácil de aprender y utilizar, porque no es tan complejo como un lenguaje de propósito general como Java y C#.

Javascript empezó a incluirse en el navegador de Netscape desde la versión 2. La compañía rival de Netscape, Microsoft, lo incluye desde la versión 3 de su navegador Internet Explorer. Microsoft también desarrolló una alternativa a Javascript, VBScript un lenguaje soportado únicamente por el explorador de esta compañía.

La *European Computer Manufacturers Association* (ECMA), estandarizó Javascript con el estándar llamando *ECMAScript*. Aunque existe este estándar, no todos los exploradores lo implementan a cabalidad en un 100%. Por eso algunos aspectos no son compatibles entre exploradores, porque cada empresa lo implementa a su manera.

Javascript es utilizado comúnmente para pedir datos, realizar confirmaciones, mostrar mensajes, crear animaciones simples y comprobar campos.

Las características más importantes de Javascript son:

- **Es interpretado:** El explorador es el encargado de interpretar las instrucciones de este lenguaje.
- **Orientado a eventos con manejo de objetos:** Es posible definir objetos dentro de la página HTML, y sobre estos objetos podemos definir diferentes eventos para poder realizar paginas interactivas.
- **Débilmente tipeado:** Las variables no se declaran de un tipo específico, sino que pueden cambiar de tipo en cualquier momento.

2.2.2. Variables

Las variables en Javascript pueden ser de alcance global o local. Una variable global es accesible desde cualquier lugar de la página mientras que una variable local sólo lo es desde la función en la que fue declarada.

Normalmente, se crea una nueva variable global asignándole simplemente un valor.

```
GlobalVariable=5;
```


Sin embargo, en el caso de una función y quiere crear una variable local que sólo tenga alcance dentro de esa función, debe declarar la nueva variable haciendo uso de *var*.

```
var localVariable=1;
```

2.2.3. Tipos de datos

Javascript manipula los siguientes tipos de datos: numéricos, lógicos, cadenas y nulos.

Tabla I. Tipos de datos en Javascript

Tipo de dato	Ejemplo
Numérico	10, 10.1
Lógicos	true, false
Cadenas	"hola mundo", 'hola mundo'
Nulo	<i>Null</i>

2.2.4. Operadores

Javascript como cualquier lenguaje de programación, tiene una gran cantidad de operadores, que se utilizan para realizar muchas tareas. La siguiente tabla resume la mayoría de operadores soportados por Javascript.

Tabla II. Operadores en Javascript

Símbolo	Significado	Símbolo	Significado
Aritméticos		Comparación	
+	Suma	==	Igualdad
-	Resta	!=	Distinto
/	División	>	Mayor que
*	Multiplicación	<	Menor que
%	Modulo	>=	Mayor o igual que
++	Incremento	<=	Menor o igual que
--	Decremento		
Lógicos			
&&	<i>And</i>		
	<i>Or</i>		
!	<i>Not</i>		

2.2.5. Sentencias

La sintaxis de las sentencias de control y bucle es similar a la de C++ y Java. La siguiente tabla resume dichas sentencias.

Tabla III. Sentencias de control y bucle en Javascript

Sentencia	Sintaxis
<i>if else</i>	if (condición) { instrucciones } [else{ instrucciones }]
<i>while</i>	while (condición) { instrucciones }
<i>for</i>	for(valor inicio; condición; incremento) { instrucciones }
<i>switch</i>	Switch (variable) { case valor1: instrucciones break; case valor2: instrucciones break; default: instrucciones }
<i>break</i>	break;
<i>continue</i>	continue;

2.2.6. Funciones

Las funciones son definidas usando la palabra reservada *function*.

```
function nombre_funcion(parametros) {  
    comandos ...  
}
```

Las funciones pueden retornar valores usando la palabra reservada *return*.

```
function cube(number) {  
    return number * number * number;  
}
```

La siguiente tabla muestra algunas funciones comunes.

Tabla IV. Funciones de uso común en Javascript

Función	Descripción
<i>prompt</i> (mensaje, valor_defecto);	La función <i>prompt</i> nos permite introducir "valores", dichos valores se guardan en variables
<i>alert</i> (mensaje);	La función " <i>alert</i> " muestra mensajes y/o valores de variables
<i>eval</i> (textoCodigo);	Interpreta el texto, como código Javascript
<i>parseInt</i> (textoNúmero, base);	Convierte el texto a un número, el segundo parámetro es opcional y representa la base del número
<i>isNaN</i> (expresión);	Devuelve verdadero, si la expresión representa contenido no numérico.

2.2.7. Objetos

Para crear un objeto en Javascript se utiliza la instrucción *new*. La sintaxis es la siguiente:

variable= *new* Objeto (parámetros);

La siguiente tabla muestra algunos objetos definidos de uso común.

Tabla V. Objetos de uso común en Javascript

Objeto	Descripción
<i>Math</i>	Este objeto tiene funciones y propiedades para valores numéricos.
<i>Date</i>	Este objeto tiene funciones y propiedades para fechas y horas.
<i>array</i>	Se pueden crear <i>arrays</i> con un tamaño fijo o variable. También permite cualquier tipo de dato en cualquier posición de un mismo <i>array</i> .

2.3. *Cascading Style Sheets* (Hojas de estilo en cascada)

2.3.1. ¿Qué son las hojas de estilo?

Las hojas de estilo CSS son un estándar para la *Web* que se empezó a aplicar a partir del Internet Explorer 3.0 en adelante, así como en Netscape 4.0 en adelante. Se desarrolló con la idea de hacer más fácil la presentación a través de una forma lógica y estándar.

Hay un aspecto interesante de las hojas de estilo, es que permiten separar la información de la presentación del contenido, con lo que se facilita mucho el diseño y revisión de las páginas, ya que se puede variar la presentación de una página, o de todo el sitio Web, sin cambiar el código HTML.

Las hojas de estilo pueden cambiar los atributos de color, tipo de letra, fondo, tamaño de todos los elementos de una página HTML a través de reglas. Las reglas consisten en dos partes: el selector y la declaración de estilo. El selector define a que elementos va a ser aplicado el estilo o el nombre de la clase.

El nombre de la clase no define a que elemento va ser aplicado el estilo, para aplicarla a un elemento, se agrega el nombre de la clase en uno de los atributos del elemento. La declaración de estilo declara qué propiedades de estilo se van a aplicar o personalizar.

Las hojas de estilo definen estilo a diferentes niveles, por lo que cada nivel se impone al anterior. Se pueden crear hojas de estilo de dos maneras: la primera es incluirla en la misma página HTML y la segunda es crear un archivo distinto, el cual se incluye en la página HTML como una referencia.

2.3.2. Definición de las hojas de estilo

Las hojas de estilo se pueden definir de varias formas. La primera es a nivel de etiqueta. La sintaxis es:

```
<etiqueta STYLE="propiedad: valor, ...">... </etiqueta>.
```

La segunda forma es en forma global y la sintaxis es la siguiente:

```
<STYLE TYPE="text/css">  
<!-- elemento o clase {propiedad: valor;} -->  
</STYLE>.
```

La tercera forma es a través de un archivo. Para incluirla se agrega la siguiente referencia en el encabezado de la página:

<LINK REL="stylesheet" TYPE="text/css" HREF="nombre_archivo.css">.

La forma de definir los estilos en el archivo es igual a la forma global, pero sin la etiqueta *STYLE*.

2.3.3. Propiedades de las hojas de estilo

La siguiente tabla muestra los tipos de unidades permitidas cuando se asigna un valor numérico a una propiedad específica.

Tabla VI. Tipos de unidades permitidas en CSS

Nombre Unidad	Abreviatura	Ejemplo
Puntos	pt	12pt
Pulgadas	in	5in
Centímetros	cm	8cm
Píxel	px	20px
Porcentaje	%	10%

La siguiente tabla muestra las propiedades más comunes para aplicar a la fuente.

Tabla VII. Propiedades de fuente comunes en CSS

Propiedades de Fuente		
Atributo	Descripción	Valores
<i>Font-size</i>	Establece el tamaño de texto.	xx unidades
<i>Font-family</i>	Establece la fuente.	nombre de la fuente
<i>Font-weight</i>	Establece el espesor de la fuente.	<i>extra-light</i> <i>light</i> <i>demi-light</i> <i>medium</i> <i>demi-bold</i> <i>bold</i> <i>extra-bold</i>
<i>Font-style</i>	Establece el estilo.	<i>normal</i> <i>italic</i> <i>oblique</i>

La siguiente tabla muestra las propiedades más comunes para aplicar al texto.

Tabla VIII. Propiedades de texto comunes en CSS

Propiedades de Texto		
Atributo	Descripción	Valores
<i>line-height</i>	Establece la distancia entre líneas.	xx unidades
<i>text-decoration</i>	Subraya o remarca el texto.	<i>none</i> <i>underline</i> <i>italic</i> <i>line-through</i>
<i>text-align</i>	Establece la justificación del texto.	<i>Left, center, right</i>
<i>text-indent</i>	Establece la sangría de la primera línea del texto.	xx unidades

La siguiente tabla muestra las propiedades más comunes para los márgenes y *padding*.

Tabla IX. Propiedades de márgenes y *padding* comunes en CSS

Propiedades de Márgenes y <i>Padding</i>		
Atributo	Descripción	Valores
<i>margin-left</i>	Establece el margen izquierdo de la página.	xx unidades
<i>margin-right</i>	Establece el margen derecho de la página.	xx unidades
<i>margin-top</i>	Establece el margen superior de la página.	xx unidades
<i>margin-bottom</i>	Establece el margen inferior de la página.	xx unidades
<i>padding-top</i>	Distancia entre el borde superior y el contenido	xx unidades
<i>padding-bottom</i>	Distancia entre el borde superior y el contenido	xx unidades
<i>padding-left</i>	Distancia entre el borde superior y el contenido	xx unidades
<i>padding-right</i>	Distancia entre el borde superior y el contenido	xx unidades

La siguiente tabla muestra las propiedades más comunes los colores y fondos.

Tabla X. Propiedades de colores y fondos comunes en CSS

Propiedades de Colores y Fondos		
Atributo	Descripción	Valores
<i>color</i>	Establece el color de primer plano del contenido de texto	Nombre_color, Valor hex, Rgb(R,G,B)
<i>background-color</i>	Establece el color del fondo del elemento.	Nombre_color, Valor hex, Rgb(R,G,B), <i>transparent</i>
<i>background-image</i>	Establece una imagen para rellenar el fondo del elemento	URL(dirección)
<i>background</i>	Establece todos los atributos del fondo en una sola propiedad	

2.4. Document Object Model (Modelo de objetos de documento)

2.4.1. ¿Qué es DOM?

DOM es un estándar de W3C, que provee una plataforma o API que permite la actualización y acceso dinámico de la estructura y estilo de los documentos. DOM permite a los programadores realizar una gran variedad de tareas como: construir documentos, navegar por la estructura, agregar, eliminar, modificar elementos y contenido de documentos HTML y XML. DOM define una estructura lógica parecida a un árbol y ha sido diseñado para ser usado con cualquier lenguaje por lo que se especificación es independiente de lenguaje.

Existen varios niveles de DOM, que se han desarrollado a lo largo de los últimos años:

- Nivel 1: Define los modelos de los documentos HTML y XML. Provee la navegación y manipulación de sus estructuras.

- Nivel 2: Incluye el modelo de objetos de las hojas de estilo así como la capacidad de manipularlo. También define un modelo de eventos y provee soporte para espacio de nombres de XML.
- Nivel 3: Provee soporte para modelos de contenido que permite la validación de documentos.

DOM provee una gran funcionalidad e interfaces para manipular y acceder documentos HTML y XML, pero hay algunas excepciones:

- DOM no es una especificación binaria: No se define ninguna manera de permitir la interoperabilidad binaria.
- DOM no es una manera de hacer persistir objetos de HTML y XML: Especifica como los documentos HTML y XML pueden representarse como objetos.
- DOM no es una colección de estructuras de datos: Es un modelo de objetos que define interfaces y relaciones lógicas.
- DOM no define la semántica detrás de HTML y XML: Solamente es un modelo de objetos que provee interfaces para acceder y manipular estos documentos.

2.4.2. La interfaz *document*

Esta interfaz representa a todo el documento HTML o XML. Representa la raíz de todos los nodos del árbol del documento. Esta interfaz también contiene los métodos necesarios para poder crear otros nodos del árbol. El atributo *documentElement* representa al nodo raíz del documento.

La siguiente tabla muestra los métodos más comunes en la interfaz *document*.

Tabla XI. Métodos comunes de la interfaz *document* (DOM)

Método	Descripción
<i>createElement</i> (nombre_elemento)	Crea un elemento del tipo que se especifica en el parámetro. Retorna un objeto <i>Element</i> .
<i>createTextNode</i> (data)	Crea un nodo de texto. Retorna un objeto <i>Text</i>
<i>createAttribute</i> (nombre_atributo)	Crea un atributo para un elemento. Retorna un objeto <i>Attr</i> .
<i>getElementsByTagName</i> (nombre_elemento)	Retorna un objeto <i>NodeList</i> que contiene una lista de todos los elementos, que pertenecen a nombre_elemento.

2.4.3. Interfaz *node*

Es el principal tipo de dato. Representa cada nodo en el árbol del documento. Todos los objetos que implementan la interfaz *node* tienen métodos para poder trabajar con los nodos hijos.

La siguiente tabla muestra los atributos más comunes de la interfaz *node*.

Tabla XII. Atributos comunes de la interfaz *node* (DOM)

Atributo	Descripción
<i>nodeName</i>	Nombre del nodo, dependiendo del tipo.
<i>nodeValue</i>	Valor del nodo, dependiendo del tipo.
<i>nodeType</i>	Código que representa al objeto del nodo.
<i>parentNode</i>	Padre del nodo.
<i>childNodes</i>	Contiene todos los hijos del nodo.
<i>firstChild</i>	Primer hijo del nodo. Si no tiene hijos retorna <i>null</i> .
<i>lastChild</i>	Ultimo hijo del nodo. Si no tiene hijos retorna <i>null</i> .

La siguiente tabla muestra los métodos más comunes de la interfaz *node*.

Tabla XIII. Métodos comunes de la interfaz *node*(DOM)

Método	Descripción
<i>replaceChild</i> (nuevo_hijo, antiguo_hijo)	Reemplaza el nodo antiguo_hijo por el nodo nuevo_hijo y retorna el antiguo nodo hijo.
<i>removeChild</i> (hijo)	Elimina el nodo hijo de la lista de nodos hijos y retorna el nodo eliminado.
<i>appendChild</i> (nuevo_hijo)	Agrega el nodo nuevo_hijo al final de la lista de nodos hijos.
<i>hasChildNodes</i> ()	Retorna <i>true</i> si tiene algún nodo hijo.
<i>cloneNode</i> (deep)	Retorna un duplicado del nodo. Si el parámetro <i>deep</i> es <i>true</i> copia todos los sub-arboles que pueda tener el nodo.

2.4.4. Interfaz *element*

Los elementos de los documentos tienen asociados ciertos atributos, por lo que esta interfaz provee una colección de métodos para poder obtener y modificar estos atributos.

La siguiente tabla muestra los métodos más comunes de la interfaz *element*.

Tabla XIV. Métodos comunes de la interfaz *element*(DOM)

Método	Descripción
<i>getAttribute</i> (nombre)	Retorna el valor del atributo como cadena
<i>setAttribute</i> (nombre, valor)	Agrega un nuevo atributo. Si el atributo ya existe el valor es cambiado por este nuevo valor
<i>removeAttribute</i> (nombre)	Elimina el atributo.
<i>getAttributeNode</i> (nombre)	Retorna un objeto <i>Attr</i> , que representa al atributo
<i>setAttributeNode</i> (nuevo_atributo)	Agrega un objeto <i>Attr</i> de un nuevo atributo. Si el atributo ya existe el valor es cambiado por este nuevo valor.
<i>getElementsByTagName</i> (nombre_elemento)	Retorna un objeto <i>NodeList</i> que contiene una lista de todos los elementos, que pertenecen a nombre_elemento.

2.5. ¿Qué es XML?

XML es un estándar avalado por W3C. XML son las siglas de *Extensible Markup Language*. Es un lenguaje para definir otros lenguajes, por lo que se le denomina un metalenguaje. Con XML es posible definir estructuras de datos, definir etiquetas y todo esto independiente de plataforma.

Actualmente se está usando como un reemplazo de EDI (*Electronic Data Interchange*). EDI es un modelo para intercambiar datos entre empresas. EDI es bastante complejo, caro y necesita una infraestructura de comunicación dedicada.

Los principales objetivos de diseño de XML son:

- Sencillo de usar sobre Internet.
- Soporta una gran variedad de aplicaciones.
- Sea compatible con SGML.
- Sea fácil de escribir programas que procesen documentos XML.
- Los documentos XML son lo suficientemente legibles y claros como para ser comprendidos por las personas.
- Los documentos XML son fáciles de crear.
- El diseño de XML es formal y conciso.

2.5.1. Declaración de XML

La primera línea de un documento XML es la declaración del tipo de documento. Esto se realiza a través de una etiqueta especial. La sintaxis es:

```
<? xml version="1.0" ?>.
```

2.5.2. Etiquetas

En un documento XML, para definir el inicio de un elemento se usa una etiqueta de inicio. Cada elemento tendrá también una etiqueta de cierre. La sintaxis es:

```
<nombre_etiqueta> contenido </nombre_etiqueta>.
```

Para utilizar las etiquetas XML se deben seguir ciertas reglas:

- Las etiquetas son sensitivas al uso de mayúsculas y minúsculas.
- Cada etiqueta de inicio tiene una etiqueta de cierre.
- Todas las etiquetas deben estar anidadas adecuadamente.
- Entre la etiqueta de inicio y la de cierre se espera el contenido.

Existe otro tipo de etiquetas llamadas etiquetas vacías. Este tipo de etiqueta es utilizado cuando el elemento no tiene ningún contenido. La sintaxis es:

```
< nombre_Etiqueta />.
```

2.5.3. Atributos

Los atributos se pueden definir solo en la etiqueta de inicio o etiquetas vacías. La sintaxis es la siguiente:

```
<nombre_etiqueta atributo="valor"> contenido </nombre_etiqueta>.
```

Los atributos se usan comúnmente para:

- Definir una colección de atributos que pertenecen al elemento.
- Establecer límites de tipo para los atributos.
- Proveer valores por defecto para los atributos.

2.5.4. Elementos y sub-elementos

En XML los elementos pueden tener otros elementos llamados sub-elementos o hijos. Las etiquetas que definen estos elementos van entre las etiquetas de inicio y de cierre del elemento raíz. Ejemplo:

```
<empleado>
  <nombre>luis</nombre>
  <domicilio>
    <departamento> Guatemala </departamento>
    <municipio> Guatemala </municipio>
    <direccion> 1 av 0-0 zona 7 </direccion>
  </domicilio>
</empleado>
```

2.5.5. DTD (*Document Type Definition*)

Un documento XML además de estar estructurado correctamente, también debe ser válido. Un documento es válido si está bien estructurado y aplica las reglas definidas en el DTD. El DTD contiene las reglas y el significado de cada etiqueta para cada documento XML en particular.

El DTD se puede incluir en el mismo documento XML o en otro archivo. Para incluirlo en el propio documento XML se realiza de la siguiente manera.

```
<?xml version="1.0"?>
<!DOCTYPE nombre_elemento_raiz [
Definición de los elementos
]>
```

La segunda forma de incluirlo es a través de otro archivo. La forma de incluirlo es la siguiente:

```
<?xml version="1.0"?>  
<!DOCTYPE nombre_elemento_raiz SYSTEM "archivo.dtd">
```

2.5.5.1. Elementos

Para describir un elemento se realiza de la siguiente manera:

```
<! ELEMENT nombre_elemento descripción_elemento>.
```

Ejemplo:

```
<! ELEMENT municipio (#PCDATA)>
```

PCDATA significa que la etiqueta será mostrada y procesada. *CDATA* significa que la etiqueta no será procesada o mostrada.

2.5.5.2. Sub-elementos

Para definir que un elemento tiene hijos o sub elementos, se realiza de la siguiente manera:

```
<! ELEMENT nombre_elemento ( elemento1,elemento2, .. ) >
```

Ejemplo:

```
<!ELEMENT domicilio (departamento,municipio,direccion)>.
```

2.5.5.3. Número de sub-elementos

Para definir el número de sub elementos que puede tener un elemento, se utilizan unos símbolos especiales.

Tabla XV. Símbolos para definir la cardinalidad de sub elementos en XML

Símbolo	Descripción
+	Una o muchas veces
*	Cero o muchas veces
?	Cero o una vez

Ejemplo:

```
<!ELEMENT datos_personales (celular*,fecha_nacimiento+)>.
```

2.5.5.4. Elementos vacíos

Para definir un elemento vacío se realiza de la siguiente manera:

```
<! ELEMENT nombre_elemento EMPTY >.
```

Ejemplo:

```
<! ELEMENT foco EMPTY>
```

2.6. JSON (JavaScript Object Notation)

2.6.1. ¿Qué es JSON?

JSON son las siglas de *JavaScript Object Notation*. Es un formato ligero para el intercambio de datos. Una señal del incremento de la popularidad de JSON, es que Yahoo ha empezado a ofrecer algunos de sus *Web Services* en JSON.

JSON es independiente del lenguaje de programación y es soportado por la mayoría de lenguajes *script* y lenguajes de programación de uso general.

JSON está construido sobre dos estructuras:

- Una colección de pares nombre – valor.
- Una lista ordenada de valores.

Estas dos estructuras son estructuras de datos que pueden implementarse en cualquier lenguaje. Por ejemplo: registros, objetos, diccionarios, vectores, listas y muchas más. Es por eso que es independiente del lenguaje de programación.

2.6.2. Objeto

Un objeto es una colección desordenada de pares nombre : valor. El objeto se define entre llaves. Los pares nombre : valor son separados por comas. El nombre va encerrado entre comillas.

Ejemplo:

```
{“nombre” : “Juan” , “apellido” : “López”}
```

2.6.3. Array

Un *array* es una colección de valores ordenados. El *array* se define entre corchetes. Los valores son separados por comas.

Ejemplo:

```
[“Toyota”,”Nissan”,”KIA”]
```

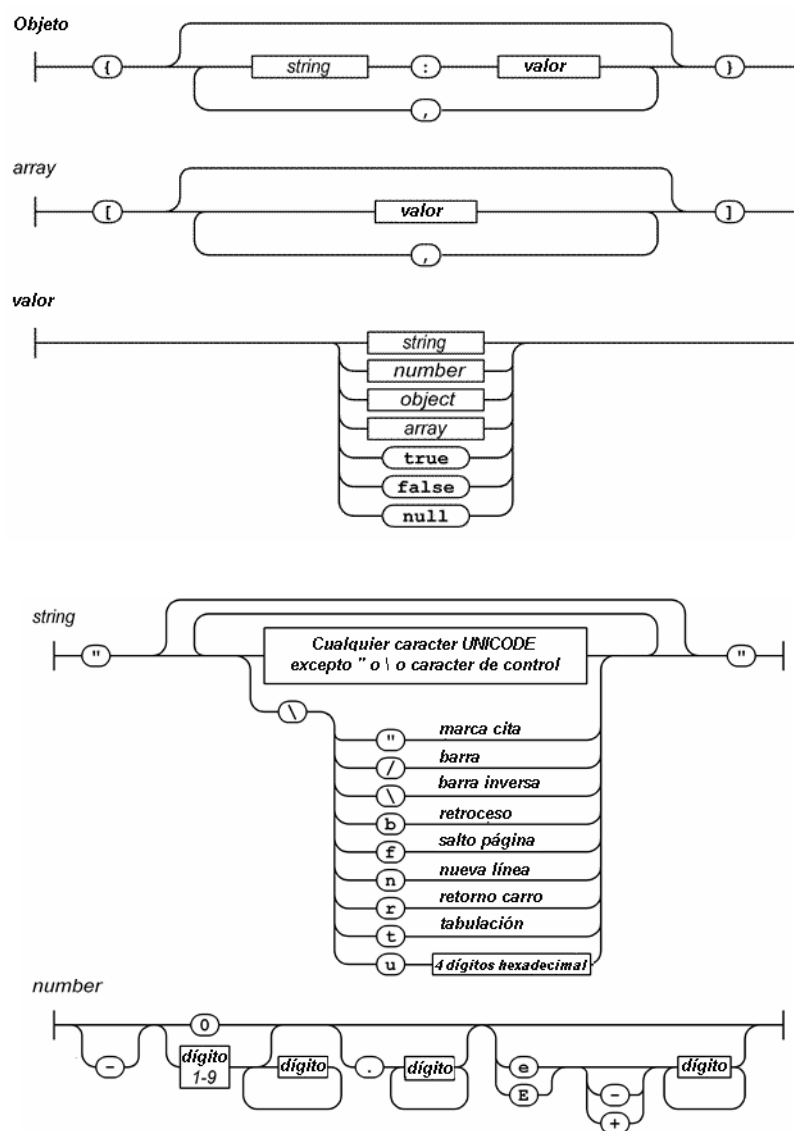
2.6.4. Valores

Los valores que pueden tomar los objetos y los *arrays* son: cadenas, números, *true*, *false*, objetos y *arrays*. Todas estas estructuras pueden anidarse.

2.6.5. Cadena

Una cadena es una colección de cero o más caracteres *unicode* encerrados entre comillas. Una cadena también puede tener caracteres de escape.

Figura 3. Estructura de JSON



Fuente: **Introducing JSON.**

<http://www.json.org/> (07/01/2008)

2.6.6. JSON alternativa de XML

JSON es un método más ligero para intercambiar datos. Si comparamos JSON y XML, JSON es más rápido porque está basado en estructuras comunes que tienen la mayoría de los lenguajes de programación y los datos enviados son más pequeños que un documento en formato XML. Es por estas características que puede ser un buen sustituto de XML, incluso puede mejorar el rendimiento de la aplicación si utilizamos JSON.

La decisión final dependerá del tipo de aplicación que se está desarrollando y de los requerimientos del mismo. Si se necesita compartir datos con otras empresas, el uso de XML sería el más adecuado porque permite de una manera formal especificar la estructura de los datos que enviamos.

2.7. El objeto *XMLHttpRequest*

2.7.1. Antecedentes históricos

El objeto *XMLHttpRequest* es el producto final después de varios años de intentos para tratar de lograr la comunicación asíncrona a través del explorador.

Uno de los primeros métodos fue a través del elemento *IFRAME* introducido desde la versión 3 del Internet Explorer. Y el elemento *LAYER* del Netscape 4 (abandonado en las siguientes versiones).

Tanto el elemento *IFRAME* y *LAYER* tienen el atributo *src*, que corresponde a una URL que puede ser cambiado en tiempo real para obtener información sin necesidad de cargar la página principal. La manipulación de esta información se puede hacer por medio de Javascript o VBScript.

En 1998 Microsoft introduce el MSRS (*Microsoft's Remote Scripting*). Con esta técnica existe un *applet* encargado de recibir las peticiones de información que hace el cliente a través de Javascript. Esta técnica es soportada desde la versión 4 de Internet Explorer y Netscape.

Después del MSRS, aparecen otras técnicas y *frameworks* que tratan de mejorar las características de las técnicas que existen en ese momento.

En el 2002 Microsoft reemplaza el MSRS con el *ActiveX XMLHttpRequest*.

2.7.2. El Objeto *XMLHttpRequest*

El Objeto *XMLHttpRequest* es un parte fundamental de una aplicación AJAX. Sin este objeto no es posible comunicarse de forma asíncrona desde el explorador del cliente hasta el servidor.

El objeto *XMLHttpRequest* aún no es un estándar de W3C. Se encuentra en revisión desde el 05 de abril de 2006. El 26 de octubre de 2007 fue publicado el sexto borrador en el sitio oficial de W3C, que se encuentran en revisión.

El 25 de febrero de 2008 se publicó el primer borrador del objeto *XMLHttpRequest* nivel 2, el cual agrega nuevas características, entre las que se incluyen: peticiones entre sitios, *status* de eventos y manejo de bloques de *bytes*.

Microsoft fue el primero en incluirlo en su explorador Microsoft Internet Explorer 5. La forma de incluirlo fue a través de un objeto *ActiveX*. Más tarde otros exploradores lo incluyen como un objeto nativo del explorador.

Con este objeto es posible pedir datos a un servidor sin tener que refrescar la página o hacerlo a través de otra página. A través de esta funcionalidad, las aplicaciones se pueden comportar como una aplicación de escritorio, dando una mejor experiencia al usuario.

Pero toda esta funcionalidad no está libre de problemas. Los exploradores que no tienen activado Javascript o versiones demasiadas antiguas, no podrán visualizar correctamente una aplicación AJAX.

2.7.3. Propiedades

La siguiente tabla muestra las propiedades del objeto *XMLHttpRequest*.

Tabla XVI. Propiedades del objeto *XMLHttpRequest*

Propiedad	Descripción
<i>onreadystatechange</i>	Manejador de eventos, se activa cuando ocurre un cambio de estado.
<i>readyState</i>	Indica el estado actual de la petición. Puede tomar los siguientes valores: 0=no_inicializado, 1=cargando, 2=cargado, 3=interactuando, 4=completado
<i>responseText</i>	Son los datos devueltos por el servidor en forma de cadena.
<i>responseXML</i>	Objeto de los datos devueltos por el servidor, compatible con DOM.
<i>status</i>	Código HTTP retornado por el servidor. Si la petición fue exitosa entonces el código será 200.
<i>statusText</i>	Es el código HTTP retornado por el servidor en forma de cadena.

2.7.4. Métodos

La siguiente tabla muestra los métodos del objeto *XMLHttpRequest*.

Tabla XVII. Métodos del objeto *XMLHttpRequest*

Método	Descripción
<i>abort()</i>	Cancela la petición actual
<i>getAllResponseHeaders()</i>	Retorna todas las cabeceras de la petición HTTP como una cadena.
<i>getResponseHeader("cabecera")</i>	Retorna el valor de la cabecera como una cadena.
<i>open("metodo","URL",asyncFlag,"nombre_usuario","clave")</i>	Abre una conexión a una URL. El método puede ser GET, POST o PUT. Solo los 2 primeros parámetros son obligatorios. La conexión es por defecto asíncrona, pero puede cambiarse por medio de <i>asyncFlag</i> , poniendo el valor a <i>false</i> .
<i>send(contenido)</i>	Envía la petición al servidor. Los datos pueden ser una cadena o un objeto.
<i>setRequestHeader("etiqueta","valor")</i>	Asigna un par etiqueta – valor a la cabecera de la petición.

2.7.5. Creación de la instancia

Antes de hacer uso de las propiedades y métodos se debe crear el objeto. Para crear una instancia en la mayoría de los exploradores se realiza de la siguiente manera:

```
httpRequest= new XMLHttpRequest();
```

Para crearlo en el Microsoft Internet Explorer se realiza por medio de un *ActiveX*. La forma de crearlo es la siguiente:

```
httpRequest= new ActiveXObject("Microsoft.XMLHTTP");
```


3. AJAX UN NUEVO ENFOQUE PARA APLICACIONES WEB

3.1. Principios de AJAX

Como se mencionó en el capítulo uno, una aplicación AJAX tiene varias diferencias en contraparte con una aplicación *Web* tradicional.

Si la aplicación *Web* y la aplicación AJAX se diseñan utilizando el modelo MVC entonces la capa de presentación o vista difiere en varios aspectos:

- Uso de un motor AJAX en el lado del usuario como intermediario entre la interfaz gráfica y el servidor.
- Las acciones del usuario son manejadas por el motor AJAX, en lugar de una petición al servidor.
- La comunicación de datos entre el motor AJAX y el servidor es en XML u otro formato de intercambio de datos.

El motor AJAX está compuesto por un conjunto de funciones Javascript que se encargan de realizar la comunicación con el servidor, así como los cambios en la interfaz gráfica. Este motor no necesita tener instalado ningún *plug-in*, porque está escrito en Javascript. Mientras Javascript este activado funcionará el motor AJAX.

Crane et al. (2006, 17) define 4 principios característicos de AJAX.

- **El explorador alberga una aplicación y no contenido.** En la manera tradicional de una aplicación *Web*, el explorador solo se encarga de visualizar los documentos HTML que el servidor envía al usuario. Cada vez que el usuario interactúa con la aplicación, el servidor procesa la petición y envía otro documento HTML. En el modelo AJAX, el explorador pasa de un simple visualizador a un elemento más inteligente. Ahora el explorador no solo se encarga de visualizar la aplicación *Web*, sino también realiza procesamiento de datos, modifica la estructura de las páginas y gestiona eventos del usuario y el explorador, todo esto a través de Javascript.
- **El servidor entrega datos y no contenido.** En la manera tradicional de una aplicación *Web*, el servidor entrega datos y contenido cada vez que se necesita actualizar la página. En muchos casos el cambio es mínimo, solo cambian algunos datos de la página. Entonces una aplicación AJAX toma esta idea. Cada vez que necesita datos del servidor, establece una petición asíncrona y solo recibe los datos que necesita. Aunque la carga inicial de una aplicación AJAX puede ser mayor, las peticiones subsecuentes son más ligeras y rápidas.
- **La interacción del usuario con la aplicación puede ser fluida y continua.** Una aplicación *Web* tradicional, tiene dos formas de comunicarse con el servidor: hipervínculos y formas. Cada vez que el usuario hace un clic sobre un hipervínculo o envía una forma haciendo clic sobre un botón, tiene que esperar que el servidor procese la petición y envíe la página resultante. Mientras tanto, el usuario tiene que esperar y no puede hacer nada más sobre la aplicación. Una aplicación AJAX evita estos problemas, cada vez que necesita algo del servidor, se comunica asíncronamente y cuando retorna la respuesta hace el cambio. En este escenario no se interrumpe el flujo de trabajo, porque el usuario puede seguir trabajando sobre la aplicación, sin tener que esperar. También se pueden manejar una gran variedad de acciones de usuario, permitiendo crear sofisticadas interfaces de usuario, que solo son encontradas en aplicaciones de escritorio.

- **Codificar esto requiere disciplina.** La aplicación AJAX que se carga en el explorador del usuario es compleja y debe correr hasta que se cierre, sin que se vuelva lenta, sin generar desbordes de memoria y sin que se cierre inesperadamente por mal manejo de errores. Por eso es importante usar estándares de programación y patrones que permitan un desarrollo más ordenado y claro.

3.2. Arquitectura AJAX

La gran diferencia entre la arquitectura de una aplicación AJAX y una aplicación *Web* tradicional es el motor AJAX.

En el capítulo uno se menciona las funciones del motor AJAX, las cuales son fundamentalmente coordinar los eventos generados por el usuario o por la misma aplicación, realizar la comunicación con el servidor, y realizar cambios en la página a nivel del DOM.

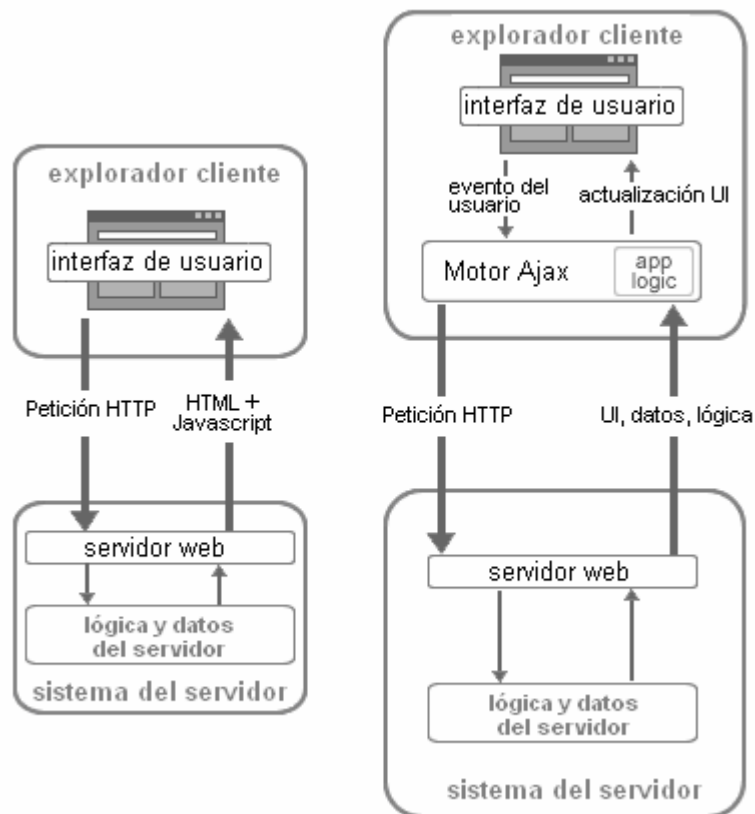
Sin el motor AJAX cada evento del usuario, podría significar una petición al servidor. El resultado sería una nueva página HTML. Ahora el explorador tiene una aplicación más inteligente, ya no solo visualiza el contenido de la página HTML sino realiza más procesamiento, aprovechando los recursos que tiene la computadora del usuario.

Desde una perspectiva general la arquitectura AJAX difiere de la arquitectura *Web* tradicional en los siguientes puntos:

- Motor AJAX del lado del cliente, como intermediario entre el servidor y las interfaces de usuario.

- Eventos generados por el usuario, generan llamadas al motor AJAX en lugar de peticiones directas al servidor.
- Comunicación asíncrona entre el motor AJAX y el servidor.
- Utiliza un formato de intercambio de datos como XML o JSON.

Figura 4. Arquitectura Web y arquitectura AJAX



Fuente: Coach K. Wei. **AJAX: Asynchronous Java + XML?**

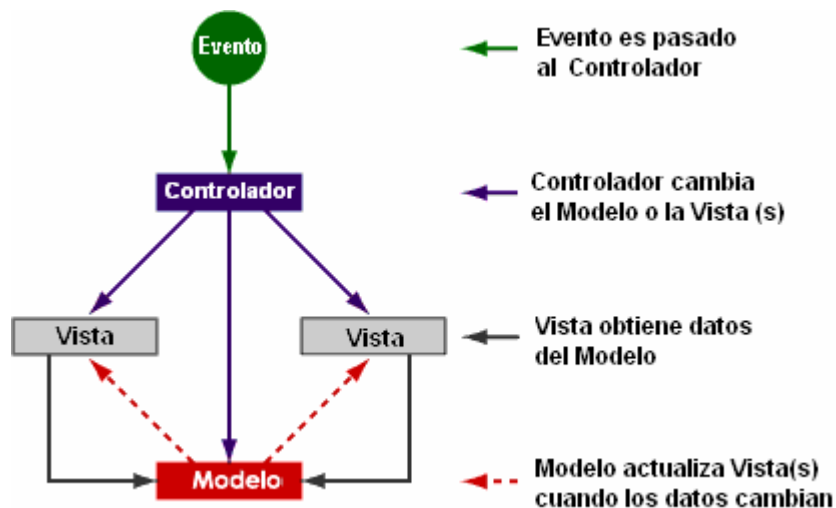
http://www.developer.com/design/article.php/10925_3526681_2. (05/01/2008)

3.3. El patrón MVC

MVC son las siglas de *Model View Controller* (Modelo Vista Controlador). Es un patrón de diseño que divide la aplicación en 3 partes:

- **Modelo:** Se encarga de la persistencia y manipulación de los datos. Representa los datos de la organización y las reglas del negocio que controlan el acceso y modificación de estos datos.
- **Vista:** Se encarga de la interfaz de usuario. Su responsabilidad es mantener la consistencia en la presentación cuando el modelo cambia.
- **Controlador:** Es el intermediario entre el modelo y la vista. Ambos no pueden interactuar directamente. También contiene la lógica del negocio.

Figura 5. Funcionamiento general del patrón MVC



Fuente: *Model-View-Controller Pattern*.

<http://www.enode.com/x/markup/tutorial/mvc.html>. (10/01/2008)

En esta figura se observa de modo general como se relacionan estos componentes. Cuando un evento se dispara el controlador cambia la vista o el modelo, y en algunos casos los dos. Cada vez que el controlador cambia el modelo, todas las vistas dependientes son automáticamente actualizadas. De igual manera cuando el controlador cambiar la vista, puede obtener los datos del modelo para actualizarse.

Al principio de este capítulo se menciona si la aplicación se diseña utilizando el modelo MVC, entonces en la aplicación *Web* tradicional el controlador y la vista suelen estar en el servidor de aplicaciones. Mientras que el modelo se encuentra en la base de datos.

Existe una diferencia entre el patrón de diseño MVC de una aplicación *Web* tradicional y una basada en AJAX. En el primero, el patrón MVC está distribuido fuera del explorador del usuario y lo único que realiza el explorador es visualizar HTML, que es la vista generada en el servidor. Con AJAX la vista es local al usuario, porque toda la lógica de presentación se encuentra en el explorador. El modelo y el controlador están distribuidos. Aunque algunas partes del controlador pueden estar también en el lado del usuario.

El patrón MVC puede ser visto de forma global, por ejemplo, el modelo y el controlador en el servidor y la vista en el explorador. En las aplicaciones de escritorio este patrón es muy a menudo aplicado a diferentes niveles (Crane op.cit., 2006,120). Por ejemplo, un *ComboBox* puede tener el modelo que son los datos que muestra. La vista es el *ComboBox* plasmado en el explorador y el controlador son las funciones que se encargan de los eventos que genera el usuario al interactuar. Visto desde esta perspectiva la aplicación que se carga en el explorador también podrá aplicarse el patrón MVC.

Entonces, en primer lugar, el modelo correspondería a las estructuras que utilizamos en *Javascript* para guardar temporalmente los datos provenientes del servidor o el estado de alguna operación. Estas estructuras son comúnmente *arrays* u objetos. Cuando cambian estos datos, necesitan reflejarse en el servidor o en la interfaz de gráfica, entonces el modelo se comunica con el controlador para que este se realice los cambios necesarios.

En segundo lugar, la vista correspondería a la página HTML que se carga al principio. Esta página sería la encargada de mostrar todos los elementos visuales así como la generación de los eventos asociados a la interfaz gráfica. Otra función es la de actualizarse cuando ocurra un cambio en el modelo o cuando el resultado de un proceso requiera la actualización de la interfaz gráfica.

Hay que tener cuidado de separar adecuadamente la vista del controlador. Una manera de hacerlo es escribir distintos archivos.

Por ejemplo, podemos tener un archivo Javascript que se encarga de gestionar todos los eventos generados por la vista. Otro archivo para realizar los cambios en la interfaz gráfica. Uno o varios archivos CSS para definir el estilo de la página. Una estrategia para mantener la vista y el controlador separados es programar la asignación del evento al control gráfico en lugar de asignárselo explícitamente en la declaración del mismo.

En tercer lugar, el controlador actúa de intermediario entre la vista y el modelo. En el caso de la aplicación AJAX, está formado por todas las funciones que manejan los eventos sean estos hechos por el usuario o por los cambios que ocurrieran en el modelo.

3.4. Modelo de eventos

En la sección anterior se mencionan los aspectos fundamentales del patrón MVC. En el caso de una aplicación AJAX, el controlador tiene un papel importante, porque se encarga de comunicarse con el servidor y realizar cambios en la página. El controlador encargado de gestionar los eventos generados por el usuario y el explorador, tiene dos modelos a disposición para llevar a cabo esta función.

Los exploradores permiten 2 modelos para el manejo de eventos.

- El primero el modelo clásico de manejo de eventos, es sencillo basado en Javascript.
- El segundo es el modelo W3C de manejo de eventos, es más complejo que el primero y no esta estandarizado en todos los exploradores.

El modelo clásico se basa en algunas propiedades para el manejo de eventos de los elementos DOM. El funcionamiento consiste en asignarle una referencia de una función. Entonces cuando ocurra el evento, se llama a la función que esta referenciada en la propiedad. La forma de hacerlo es la siguiente:

```
elemento.propiedad = realizar_accion;
```

La siguiente tabla muestra las propiedades más comunes para el manejo de eventos.

Tabla XVIII. Propiedades para el manejo de eventos en el explorador

Propiedad	Descripción
<i>onclick</i>	Se activa cuando el ratón da un clic a una región del elemento.
<i>Onfocus</i>	Se activa cuando el elemento tiene el foco
<i>onblur</i>	Se activa cuando el elemento pierde el foco
<i>onmouseover</i>	Se activa cuando el puntero del ratón pasa sobre el la regio del elemento
<i>Onmouseout</i>	Se activa cuando el puntero del ratón ha pasado sobre la región del elemento

Este modelo tiene la desventaja de asignarle solo una función de manejo de eventos a un elemento determinado.

El manejo de eventos de W3C si permite la asignación de varios manejadores de eventos a un mismo elemento. Esto es realizado por medio de dos funciones: una para agregar un manejador y otro para eliminarlo. La desventaja de este modelo es que no está soportado por todos los exploradores y no esta implementada siguiendo a cabalidad el estándar.

Una forma de darle solución a este problema es el uso de patrones de diseño. Los patrones *Facade* y *Observer* pueden ser de utilidad para diseñar un manejador de eventos que sea funcional de acuerdo a los requerimientos y sea compatible con la mayoría de exploradores.

3.5. Tecnologías alternativas

Actualmente, AJAX no es la única ni la mejor solución para las soluciones que necesitan algo más que HTML. Existen otras opciones que proveen características que AJAX no soporta. Entre estas características se encuentran: el soporte multimedia y el acceso a periféricos (*Web cams* y micrófonos).

Flash y Java son las tecnologías más comunes cuando se necesita tener una aplicación Web que tenga elementos que HTML no puede proporcionar como:

- Interactividad.
- Interfaz de comunicación síncrona y asíncrona con el servidor.
- Interfaces de usuario vistosas.
- Acceso a periféricos.
- Gráficas de vectores.

Además de Flash y Java existen otras tecnologías emergentes de empresas como Microsoft y Mozilla que responden a estas tecnologías con XAML y XUL respectivamente

3.5.1. Flash

Flash es una tecnología desarrollada por Macromedia y ahora propiedad de Adobe Systems desde el 2005. Con esta tecnología es posible crear aplicaciones interactivas, animaciones e interfaces de usuario visualmente ricas.

Flash fue creado para que fuera usado fácilmente, en comparación de los *applets* y otros productos que requieren mayores destrezas de programación. Las aplicaciones en Flash usan *Action-Script* como lenguaje de programación. Tiene soporte para una gran variedad de controles gráficos, buen soporte de gráficas de vectores y soporte para audio y video.

Flash necesita tener instalado un *plug-in* en el explorador para poder visualizar la aplicación. Es multiplataforma, porque tiene un *plug-in* para la mayoría de sistemas operativos y exploradores. Para poder desarrollar una aplicación con Flash se necesita obtener una licencia para el uso de las herramientas que esta empresa provee.

Ventajas:

- Multiplataforma.
- Rápido.
- Herramientas de desarrollo sofisticadas.
- Soporte nativo de gráficas de vectores, audio y video.

Desventajas:

- Necesita *plug-in*.
- No está integrado totalmente con el explorador (botón de actualización, botón de atrás y *bookmarks*).
- El contenido de una aplicación Flash no puede ser indexado por los buscadores.

Figura 6. Ejemplo de una aplicación Flash



Fuente: **Vialibre.**
<http://www.vialibrepl.com/>. (10/03/2008)

3.5.2. Java Applets

Otra opción para desarrollar aplicaciones *Web* que proporcionen toda la funcionalidad de una aplicación de escritorio son los *applets* de Java. Un *applet* es un programa compilado escrito en Java, que se encuentra incrustado en una página HTML. Al igual que Flash un *applet* no puede ejecutarse por sí solo. Necesita de la máquina virtual de Java para poder funcionar.

Un *applet* a diferencia de una aplicación de escritorio desarrollada en Java, tiene algunas restricciones de seguridad como:

- Generalmente no tienen acceso al sistema de archivos, por lo que no puede leer o escribir archivos.
- No puede cargar librerías nativas del sistema operativo.
- No puede ejecutar programas en el lado del usuario.
- Algunos *applets* no funcionan en determinadas versiones de la máquina virtual de Java.

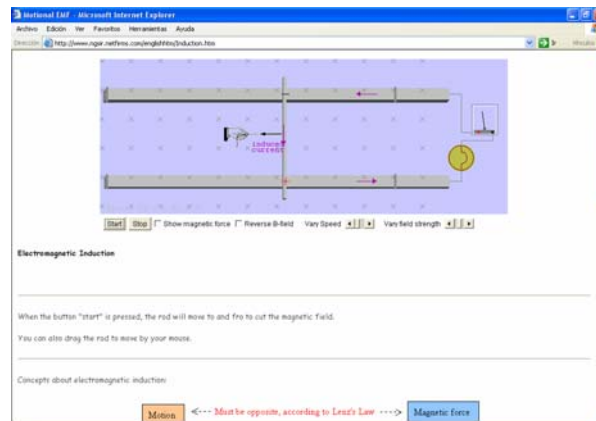
Ventajas:

- Rápido.
- Multiplataforma.
- Robusto.

Desventajas:

- Necesita la máquina virtual de Java.
- La primera vez que se carga toma un poco de tiempo.
- El contenido de los *applets* no puede ser indexado por los buscadores.

Figura 7. Ejemplo de un Java Applet



Fuente: *Physics Java Applets*.

<http://www.ngsir.netfirms.com/englishhtm/Induction.htm>. (10/03/2008)

3.5.3. XAML

XAML son las siglas de *Extensible Application Markup Language*. Es un lenguaje de marcas de alto rendimiento para crear interfaces de usuario dinámicas y visualmente ricas. Es parte de *WPF (Windows Presentacion Foundation)* la nueva generación de interfaces de usuario de Microsoft.

Está basado en XML y permite el desarrollo de interfaces complejas de usuario en aplicaciones de escritorio y *Web*. Cada etiqueta XAML tiene su correspondiente clase en *.Net Framework*.

Está disponible en Windows Vista y en versiones superiores al Internet Explorer 6.0.

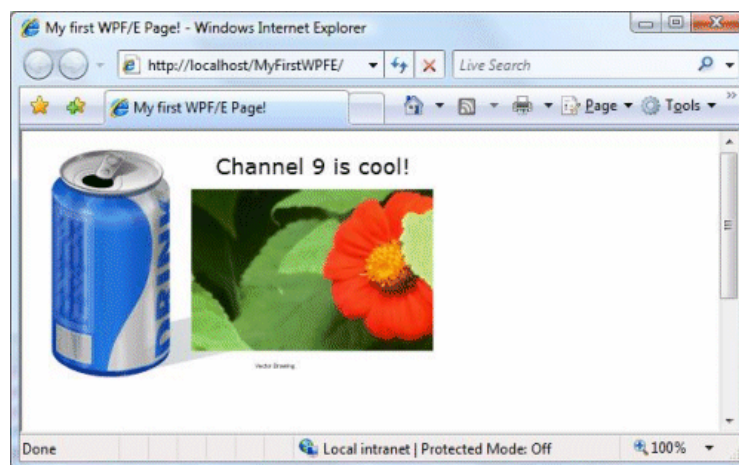
Ventajas:

- Alto rendimiento.
- Robusto.
- Altamente configurable.
- Compatible con varios exploradores a través de un *plug-in*.

Desventajas:

- En los exploradores que no tienen el WPF se tienen que instalar un *plug-in*.
- Requiere el *.Net Framework* desde la versión 3.0.

Figura 8. Ejemplo de una aplicación utilizando XAML



Fuente: **Introducción a WPF/E.**

<http://www.microsoft.com/spanish/msdn/articulos/archivo/150107/voices/bb190632.msp>. (10/03/2008)

3.5.4. XUL

XUL son las siglas de *XML-based User-interfase Language*. Al igual que XAML es un lenguaje de marcas para crear interfaces de usuario dinámicas y visuales. Es parte de la familia de exploradores Mozilla y otras aplicaciones relacionadas.

Esta tecnología provee una colección de controles visuales de alto rendimiento, para poder crear aplicaciones sofisticadas y complejas. Utiliza XBL (*Extensible Bindings Language*) para definir nuevos elementos, contenido y manejadores de eventos para las interfaces XUL. Permite extender XUL, al personalizar los elementos existentes para crear nuevos tipos.

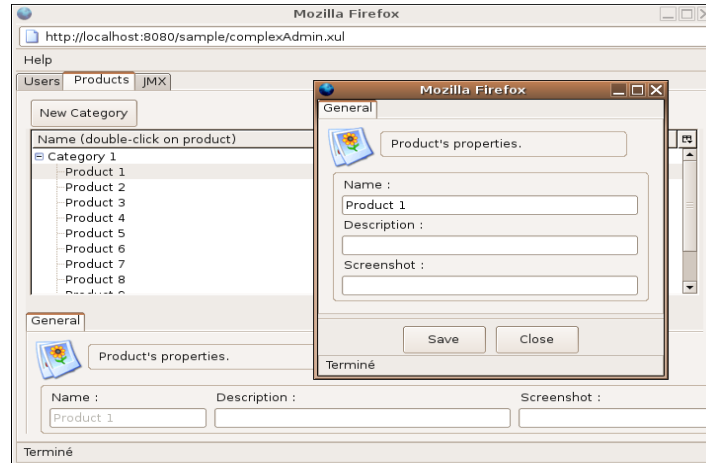
Ventajas:

- Alto rendimiento.
- Soporte de Javascript.
- Basado en XML.
- Es multiplataforma.

Desventajas:

- Necesita el *Gecko Runtime Environment (GRE)* para funcionar

Figura 9. Ejemplo de una aplicación Web basada en XUL



Fuente: **Xulfaces**.

<http://xulfaces.sourceforge.net/screenshots.html>. (10/01/2008)

3.5.5. Comparación de tecnologías alternativas y AJAX

En los apartados anteriores se resumen las principales características de Flash, Java *applets*, XAML y XUL. Cada una de estas tecnologías tiene ventajas y desventajas, dependiendo del enfoque de la aplicación. La siguiente tabla comparativa tiene como objetivo, ayudar a decidir que tecnología es la mejor de acuerdo a los requerimientos de la aplicación que se quiere desarrollar.

Tabla XIX. Comparación de tecnologías alternativas y AJAX

Característica	AJAX	Flash	Java Applets	XAML	XUL
Actualización en tiempo real	Si	Si	Si	Si	Si
Soporte de audio y Video	No	Si	Si	Si	No
Es necesario un <i>plug-in</i> o componente.	No	Si	Si	Si	Si
Flexibilidad para desarrollar nuevas interfaces	Si	Si	Si	Si	Si
Proporciona un IDE para desarrollar aplicaciones	No	Si	Si	Si	Si
Multiplataforma	Si	Si	Si	Si	Si
Necesita Licencia	No	Si	Si	Si	No
Costo de Desarrollo	Bajo	Medio	Medio	Alto	Bajo
Soporta gráficas de vectores	No	Si	Si	Si	No
Acceso a recursos del sistema	No	Si (Acceso a algunos periféricos como <i>Web cams</i> y micrófonos.)	Si (Tiene restricciones para acceder al sistema de archivos)	Si (Acceso a periféricos como <i>Web cams</i> y micrófonos a través de WPF)	Si (Escritura y lectura archivos.)
Dificultad para mantener el código	Alta	Alta	Medio	Medio	Bajo

3.5.6. Eligiendo la mejor opción

En los apartados anteriores se resumen y comparan las características más importantes de AJAX, Java *Applets*, Flash, XUL y XAML.

Entonces a la hora de elegir una determinada tecnología, es necesario decidir si realmente el HTML es suficiente para los requerimientos de la aplicación, o se necesita otra opción. Entre los aspectos a tomar en cuenta para el HTML se encuentran:

- Interfaces de usuario simples.
- Poco nivel de interactividad.
- Incompatibilidades entre exploradores no afectan de forma importante el acceso a la aplicación.

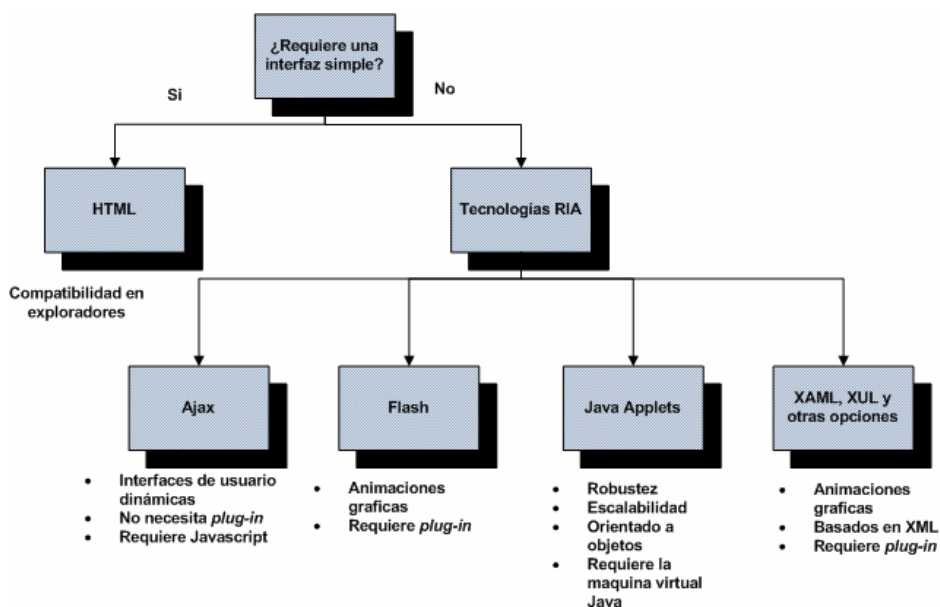
Si los requerimientos de la aplicación requieren el uso de una tecnología RIA, entonces existen las siguientes opciones:

- **AJAX.** Cuando se necesita que la aplicación se ejecute en la mayoría de los exploradores y no se necesite instalar ningún *plug-in*, además de tener interfaces de usuario más complejas que las del HTML normal. Por la propia naturaleza de Javascript es posible que no todas las interfaces de usuario que se necesiten estén disponibles o en rendimiento se vea superado por otras opciones como Java.
- **Java Applets.** Cuando se necesita estar respaldado por una tecnología ampliamente reconocida y provea herramientas de desarrollo de alta calidad, escalabilidad, seguridad y robustez. Además este tipo de aplicación puede ser ejecutada en una amplia variedad de sistemas operativos y plataformas como PDAs y celulares. Las interfaces de usuario disponibles son muchas, además de la gran variedad de controles gráficos que tiene, se pueden agregar otros controles ya sean estos comprados o desarrollados internamente. Hay que tener en cuenta que un programa desarrollado en una versión específica de la máquina virtual, puede no funcionar en otra versión.

- **Flash.** Cuando se necesita una aplicación que tenga animaciones o manipulación de gráficas de vectores. Al igual que Java, tiene la desventaja de las distintas versiones. Esto se debe tomar en cuenta ya que algunas versiones de Flash, no son soportados en algunos sistemas operativos y plataformas. En lo que se refiere a las interfaces de usuario, tiene ciertas desventajas ante Java porque Flash es diseñado para realizar animaciones, presentaciones y películas.
- **XAML.** La opción de Microsoft para la generación de interfaces visualmente ricas para aplicaciones *Web* y de escritorio. El inconveniente que presenta esta tecnología es que está atada a productos Microsoft, por lo que hay que pagar una licencia para poder utilizar las herramientas.
- **XUL.** Es similar a XAML en el sentido que utiliza XML para definir las interfaces gráficas. Pero se ve superado en funcionalidad y robustez, ya que XAML es parte del .Net Framework 3.0 que ya tiene muchos años en el mercado. La principal ventaja de XUL es que es *open source*.

En la siguiente figura se resume en forma de árbol lo descrito anteriormente.

Figura 10. Árbol de decisión entre tecnologías



3.6. Ejemplos de aplicaciones AJAX

3.6.1. Google maps

Este es uno de los ejemplos típicos de AJAX. Esta aplicación permite ver mapas de distintas regiones del mundo así como buscar información sobre esos mapas.

El usuario puede navegar por el mapa usando el ratón. El mapa está compuesto por pequeñas imágenes y cuando el usuario explora el mapa, las imágenes que se necesitan se bajan asincrónicamente del servidor. Las imágenes se almacenan en una *cache* para mejorar el rendimiento.

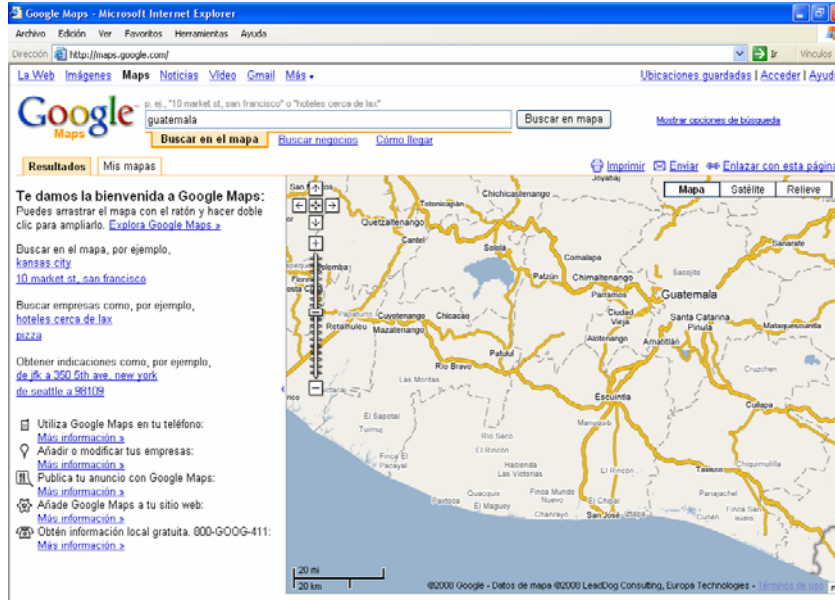
Ventajas:

- Permite una navegación fluida.
- Permite salvar imágenes del mapa.

Desventajas:

- Mensajes de error que no informan si la localidad no está disponible en el sistema.
- Pequeño desfase de la imagen del mapa cuando se refresca el mapa. Esto depende del ancho de banda.

Figura 11. Google Maps



Fuente: Google Maps.

<http://maps.google.com/>, (10/02/2008)

3.6.2. Gmail

Este es otro de los ejemplos típicos de AJAX desarrollado por Google. La aplicación *Web* ofrece un servicio gratuito de correo electrónico.

Esta aplicación *Web* fue lanzada a principios del 2004. Entre las características que sobresalen se encuentran: el tamaño del buzón, la interfaz permite abrir varios correos de una sola vez, actualización automática de la lista de correos, visualización de algunos tipos de documentos y más recientemente la modificación de documentos propios de *Microsoft Office*.

Gmail usa varios *frames* e *iframes* para administrar y guardar los cambios hechos por el usuario. Esto permite que funcionen adecuadamente los botones de atrás y adelante del explorador.

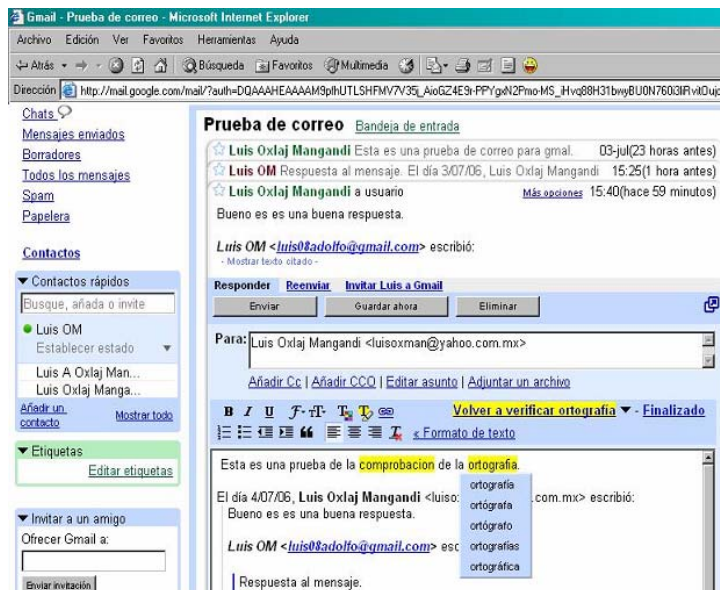
Ventajas:

- Interfaz sencilla y fácil de usar.
- Rápido.
- Opciones para la fácil manipulación de los correos.
- Funciona en todos los exploradores modernos.
- Integra la opción de *Chat*.

Desventajas:

- Al principio se tarda un poco en cargar la página *Web*.

Figura 12. Gmail



Fuente: **Gmail**.

<http://mail.google.com>. (10/02/2008)

4. FRAMEWORKS PARA LOS LENGUAJES WEB MÁS COMUNES

4.1. Clasificación de los *frameworks*

Desde que AJAX vio la luz en febrero de 2005, las aplicaciones *Web* basadas en AJAX están creciendo cada vez más. Como consecuencia la cantidad de herramientas y *frameworks* también está creciendo.

Existen diversos tipos de alternativas. Algunas son *open source*, otras con licencia y cada una con diferentes características. Pero todos con un fin, facilitar el desarrollo. Todas estas alternativas pueden desarrollarse como parte de la aplicación, pero muchas de estos componentes son comunes no importando el tipo de la aplicación. Hakman (2006) propone una clasificación para todas estas alternativas:

- **Librerías de comunicación.** Proveen componentes genéricos para realizar la comunicación asíncrona., así como manipulación de DOM y CSS.
- **Componentes para interfaces de usuario.** Proveen interfaces gráficas que se comunican asíncronamente con el servidor, y luego se actualizan cuando es necesario. Usadas comúnmente para mejorar algún aspecto de la página HTML.
- **RIA *frameworks*.** Son librerías que proveen componentes gráficos de usuario interactivos. Tienen un esquema común de adquisición de datos, persistencia y comunicación. Son utilizadas para crear aplicaciones *Web* totalmente interactivas y con funcionamiento similar a una de escritorio.
- **RIA *frameworks* con herramientas visuales robustas.** Tienen las mismas características que la anterior clasificación, pero también incluye un IDE o un *plug-in* que puede ser utilizado en alguna herramienta de desarrollo.

En el resto del capítulo se mencionan algunos *frameworks* para Java, PHP, .Net y Javascript. Cada uno de estas opciones tiene una gran cantidad de alternativas con diferentes características. La elección dependerá de las necesidades de la aplicación y de los recursos asignados.

La nomenclatura que se usa para distinguir el tipo de *framework* es la siguiente:

- A. Librerías de comunicación.
- B. Componentes para interfaces de usuario.
- C. RIA *frameworks*.
- D. RIA *frameworks* con herramientas visuales robustas.

4.2. Java

En la tabla de abajo se describen algunos *frameworks* para Java. Existe una gran cantidad disponible en el mercado dado la popularidad de Java. En la tabla solo se mencionan los más comunes.

Tabla XX. Frameworks de AJAX para Java

	Descripción	Exploradores soportados	Licencia	Tipo
AJAXAnywhere	Es un <i>framework</i> que convierte los componentes existentes de JSF, JSP y cualquier otro, en componentes AJAX sin tener que programar Javascript. Sigue el patrón MVC. Dirección de Internet: http://AJAXanywhere.sourceforge.net (01/03/2008)	Microsoft Explorer, Mozilla Firefox y Opera.	<i>Open source.</i>	A

DWR (Direct Web Remoting)	Es un <i>framework</i> permite llamar métodos de Java directamente desde Javascript. Es soportado en la mayoría de Web <i>frameworks</i> como struts, spring y tapestry. Dirección de Internet: http://www.getahead.ltd.uk/dwr/ . (01/03/2008)	Microsoft Explorer, Mozilla Firefox, Netscape, Opera, Safari, y Konqueror.	<i>Open source.</i>	A
SWATO	Es un <i>framework</i> compuesto por un conjunto de librerías Java y Javascript. Incluye un conjunto de componentes gráficos para el lado del cliente. Utiliza un archivo XML para realizar la configuración. Dirección de Internet: https://swato.dev.java.net/ . (01/03/2008)	Microsoft Explorer y exploradores basados en Mozilla.	<i>Open source.</i>	C
WidgetServer	Es un <i>framework</i> que permite desarrollar aplicaciones AJAX utilizando solamente Java. Genera automáticamente los archivos necesarios como HTML y Javascript. Dirección de Internet: http://www.c1-setcon.de/widgetserver/ . (01/03/2008)	Microsoft Explorer y exploradores basados en Mozilla.	<i>Open source y comercial.</i>	D

4.3. PHP

En la tabla de abajo se describen algunos *frameworks* para PHP. La cantidad de *frameworks* disponibles es tan grande como las disponibles para Java. En la tabla solo se mencionan los más comunes.

Tabla XXI. Frameworks de AJAX para PHP

	Descripción	Exploradores soportados	Licencia	Tipo
AJAXAC	<p>Este <i>framework</i> encapsula toda la aplicación AJAX en una clase PHP. Tiene incorporado el manejo de eventos Javascript, creación y manejo de sub-peticiones, y una estructura extensible de controles gráficos.</p> <p>Dirección de Internet: http://AJAX.zervaas.com.au/. (02/03/2008)</p>	Microsoft Explorer y Mozilla Firefox.	<i>Open source.</i>	A
JPSpan	<p>Es un <i>framework</i> que traduce funciones PHP a su equivalente Javascript. Utiliza reflexión para analizar las clases PHP, y luego crea las funciones JavaScript con el mismo número de parámetros y datos equivalentes.</p> <p>Dirección de Internet: http://sourceforge.net/projects/jpspan. (02/03/2008)</p>	Microsoft Explorer y Mozilla Firefox.	<i>Open source.</i>	A
PEAR HTML AJAX	<p>Es un <i>framework</i> que provee soporte AJAX como parte del proyecto PEAR. Está formado por un conjunto de librerías Javascript y PHP. Sus características más importantes son: librerías PHP y Javascript para agregar funcionalidad AJAX y serialización de datos enviados entre el usuario y el servidor.</p> <p>Dirección de Internet: http://pear.php.net/package/HTML_AJAX. (02/03/2008)</p>	Microsoft Explorer y Mozilla Firefox.	<i>Open source.</i>	A

JPOP	<p>Es un <i>framework</i> que genera todo el código Javascript, hojas de estilo y código HTML para generar aplicaciones interactivas. Incluye un set de controles gráficos para ser incluidos dentro de la aplicación <i>Web</i>.</p> <p>Dirección de Internet: http://sacratee.com/jpop/index.php. (02/03/2008)</p>	Microsoft Explorer y Mozilla Firefox.	<i>Open source.</i>	C
XAJAX	<p>Es un <i>framework que transforma</i> las llamadas Javascript a funciones PHP. Tiene funciones genéricas para la comunicación con el servidor a causa de las distintas implementaciones del objeto <i>XMLHttpRequest</i>. Es posible agregarle controles gráficos a través de <i>plug-ins</i>.</p> <p>Dirección Internet: http://www.xAJAXproject.org/ (02/03/2008)</p>	Microsoft Explorer y Mozilla Firefox.	<i>Open source.</i>	C

4.4. .Net

Otro de los lenguajes de programación para realizar aplicaciones *Web* es .Net. La cantidad de *frameworks* disponibles no es están grande en comparación a los disponibles para Java y PHP. En la tabla solo se mencionan los más comunes.

Tabla XXII. Frameworks de AJAX para .Net

	Descripción	Exploradores soportados	Licencia	Tipo
AJAX.Net	<p>Librería disponible para las versiones 1.1 y 2.0 del <i>framework</i> de .Net. Provee la capacidad de invocar funciones .NET desde el código Javascript del usuario. Además permite el acceso de los datos de la sesión, soporte en Javascript para tipos de datos comunes y complejos de .NET por medio de JSON.</p> <p>Dirección Internet: http://AJAX.schwarz-interactive.de/csharpsample/default.aspx. (02/03/2008)</p>	Microsoft Explorer.	<i>Open source.</i>	A
ASP.Net AJAX	<p>Es un proyecto de Microsoft para agregar soporte AJAX a su plataforma de desarrollo .NET. Provee un conjunto de librerías para ser usadas en el cliente, integración con <i>Web Services</i> y nuevos controles ASP.NET. Es soportado por la versión 2.0 y 3.0 del <i>framework</i> .Net.</p> <p>Dirección Internet: http://www.asp.net/AJAX/. (02/03/2008)</p>	Microsoft Explorer.	<i>Open source.</i>	D
MagicAJAX .NET	<p>Es un <i>framework</i> que integra la funcionalidad AJAX a los controles ASP.NET sin tener que programar código Javascript. La forma de funcionamiento consiste en agregar los controles deseados en un panel especial, y automáticamente generará el código Javascript necesario para tener una aplicación AJAX.</p> <p>Dirección Internet: http://www.magicAJAX.net/ (02/03/2008)</p>	Microsoft Explorer, Firefox, Netscape y Opera.	<i>Open source.</i>	A

Visual WebGUI	<p>Es un <i>framework</i> que permite crear una aplicación AJAX como si estuviera utilizando controles <i>WinForms</i>. Todo la programación es del lado el servidor, todos los archivos Javascript, CSS, XML y HTML se general automáticamente. Disponible para las versiones 1.1, 2.0 y 3.5 del <i>framework</i> .Net.</p> <p>Dirección Internet: http://www.visualwebgui.com/. (02/03/2008)</p>	Microsoft Explorer y Firefox.	<i>Open source.</i>	D
----------------------	---	-------------------------------	---------------------	---

4.5. Javascript

Los primeros *frameworks* que empezaron a salir al mercado fueron los de Javascript. Estos además de la comunicación asíncrona con el servidor, proveen de componentes gráficos listos para usar en cualquier página Web. En la tabla solo se mencionan los más comunes.

Tabla XXIII. Frameworks de AJAX para Javascript

	Descripción	Exploradores soportados	Licencia	Tipo
Bindows	<p>Es un <i>kit</i> de Desarrollo SDK que genera aplicaciones Web altamente interactivas, que provee de una gran cantidad de componentes visuales. El API es orientado a objetos y tiene soporte nativo de XML, SOAP y XML-RPC.</p> <p>Dirección Internet: http://www.bindows.net/. (02/03/2008)</p>	Microsoft Explorer, Firefox, Netscape y Opera.	Comercial	D

Dojo	El objetivo es la de proveer una herramienta para el desarrollo de aplicaciones <i>Web</i> que sean visualmente ricas, además de enfocarse también en la usabilidad. Dirección Internet: http://dojotoolkit.org/ (02/03/2008)	Microsoft Explorer, Firefox, Safari, Konqueror y Opera.	<i>Open source.</i>	B
Rico	Es un <i>framework</i> que se enfoca en las interfaces de usuario. Provee varios controles y efectos como: <i>datagrids</i> , <i>tree views</i> , desvanecimiento y movimiento de controles. Dirección Internet: http://openrico.org/ (02/03/2008)	Microsoft Explorer, Firefox y Camino.	<i>Open source.</i>	B
YUI (Yahoo! User Interface)	Es una librería que provee de utilidades y una gran cantidad de controles gráficos para generar aplicaciones interactivas. Dirección Internet: http://developer.yahoo.com/yui/ (02/03/2008)	Microsoft Explorer, Firefox, Opera y Safari.	<i>Open source.</i>	B

5. CONSIDERACIONES DE SEGURIDAD EN AJAX

5.1. Seguridad en AJAX

La seguridad en aplicaciones *Web* se está convirtiendo en un punto importante a tomar en cuenta, esto debido al incremento de los ataques que afectan a cualquier sitio en Internet. Dependiendo del tipo de aplicación *Web*, la seguridad puede ser un aspecto crítico, aún más si involucra transacciones de dinero o datos confidenciales. Estos ataques no afectan únicamente a los sitios de las grandes empresas, sino cualquier aplicación *Web* disponible desde Internet puede ser el objetivo de un ataque.

El desarrollo de aplicaciones AJAX puede traer más consideraciones de seguridad que cualquier aplicación *Web* tradicional, esto por las tecnologías en las que se basa y en la forma en que se diseña. Garret (2006, citado por Frye, 2006a) menciona que hasta cierto punto, cuando se está haciendo una aplicación AJAX, parte de la lógica del negocio se mueve del servidor al cliente. Al mover esta lógica al cliente, se expone a todo el mundo, esto representa riesgos potenciales de seguridad dependiendo de la aplicación. Además la comunicación con el servidor se hace invisible al usuario, ahora hay datos que se transmiten sin el conocimiento del usuario. Esto abre algunos importantes riesgos.

Estos aspectos no son los únicos a tomar en cuenta. Negm (2006, citado por Frye Op.cit) considera las siguientes como posibles cuestiones de seguridad para AJAX:

- **Datos corruptos.** Alguien malintencionado puede realizar un ataque al enviar datos alterados.
- **Acceso sin autorización.** En una aplicación AJAX, un *hacker* puede fácilmente elevar sus privilegios si el servidor no tiene mecanismos de seguridad.

- **Datos malformados.** Esto podría causar una denegación de servicio, al tratar de agotar los recursos de servidor
- **Rendimiento.** La validación de datos, puede afectar el rendimiento del explorador, y con esto la experiencia del usuario se puede ver afectada.

Los puntos mencionados en el párrafo anterior se enfocan en ciertos aspectos específicos de una aplicación AJAX. Pascarello (2006, citado por Frye, 2006b) define algunas reglas generales a tomar en cuenta cuando se desarrolla una aplicación AJAX. Estas no son exclusivas de una aplicación AJAX, muchas de ellas son aplicables a cualquier aplicación *Web*.

- Si se utiliza autenticación de usuario, asegúrese de revisarlo en una forma adecuada.
- Verifique las inyecciones SQL.
- Verifique las inyecciones Javascript.
- Mantenga la lógica del negocio en el servidor.
- No asuma que cada petición es real.
- Verifique los datos con alguna validación.
- Revise la cabecera de la petición y asegúrese que es correcta.

Cualquier aplicación *Web* es propensa de sufrir algún ataque por parte de alguien mal intencionado. En el caso de una aplicación *Web* y AJAX son muchos los tipos de ataques que pueden sufrir. En este tipo de aplicaciones hay dos tipos que merecen una especial atención por lo desastrosos de sus resultados: las secuencias de comandos entre sitios y las inyecciones SQL.

5.2. Secuencias de comandos entre sitios

Las secuencias de comandos entre sitios es una vulnerabilidad de las aplicaciones *Web* que es generada al no validar los datos de entrada. Esta vulnerabilidad también es conocida como CSS, siglas de *Cross Site Scripting*, y XSS.

Al no validar esas entradas, alguien mal intencionado puede incrustar código HTML malicioso. Generalmente este código es Javascript y el propósito de este código consiste en robar los *cookies* de autenticación y cambiar el contenido del sitio. Estos ataques se enfocan en como el contenido HTML es generado e interpretado por los exploradores.

Cuando alguien malintencionado se roba los *cookies*, es porque sabe que los *cookies* pueden almacenar información sensible de los usuarios. Esta información puede abarcar desde usuarios y contraseñas hasta números de cuentas bancarias y tarjetas de crédito. Un ataque consistiría en inyectar código *Javascript* para acceder a la *cookie* y luego lo envía a otro sitio. Al tener los *cookies* de algún usuario, puede usarlos para hacerse pasar por ese usuario. Esto es conocido como *spoofing*.

Por eso hay que tener cuidado qué información se almacena en las *cookies*. La información que es sensible es mejor encriptarla para que no sea fácil robar la información.

Otro objetivo del XSS consiste en cambiar el contenido del sitio. Una de las razones para cambiar el contenido es porque pueden publicar información falsa, así como proveer los medios para robar información del usuario. Un ejemplo de esto sería agregar un link para verificar o actualizar los datos personales de un usuario. Este link podría re direccionar a otro sitio ajeno al original y el usuario confiado podría revelar información importante, como usuarios y contraseñas.

Como se mencionó antes, esto es el resultado de no validar los datos de entrada que utiliza la aplicación, pero también de no codificar en forma adecuada la página resultante. La solución general consistiría en atacar estos dos errores.

El primer punto es la restricción de las entradas. Suponga que todas las entradas no son validas. Entonces valide la longitud, tipo y el formato. Una manera de hacerlo es por medio de expresiones regulares. Con las expresiones regulares podemos definir reglas para que verifiquen las cadenas de entrada. En el caso de las cadenas que representan números hay que convertirlos a su tipo, para verificar si el valor es válido.

El segundo punto es codificar la página resultante. La función de esto es no permitir que se ejecute código que ha sido codificado. Lo interpreta como texto inofensivo. Al codificarlo como HTML se reemplazan los caracteres que tengan un significado especial como los signos de mayor y menor, por otros que se interpretan como texto.

Los objetivos y los métodos para realizar el ataque pueden ser muchos. Los puntos anteriores son una manera de minimizar los riesgos de sufrir un ataque de XSS. La CERT (2000) enumera una serie de pasos para mitigar esta vulnerabilidad.

1. **Fijar la codificación de caracteres para cada página generada por el servidor Web.** Muchas páginas *Web* dejan sin definir la codificación de caracteres. El problema con esto es que a veces el explorador no sabe que caracteres son especiales. La especificación estándar es el ISO-8859-1.
2. **Identificar los caracteres especiales.** La especificación HTML determina que caracteres son especiales. Estos afectan como la página es desplegada. Los desarrolladores deben revisar la aplicación y determinar que caracteres afectan a la aplicación.

3. **Codificar los elementos dinámicos de salida.** Cada carácter en la especificación ISO-8859-1 tiene su equivalente a un valor numérico. Codificar los datos, permite el filtrado y la preservación de la apariencia visual en el explorador. Sin embargo este proceso requiere muchos recursos.
4. **Filtrar caracteres específicos en elementos dinámicos.** No es claro que caracteres o combinación de estos pueden ser usados para exponer otras vulnerabilidades. Es recomendable seleccionar el conjunto de caracteres que son validos, en lugar de excluir los que no lo son. Este proceso de filtrado puede ser parte de la entrada de datos así como en proceso de salida, justo antes de integrarlo como parte de la página HTML.
5. **Examinar los *cookies*.** Un ataque podría consistir en insertar contenido malicioso en la *cookie*. Por eso es importante verificar que las *cookies* no han sido alteradas.

5.3. Inyecciones SQL

Las inyecciones SQL y las secuencias de comandos entre sitios, son los ataques más comunes que sufre un sitio *Web*.

Este tipo de ataque se produce cuando una aplicación utiliza las entradas de datos directamente para crear instrucciones SQL dinámicas. Cuando el ataque es exitoso, la persona malintencionada puede ejecutar comandos de la base de datos.

Esta vulnerabilidad afecta tanto a aplicaciones *Web* como aplicaciones de escritorio. El problema consiste en la forma que se generan las sentencias SQL. Cuando se generan concatenando los parámetros a una cadena, sin realizar ninguna verificación, entonces se abren varios agujeros de seguridad. Uno de estos fallos consiste en poder cambiar la sentencia original o agregar más sentencias para borrar y modificar tablas. Un ejemplo de este fallo es el siguiente:

consulta = “*select* usuario, clave *from* usuario *where* usuario=’” + usuario + “””;

Al generar la consulta el parámetro “usuario” debería tener el identificador del usuario, pero en un ataque el valor podría ser: ’ ; *drop table* seguridad -- . En este caso la consulta quedaría:

consulta = “*select* usuario, clave *from* usuario *where* usuario=’” ;
drop table seguridad --’

En este caso, además de realizar la consulta, borraría la tabla seguridad. Lo que podría hacer el atacante depende de los permisos de la cuenta utilizada para acceder a la base de datos.

Los objetivos de estos ataques pueden ser muchos. Finnigan (2000) enumera dos objetivos primarios por las que se realizan los ataques de inyecciones SQL.

1. Robar datos de la base de datos que normalmente no están disponibles, u obtener datos de la configuración del sistema para construir un perfil de ataque.
2. Obtener acceso a las computadoras de la organización por medio del servidor de la base de datos. Esto por medio de paquetes y extensiones de lenguaje 3GL que permiten el acceso al sistema operativo.

Al igual que las secuencias de comandos entre sitios, esta vulnerabilidad se puede mitigar al aplicar algunas estrategias de filtrado y políticas de seguridad como las siguientes:

- **Validar los datos de entrada.** Cualquier dato que ingrese el usuario, es necesario verificar el tipo, longitud, formato e intervalo.

- **Utilizar parámetros SQL.** La mayoría de los lenguajes de programación, tienen alguna manera de generar sentencias SQL de una forma más segura. Esto por medio de colecciones de parámetros o funciones. El objetivo es hacer una segunda comprobación del punto anterior. En el caso de alguna incongruencia se genera un error o excepción.
- **Utilizar una cuenta con permisos reducidos.** La cuenta que accede a la base de datos debe tener solo los permisos mínimos para poder funcionar correctamente. De esta manera si alguien tiene acceso de manera ilícita a la base de datos, tendrá muchas limitaciones para hacer el ataque.
- **Manejar adecuadamente los errores.** Entre las técnicas que se utilizan para realizar un ataque de inyección SQL, se encuentra la de generar errores en la base de datos, para poder obtener nombres de tablas y campos. Por esa razón hay que asegurarse que en caso de error, no se dé un mensaje detallado al usuario sino uno simple y sin mayores detalles.

5.4. Protección de datos confidenciales

Las secciones anteriores se enfocan en las vulnerabilidades más comunes que pueden haber en una aplicación *AJAX* y *Web* cuando no se desarrolla pensando en la seguridad. Aparte de esos puntos también hay que prestarle atención a como los datos se transmiten sobre la red y en la forma de almacenarla.

Los datos que se transmiten sobre HTTP, van en texto plano, permitiendo que cualquiera pueda leerlos o modificarlos antes de que lleguen a su destino.

Una alternativa es usar HTTPS, que utiliza el modelo de la clave pública y privada para encriptar los datos en ambos sentidos.

HTTPS requiere soporte nativo en el explorador como en el servidor. Este protocolo asegura solo la transmisión de los datos, no es la solución completa para el problema de la seguridad. Pero es recomendado para transmitir información sensible sobre la red. Una alternativa cuando no se dispone de HTTPS es encriptar los datos usando algún algoritmo simétrico de encriptación en Javascript. MD5 es un algoritmo simétrico, que puede usarse para encriptar los datos. La mayoría de lenguajes del lado del servidor tiene este algoritmo disponible. También existen versiones en Javascript de este algoritmo.

Esta es una solución que tiene sus desventajas, el algoritmo utilizado en Javascript puede ser estudiado por medio de ingeniería inversa. También el código Javascript revela la lógica de encriptado, permitiendo que alguien pueda crear un sistema para ver los datos y romper la seguridad.

No solo hay que proteger los datos que viajan en la red, también se deben proteger los datos confidenciales que se encuentran en la base de datos.

5.5. Javascript y seguridad del explorador

Esta sección se enfoca en algunas políticas de seguridad que tienen los exploradores. Estas políticas son importantes, porque sino se toman en cuenta pueden afectar el funcionamiento de la aplicación AJAX.

Cuando se carga una aplicación AJAX en el explorador, todo el código Javascript necesario se transmite sobre la red y luego se ejecuta. Esta característica es propia del llamado código móvil. Es código móvil si está almacenado en una determinada máquina, y puede transmitirse a través de la red para ejecutarse en otra (Crane et al., 2006, 248).

Esta característica del código móvil, hace que los exploradores incluyan varias políticas de seguridad. Javascript se ejecuta en un ambiente llamado caja de arena. Este ambiente aislado hace que Javascript no tenga acceso a los recursos del sistema, como el sistema de archivos.

Otra política es la restricción de solo permitir la conexión e interacción de *scripts* para un solo un dominio, esto conocido como política de servidor de origen. Esta política previene que alguien mal intencionado pueda hacer modificaciones para cambiar el comportamiento original de la aplicación.

Este modelo, no permite la comunicación con otros *scripts* que no son los del origen, pero hay ocasiones cuando es necesario, por ejemplo al tratar de acceder a un *Web Service* de otro dominio.

Hay que tener cuidado con los datos que provienen de servidores de terceras personas, porque pueden estar modificadas, para hacer daño cuando se *parsean*, borrando o modificando información vital.

6. VENTAJAS Y DESVENTAJAS DE AJAX

6.1. Ventajas

6.1.1. Rendimiento

Una de las ventajas de una aplicación AJAX, es el mejor rendimiento que puede tener en determinados escenarios. Pero también existe la posibilidad que el rendimiento no sea lo suficientemente bueno para cambiar de una aplicación *Web* tradicional a una AJAX.

En los capítulos anteriores se mencionan los aspectos positivos de este enfoque. Entre esas ventajas se encuentran: menor cantidad de datos transmitidos desde el cliente y el servidor, rapidez de respuesta y menor consumo de ancho de banda.

La mejor manera y la más objetiva de comparar una aplicación *Web* tradicional y una AJAX es desarrollar las dos versiones. De esta manera es posible medir determinadas variables en determinados escenarios, para luego realizar algún estudio estadístico.

En el capítulo ocho se desarrolla un ejemplo utilizando AJAX. Se utiliza el *framework Magic* AJAX que tiene la característica de poder desactivar la funcionalidad AJAX. Después de presentar el ejemplo se realiza una comparación de los bytes transmitidos y el tiempo necesario para completar la tarea cuando se usa y no se usa AJAX. Los resultados son satisfactorios cuando se utiliza AJAX disminuye la cantidad total de bytes transmitidos así como el tiempo necesario para completar las tareas.

Estas mejoras se deben a muchos factores entre los que se mencionan:

- Uso del *cache* del explorador.
- Transferencia de datos en lugar de páginas.
- Menor carga al servidor.

6.1.2. Mejores Interfaces de usuario

Una aplicación AJAX puede ser intuitiva, además de proveer interacción natural al usuario. Por ejemplo no es necesario dar clic a un control para realizar alguna acción, el movimiento del ratón es suficiente para disparar un evento.

La comunicación asíncrona reemplaza el modelo síncrono petición-respuesta. Ahora el usuario puede seguir utilizando la aplicación mientras la aplicación pide datos al servidor en segundo plano.

Al integrar AJAX a una aplicación Web es posible mejorar las interfaces de usuario y controles con:

- Vistas de árbol.
- Menús.
- Barras de progreso.
- Pestañas.
- Calendarios.

Esto brinda una mejor experiencia ya que no se necesita actualizar la página cada vez que se utilizan estos controles. También la funcionalidad que agrega AJAX a las interfaces de usuario estándar, tiene sus beneficios en situaciones como las siguientes:

- **Búsqueda de una palabra en un diccionario.** Por ejemplo, mientras se escribe la palabra en un área de texto, aparecen posibles alternativas de la palabra y al finalizar de escribirla aparece la definición de la misma.
- **Para llenar formularios.** Un escenario típico es cuando se llena un formulario para registrarse a un sitio. Por ejemplo, se validan los campos automáticamente mientras se ingresan los datos y se obtienen nuevos datos para mostrar en el formulario, sin necesidad de actualizar la página.
- **Selección de opciones.** Cuando se necesita que el usuario seleccione una opción de un *combo box* para cambiar parte de las opciones del usuario o para filtrar los datos que utilizaran otras interfaces de usuario, todo este proceso sin necesidad de actualizar la página. Por ejemplo, un *combo box* para seleccionar un país y en otro *combo box* se elige una ciudad del mismo.

6.1.3. Mejor Interacción

Como se mencionó anteriormente, las aplicaciones AJAX realizan peticiones al servidor para traer solo los datos que necesitan. Esto permite que las interfaces de usuario tengan una mejor respuesta porque la cantidad de datos entre el cliente y el servidor se reduce al no tener que transmitir de nuevo una página entera.

Cada elemento dentro la interfaz de la página puede ser rápidamente e individualmente actualizada sin afectar el resto de la interface grafica.

La actualización parcial de la página reemplaza la carga completa del documento del modelo tradicional de una aplicación *Web*. Esto resulta en una respuesta más rápida, porque solo los elementos visuales que contienen nueva información son actualizados. Además, el resto de la interface del usuario permanece funcional sin interrupción, manteniendo el flujo de trabajo del usuario.

Estas características permiten el desarrollo de aplicaciones *Web* más interactivas, ya que se pueden diseñar interfaces de usuario más inteligentes y ergonómicas.

Tareas como actualizar y eliminar registros de una tabla, retornar el resultado de una consulta, llenar un formulario y otras tareas comunes pueden realizarse de una manera mucho más fácil y rápida.

6.1.4. Portabilidad

Las aplicaciones AJAX están basadas en estándares *Web* que permiten su funcionamiento en la mayoría de exploradores de los sistemas operativos. Esta es una de las ventajas que tiene AJAX con comparación de las demás opciones. Solo es necesario que el explorador tenga habilitado Javascript y la versión del explorador sea lo suficientemente reciente para poder utilizar el objeto *XMLHttpRequest*.

Esto permite que AJAX sea multiplataforma, aunque siempre existe el inconveniente de las incompatibilidades de los exploradores. Estos inconvenientes se resuelven en parte, detentando el explorador y la versión para realizar los cambios necesarios en la parte del código donde sea específica del explorador.

6.2. Desventajas

6.2.1. Usabilidad

Algunos problemas se derivan de la forma en que funciona una aplicación AJAX. Por ejemplo, los usuarios están acostumbrados a que cada vez que realiza una acción esperan unos segundos para ver la nueva página. Entonces cuando utilizan una aplicación AJAX el cambio puede ser muy leve, permitiendo que se haga bastante rápido, el usuario no se podría dar cuenta y estaría esperando un cambio ya realizado. Además la dirección URL no cambia, confundiéndolo más.

Una posible solución es tener algún indicador lo suficientemente visible, que se encarga de indicarle al usuario de algún cambio o finalización de proceso. Este indicador podría utilizar distintos colores, para indicarle al usuario que ha ocurrido un cambio. Esto puede no ser suficiente por lo que hay que incluir otros signos que actúen como respaldo.

Otro problema es el uso del botón de atrás del explorador, los usuarios están acostumbrados a usarlo para regresar a un estado determinado de la página. En una aplicación AJAX no va funcionar de la misma manera porque los cambios se hacen sobre una página. Todo el proceso hecho por Javascript para generar los cambios, no se graba en el explorador. Esto puede causar que los usuarios no se sientan cómodos con la aplicación o tarden más tiempo de lo normal para aprender a usar la aplicación.

El uso de *bookmarks* o marcado de favoritos, también se ve afectada porque la página principal es la que cambia dinámicamente, entonces no se puede grabar en que punto se dejó la página, porque no ha cambiado la página principal hacia otra página.

En una aplicación AJAX es necesario comunicar a los usuarios que pueden hacer en la página. La interfaz debe mostrar que puede hacer y sobre qué elementos puede realizar el usuario una acción (Fidalgo, 2005, citado por García, 2005). Por ejemplo, si un sitio *Web* para comprar libros permite arrastrar la foto del libro, para agregarla a la carretilla, tiene que haber alguna manera de comunicarle al usuario que puede hacer eso. El usuario no puede estar adivinado que puede o no hacer.

6.2.2. Accesibilidad

Los problemas más grandes se encuentran en las incompatibilidades de Javascript y el objeto *XMLHttpRequest* entre los exploradores. Agregando los exploradores que no soportan alguna de estas tecnologías o no los tengan activados.

Una solución sería tener una versión del sitio que no dependiera de Javascript, pero eso lleva más tiempo y recursos. Y en la mayoría de ocasiones el tiempo y los recursos son limitados. Además algunas aplicaciones basadas en AJAX, no son factibles para convertirlas en aplicaciones *Web* tradicionales, porque perderían parte de su funcionalidad básica.

Un ejemplo sería Google maps. Google maps se basa en AJAX, para ofrecer un servicio fluido cuando se exploran los mapas. Si estuviera construida en la forma tradicional, cada vez que se necesitará recorrer el mapa, tendría que pedir la actualización al servidor. Este le enviaría una página HTML al cliente, y después la visualizaría. Sería tan lento y disfuncional que no valdría la pena desarrollarla.

6.2.3. Problemas más comunes

Con AJAX es posible realizar una gran variedad de aplicaciones. Pero las tecnologías en las que se basa no permite realizar otro tipo de tareas como:

- **Acceso local al sistema archivos.** No es posible escribir o leer del disco duro del cliente.
- **Audio y video.** Una aplicación AJAX no tiene soporte para incluir material multimedia como música y video.
- **Acceso al Hardware.** No se tiene acceso a cámaras *Web*, micrófonos e impresoras.
- **Atajos de teclado.** No se provee un mecanismo para tener atajos de teclado que sean independientes del explorador. Puede ocurrir un conflicto con los definidos en el explorador.
- **Otros protocolos de comunicación.** El único protocolo que puede utilizar una aplicación AJAX para comunicarse con el servidor es el HTTP y HTTPS.

- **Buscadores.** Una aplicación AJAX puede ser difícil de indexar por los buscadores, porque normalmente ignoran el código Javascript.

Otra desventaja es la relacionada con la depuración de aplicaciones. Una aplicación AJAX va tener mucho más código Javascript que una aplicación *Web* tradicional. Si ocurre un error va ser más difícil encontrarlo si no se utiliza alguna metodología de codificación.

Además las herramientas para depurar Javascript son muy pocas y aún no son lo suficientemente maduras, porque el uso de Javascript consiste en su mayoría en realizar validaciones, entonces no era necesario utilizar una herramienta para depurar ese código. Con la introducción del DHTML y AJAX, que utilizan en mayor medida Javascript si se hace necesario una mejor revisión el código.

Bosworth (2005a) lista una serie de errores que los desarrolladores cometen cuando hacen una aplicación AJAX.

- **No dar pistas visuales inmediatas al darle clic a los controles gráficos.** Cuando algún elemento visual dispara alguna acción Javascript, por efecto del ratón, entonces es necesario dar alguna señal visual que algo está sucediendo.
- **Destellando y cambiando partes de la página repentinamente.** AJAX utiliza mensajes asíncronos para comunicarse con el servidor. El problema con este tipo de mensajes es que pueden ser confusos si aparecen repentinamente. Estos cambios deberían ocurrir en porciones bien definidas. Evitar poner mensajes destellantes en lugares que puedan desconcentrar al usuario.

- **No usar links, que pueda pasar a mis amigos o agregar a favoritos.** Una característica de las aplicaciones *Web* tradicionales, es que puedo compartir la dirección de un sitio o agregarla a mis sitios favoritos. El problema es que la aplicación AJAX se genera dinámicamente y el URL no cambia. Por eso algunas aplicaciones AJAX generan algunos *tokens* que agregan al URL original, para simular el cambio de página.
- **Demasiado código hace el explorador lento.** AJAX genera gran cantidad de código Javascript. Entre más código Javascript mayor trabajo para el explorador, lo que significa que se necesita una computadora con más recursos, para esos sitios que codifican de una manera incorrecta.
- **Inventando nuevas convenciones para interfaces de usuario.** Es un error hacer un control que provea una acción no obvia para obtener un resultado no obvio. Un ejemplo sería la capacidad de arrastrar elemento en una determinada área. Al principio el usuario no tendría idea de esa característica. Pero con el tiempo aprendería a usarla. El problema es que el tiempo para aprender a usar la aplicación se incrementa.
- **No extender cambios locales a otras partes de la página.** Cuando se realiza un cambio en la página, porque ha cambiado el estado del mismo, es fácil olvidar cambiar todas partes que no sean las obvias.

7. MEJORES PRÁCTICAS PARA EL USO DE AJAX

7.1. Mejores prácticas utilizando Javascript

Las aplicaciones AJAX utilizan Javascript como lenguaje de programación. Govella (2005), recomienda cinco mejores prácticas para aplicar Javascript de una forma más efectiva.

- 1. Separar el comportamiento del contenido y la presentación.** Es común separar el estilo por medio de archivos CSS, también el código Javascript se debe separar escribiéndolo en varios archivos. La presentación, documentos HTML o XHTML, deberían tener solo los links a estos archivos. El comportamiento de los controles se agrega dinámicamente por medio de DOM. Este enfoque tiene sus ventajas. Permite leer y revisar el código más fácilmente, así como trabajar en varios archivos al mismo tiempo.
- 2. Proveer una mejora progresiva.** El contenido debe permanecer útil y usable sin Javascript. Este debería usarse para agregar funcionalidad al contenido y no para hacerlo inaccesible.
- 3. Codifique para flexibilidad.** Para hacer la codificación del comportamiento más flexible permita que los desarrolladores tomen decisiones en el diseño del comportamiento.
- 4. Altere el contenido tan poco como sea posible.** Si algún evento cambia la presentación y no el contenido, entonces es mejor cambiar la clase CSS, ya que esta es la encargada de la presentación. Ahora si el contenido es el que cambia entonces Javascript debe modificar la estructura del contenido por medio de DOM y asegurarse que el cambio siga generando un documento HTML valido.

5. Documente salidas, parámetros, y dependencias. Es una buena práctica documentar los aspectos importantes de la aplicación. Estos incluyen ejemplos de parámetros, salidas de las funciones, dependencias que tenga las funciones con alguna librería, etcétera. Esto ayuda a revisar y optimizar la aplicación.

Javascript permite realizar muchas acciones de diferentes maneras. Algunas pueden no ser compatibles entre exploradores. Se mencionan algunas sugerencias para algunas acciones comunes.

Propiedades de los objetos. Javascript permite acceder a las propiedades de los objetos de dos maneras.

1. Notación de punto: objeto.propiedad1
2. Notación de corchetes: objeto[“propiedad1”]

Es una buena práctica utilizar la notación de corchetes, porque permite acceder a diferentes propiedades en tiempo de ejecución. Esto es útil cuando se crean nuevos objetos en tiempo de ejecución, porque el uso del punto para acceder a la propiedad no es posible.

Referenciar formas y elementos de las formas. Para referenciar las formas y los elementos se puede utilizar la notación de corchetes o por medio de un índice. Es mejor utilizar la notación de corchetes porque el índice inicial varía de explorador a explorador. En algunos empieza en 0 y en otros en 1.

Ejemplo:

```
document.forms[“nombre”].elements[“nombre”]
```


El operador `+`. Este operador es usado para sumar y para concatenar cadenas. Esto puede causar problemas en algunas situaciones.

Para forzar a Javascript que los valores se tienen que sumar, podemos utilizar varias estrategias.

1. Si restamos 0 a los valores, entonces Javascript convertirá el valor a número.
Ejemplo: `total= (valor-0) + (valor1-0);`
2. Si le agregamos el operador `+` al principio de la variable, Javascript lo tratará como número. Ejemplo: `total = (+valor) + (+valor1);`

Eval. La función `eval()` es utilizada para interpretar código Javascript que este como cadena. Es mejor utilizar otro método para realizar la tarea antes de utilizar `eval`. Esta función puede abrir algunos agujeros de seguridad, si no se valida lo que se va a interpretar.

Objetos. En Javascript, como en cualquier otro lenguaje, es posible hacer cosas de diferente manera. La creación de objetos en Javascript no es la excepción. Las dos maneras son:

1. Utilizando la palabra reservada `new`: `variable= new object();`
2. Utilizando la notación JSON: `variable = { propiedad: "1", propiedad1: "2" }`

Es más rápido utilizar la notación JSON, ya que al utilizar `new`, convierte la variable de la misma manera solo que utilizando más tiempo y recursos.

7.2. Patrones de diseño

Un patrón de diseño es una solución probada que se puede aplicar con éxito a un determinado tipo de problema que aparecen repetidamente y bajo determinadas condiciones. Proveen un esqueleto básico que cada diseñador adapta a las características de la aplicación.

Un patrón de diseño se compone de un nombre, el cual referencia al problema y la solución. La descripción del problema y bajo qué condiciones puede aplicarse el patrón. La descripción de la solución sin tener una implementación en particular. Y las consecuencias de utilizarlo.

Los patrones de diseño pueden tener muchas clasificaciones, pero comúnmente son clasificados de acuerdo a su propósito: creacionales, estructurales y de comportamiento.

Los patrones de creación se refieren a como se puede crear un objeto. Los estructurales se enfocan a como están compuestas las clases y los objetos. Los de comportamiento se enfocan a como interactuarán los objetos entre si.

Existen una gran cantidad de patrones y cada uno es una solución a un problema específico. Particularmente en AJAX existen 3 patrones de diseño que pueden ser muy útiles para solucionar algunos problemas relacionados con las incompatibilidades de los exploradores y el manejo de eventos.

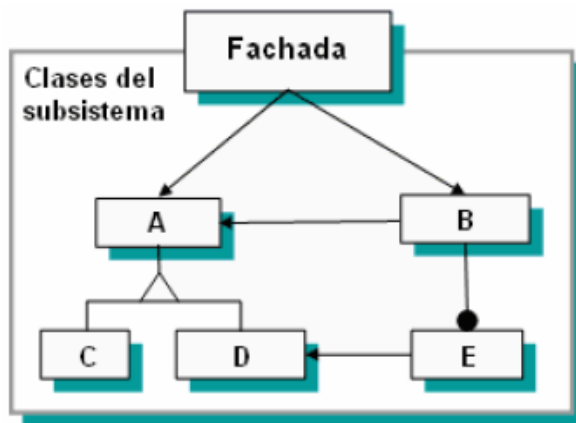
En la siguiente sección se resumen los puntos más importantes del patrón fachada, adaptador y observador

7.2.1. El patrón fachada

Es un patrón estructural que proporciona una interfaz común para un conjunto de interfaces en un subsistema, haciéndolo más fácil de usar.

En una aplicación AJAX es útil para aplicarlo en las incompatibilidades que existen con el objeto *XMLHttpRequest* y la manipulación de los elementos DOM.

Figura 13. Estructura del patrón Fachada



Fuente: Cesar Acuña. **Patrones de Diseño**. Página 48.

<http://kybele.escet.urjc.es/Documentos/ISI/Patrones%20de%20Diseño.pdf> . (22/02/2008)

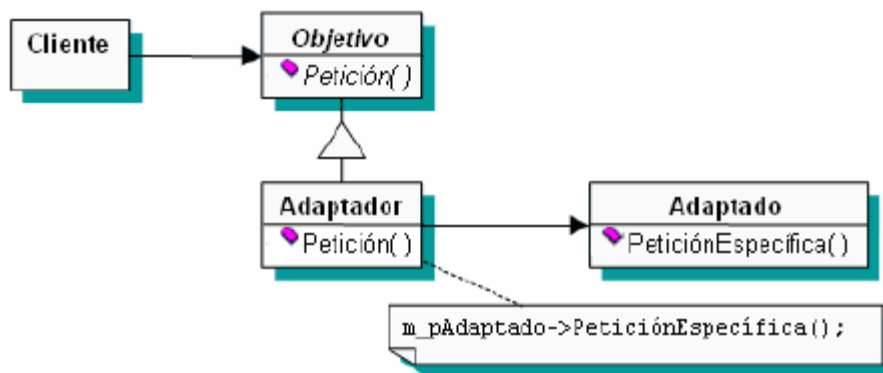
Algunos aspectos a tomar en cuenta al utilizar este patrón:

- Se pueden cambiar las clases del subsistema sin necesidad de cambiar los clientes. Solo hay que realizar los cambios en la fachada
- No evita que se puedan usar las clases del subsistema.
- Favorece el acoplamiento débil entre los clientes y el subsistema.

7.2.2. El patrón Adaptador

Es un patrón estructural que convierte la interfaz de una clase en otra esperada por el usuario. Permite que clases con interfaces incompatibles puedan comunicarse. Al igual que el patrón fachada, es común utilizarlo para tratar de resolver el problema de las diferentes implementaciones del objeto *XMLHttpRequest*.

Figura 14. Estructura del patrón Adaptador



Fuente: Cesar Acuña. **Patrones de Diseño**. Página 34.

<http://kybele.escet.urjc.es/Documentos/ISI/Patrones%20de%20Diseño.pdf>. (22/02/2008)

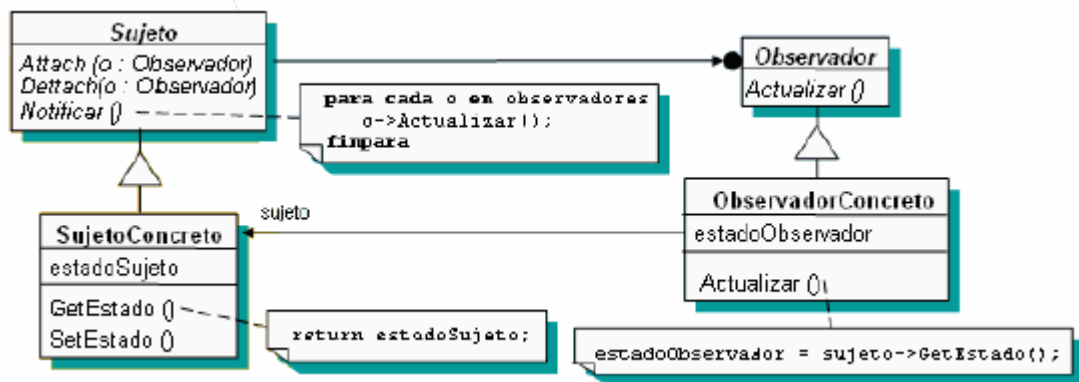
Algunos aspectos a tomar en cuenta al utilizar este patrón:

- No es posible heredar todas las sub clases de una clase
- El adaptador hereda el comportamiento de adaptado
- El adaptador puede funcionar con varios adaptados

7.2.3. El patrón observador

Es un patrón de comportamiento que define una dependencia de uno a muchos entre objetos. Cuando un objeto cambie su estado, los n objetos serán notificados y se actualizarán. Es aplicable en el sentido que Javascript no provee un mecanismo para gestionar múltiples eventos para un mismo elemento. Aunque existe el modelo de eventos de W3C que si lo permite, no está implementado y soportado en todos los exploradores de la misma manera.

Figura 15. Estructura del patrón Observador



Fuente: Cesar Acuña. **Patrones de Diseño**. Página 78.

<http://kybele.escet.urjc.es/Documentos/ISI/Patrones%20de%20Diseño.pdf>. (22/02/2008)

Algunos aspectos a tomar en cuenta al utilizar este patrón:

- El acoplamiento entre el sujeto y el observador es abstracto
- Un pequeño cambio en un observador puede significar grandes cambios en otros
- Actualización inesperada cuando no se detecta que ha cambiado.

7.2.4. Patrones de diseño AJAX

En los apartados anteriores se mencionan 3 patrones de diseño de uso general, aplicables a cualquier tipo de aplicación. En las aplicaciones AJAX también existen patrones de diseño que resuelven un problema específico.

Los patrones de diseño de AJAX tienen tres pasos (Scott, 2006):

- **Disparador.** Cada patrón empieza con un evento del usuario o un evento de un temporizador. Ejemplos de estos eventos son: el movimiento del ratón, tecla presionada, botón ratón presionado, etcétera.
- **Operación.** Después del evento se realizan cualquiera de las siguientes operaciones:
 - Consultar : Obtener información cuando se necesite
 - Persistir: Guardar información en tiempo real
 - Validar: Prevenir errores antes de que sucedan.
 - Invocar: Realizar acciones en el instante.
 - Notificar: Comunicar algo en el instante
- **Actualización.** Después de realizar el proceso se refleja un cambio visual cuando es necesario. Ejemplos de estos cambios visuales son: transición de expansión, transición de movimiento, indicador de estado, etcétera.

Figura 16. Esquema de un patrón de diseño AJAX



Fuente: Bill Scott. **Designing for AJAX.**

<http://billwscott.com/share/presentations/realworldnyc/DesigningForAJAX-RWA-NYC.pdf>. (22/02/2008)

7.3. Optimización de Código

Javascript no es precisamente un lenguaje que se ejecute rápido. Las aplicaciones AJAX utilizan bastante código Javascript entonces es importante tomar algunas consideraciones.

Cuando se trata de medir el tiempo que tarda la aplicación en realizar alguna tarea o bien el tiempo que tarda una función, podemos hacerlo de varias maneras. La primera opción consiste en adquirir alguna herramienta que tenga esta función. Venkman es un *plug-in* para los exploradores Mozilla que permite depurar el código Javascript de la aplicación Web. La segunda opción es programar en Javascript algunas funciones mediante el objeto *Date*, que sirvan para este propósito.

7.3.1. Optimizar los ciclos

Es una mala práctica que se incluya una función en la condición de un ciclo, si solo retorna el valor que permite seguir ejecutando el ciclo.

Ejemplo:

```
while (contador<valorMaximo()){
    // instrucciones
    total = total+1;
}

for (contador=0;contador<valorMaximo();contador++){
    // Instrucciones
    total = total+1;
}
```

El problema con este código es que cada vez que se repite el ciclo, se llama a la función una y otra vez. Si la función `valorMaximo()` ejecuta código complejo, entonces puede causar que el ciclo se tarde más tiempo de lo requerido. Es mejor obtener el valor de la función afuera del ciclo, y después utilizar el valor en la condición.

Ejemplo:

```
int_valorMaximo=valorMaximo();
while (contador<int_valorMaximo){
    // instrucciones
    total = total+1;
}
```

```
int_valorMaximo=valorMaximo();
for (contador=0;contador<= int_valorMaximo;contador++){
    // instrucciones
    total = total+1;
}
```

7.3.2. Agregar nodos DOM

Cuando se crea un nuevo nodo DOM y se le agrega al árbol del documento, este automáticamente se actualiza en la ventana del explorador. Esta actualización requiere de muchos cálculos internos, por lo que es una operación pesada para el explorador.

Es una buena práctica primero crear y ensamblar todos los nuevos nodos DOM, y luego agregar esa estructura al árbol del documento.

7.3.3. Minimizar la notación de punto

Cada vez que el intérprete de Javascript encuentra un punto, busca la variable hijo en la variable padre. Como toda búsqueda, esto requiere tiempo y recursos.

Ejemplo:

```
temperatura = automovil.motor.temperatura;  
velocidad = automovil.motor.velocidad;  
nivel_aceite = automovil.motor.nivel_aceite;  
nivel_agua = automovil.motor.nivel_agua;
```

Forma sugerida:

```
motor= automovil.motor;  
velocidad = motor.velocidad;  
nivel_aceite = motor.nivel_aceite;  
nivel_agua = motor.nivel_agua;
```

En el segundo caso se asigna a una variable el objeto motor de la variable automovil. El objetivo es tratar de minimizar la notación de punto.

7.4. Cuándo utilizar AJAX

AJAX puede ser aplicado a una gran variedad de situaciones. Sin embargo hay escenarios donde el enfoque tradicional es la mejor opción. Bosworth (2006b) recomienda algunos lugares donde usar AJAX.

- **Formularios interactivos.** Hay ocasiones que es necesario llenar un formulario con muchas opciones. En ciertas opciones requiere un cambio en el formulario. El ejemplo común es cuando se selecciona el país. La página se actualiza pidiendo los cambios al servidor, y luego aparece con los nuevos datos o con menos opciones. Esto se puede repetir mientras se llena el formulario, haciéndolo molesto y lento. Con el enfoque AJAX, los cambios son más rápidos pidiendo los datos necesarios y actualizando cuando se tenga disponible, permitiendo llenar otras partes del formulario.
- **Navegación por árboles jerárquicos profundos.** En este escenario es complicado y lento tener que actualizar un árbol jerárquico cada vez que el usuario despliega sus nodos hijos. Al utilizar AJAX la experiencia se mejora, porque el despliegue de las opciones es más rápido y los datos necesarios son pedidos al servidor en demanda.
- **Comunicación rápida usuario a usuario.** En aplicaciones *Web* que crean discusiones inmediatas basadas en HTML, es necesario refrescar la página para ver los nuevos comentarios. Utilizando AJAX ese problema es resuelto al actualizar automáticamente cuando un nuevo comentario ha llegado al servidor.
- **Manipulación y filtrado de datos.** Acciones como ordenar por fecha, nombre, cantidad, agregar y eliminar filtros. Son más interactivas y rápidas si se realizan en AJAX, porque en el enfoque tradicional cada una de estas acciones se traduce a petición de una página HTML que el servidor tiene que procesar y enviar.
- **Autocompletación.** Sugerir frases de texto comunes. Es útil cuando se llena un formulario o cuando se realiza una búsqueda.

7.5. Manejo de Errores

Una aplicación AJAX, como cualquier tipo de aplicación, no está libre de errores inesperados que pueden suceder de un momento a otro. Por ese motivo un buen manejo de esos errores hará la aplicación más robusta y confiable.

Javascript permite el manejo de los errores como Java y C# lo implementan. También el manejo de los errores no está restringido solo al *try/catch*, existe un evento genérico, que puede atrapar cualquier error en tiempo de ejecución.

Estructura general del *try/catch*.

```
function crear_tabla(){
    try{
        crearEstructura();
        agregarTabla();
    }catch (e) { // el objeto e tiene dos propiedades: name y message
        alert (“Ha ocurrido el error: “ + e.name + e.message”);
    }finally {

    }
}
```

Si se escapa algún error entonces podemos asignarle al evento *onerror* una función que se encargue del manejo del error.

```
// Agrega la referencia de la función encargada de manejar el error
window.onerror = manejar_errores;
function manejar_errores(mensaje, URL , line){
    // Código para gestionar los errores
    return true;
}
```

Un error a considerar es cuando una petición al servidor nunca regresa. Una solución consistiría en proveer un mecanismo que verificará el tiempo máximo de espera, antes de considerarla perdida. Otro error sería el caso en que la respuesta del servidor retorne con un código de error. En ambos casos sería oportuno informar al usuario, para que tome alguna acción al respecto.

8. EJEMPLO UTILIZANDO AJAX

8.1. Descripción

Actualmente, la opción de cursos asignados de la Universidad Virtual de la carrera de Ciencias y Sistemas de la Facultad de Ingeniería funciona de la siguiente manera:

- Cuando se selecciona esta opción primero aparecen los cursos asignados el semestre actual y luego una gran lista de todos los cursos que nos hemos asignado a lo largo de la carrera.
- Si queremos ver qué links disponibles tiene determinado curso, tenemos que hacerle un clic al curso para verlos.
- Si queremos buscar un curso tenemos que buscarlo entre la lista que aparece debajo de los cursos del semestre actual. A veces la búsqueda es un poco molesta porque no ofrece otros métodos para realizar una búsqueda de un curso.

La propuesta usando AJAX sería tener 3 opciones en forma de pestañas en forma de acordeón. La primera por defecto mostraría los cursos asignados actualmente. La segunda opción sería organizar los cursos por semestre y año. La tercera opción sería clasificarlos en orden alfabético. Se tendrá la opción de buscar un curso por el nombre del mismo. Esta opción funcionará al estilo Google Suggest. Se usa un *framework open source* para la parte de los controles gráficos y la comunicación con el servidor.

8.2. Solución

8.2.1. Requisitos

Sistema Operativo

La aplicación puede funcionar en cualquiera de estos sistemas operativos:

- Windows XP Profesional cualquier versión, Service Pack 2.
- Windows 2000 Server Service Pack 4.
- Windows 2003 Server cualquier versión, Service Pack 1.

Bases de Datos

La aplicación puede utilizar cualquiera de las siguientes bases de datos:

- SQL Server Express 2005.
- SQL Server 2000.
- SQL Server 7.

Componentes necesarios

- *Framework .Net* version 2.050727 en adelante.
- IIS (*Internet Information Server*).

8.2.2. Descripción

El ejemplo fue desarrollado utilizando Microsoft Visual Studio 2005 utilizando como bases de Datos a SQL Express 2005.

Los *frameworks* que se utilizaron son:

- **Magic AJAX.** *Framework open source* del lado del servidor para .Net que genera automáticamente todo el código Javascript necesario para agregar funcionalidad AJAX.

- **RICO.** *Framework open source* del lado del cliente, que provee librerías para la comunicación con el servidor y algunos componentes gráficos.
- **YUI.** *Framework open source* del lado del cliente, que provee librerías para la comunicación con el servidor y componentes gráficos como: árboles, ventanas, autocompletación y menús.

El ejemplo es un sitio *Web* que simula la Universidad Virtual de la Escuela de Ciencias y Sistemas. Al ser un ejemplo ilustrativo de la funcionalidad de AJAX, es una versión sencilla y resumida de la verdadera Universidad Virtual. No tiene todas las opciones y validaciones que esta tiene.

El ejemplo se compone de dos partes: La primera parte que son las opciones para el estudiante y la segunda parte que son las opciones para el catedrático o administrador.

8.2.3. Opciones del estudiante

Para acceder a cualquier opción dispone es necesario que el usuario se autentique. La siguiente imagen muestra la pantalla para autenticarse.

Figura 17. Pantalla de autenticación



The screenshot shows a web page for the 'Facultad de Ingeniería Escuela de Ciencias y Sistemas'. The page features a logo on the left and a central login form titled 'Ingreso al Sistema'. The form includes fields for 'Usuario' (containing '200112001') and 'Clave' (masked with dots), a checkbox for 'Recordar la proxima Vez.', and an 'Ingresar' button. Below the form is a link for '¿Desea Registrarse?' and a dropdown menu currently set to 'Estudiante'.

Al ingresar correctamente, el usuario tendrá disponibles las siguientes opciones:

- **Inicio.** Es la página principal para las opciones del usuario. Muestra el escudo de la universidad y del sitio.
- **Cursos Asignados.** En esta opción el usuario podrá buscar información relacionada a los cursos que se ha asignado. Podrá seleccionar entre varias opciones: cursos actuales, cursos por semestre y cursos ordenados en orden alfabético.
- **Asignar Cursos.** En esta opción el usuario podrá asignarse cursos así como desasignar cursos del presente ciclo.

Figura 18. Opciones para los estudiantes



8.2.3.1. Cursos asignados

En esta opción se podrá buscar de dos maneras: Por medio de árboles jerárquicos y por medio del nombre del curso.

La búsqueda por medio de árboles jerárquicos está dividida en tres partes: La primera por los cursos asignados actualmente, la segunda por cursos ordenados por semestre y la tercera por cursos ordenados alfabéticamente.

La búsqueda por medio del nombre del curso se encuentra en la parte superior de la página. En esta opción cuando se empieza a escribir el nombre, aparecen posibles sugerencias del nombre del curso que se desea buscar. Al presionar el botón de búsqueda, aparecen todas las secciones a las que se ha asignado el usuario.

Figura 19. Búsqueda por nombre curso

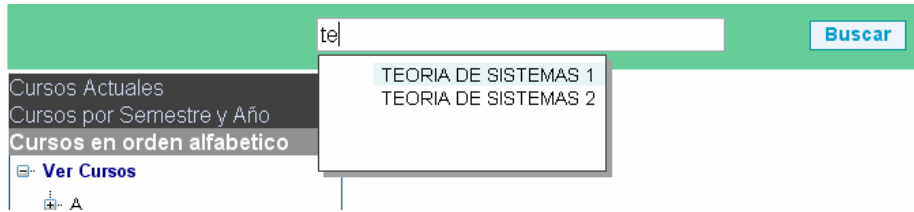
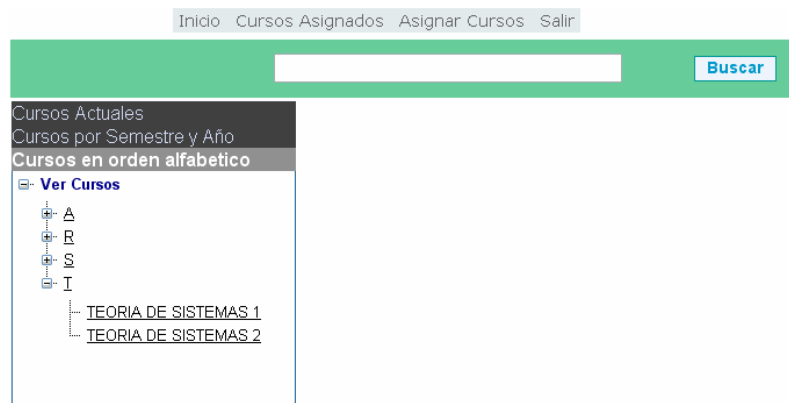
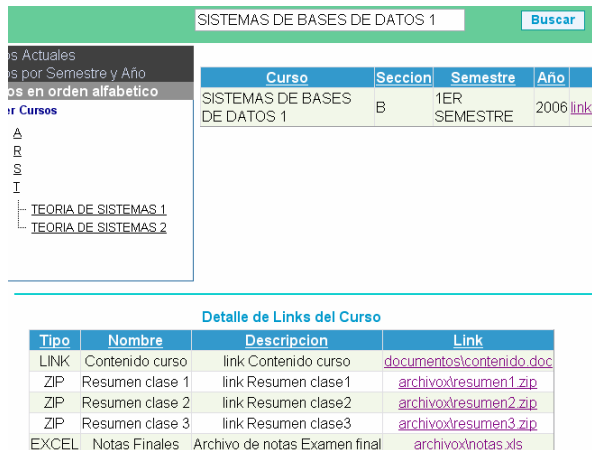


Figura 20. Búsqueda en árboles jerárquicos



Al seleccionar un curso ya sea por una búsqueda o por medio de los árboles jerárquicos, aparece al lado superior derecho la información relevante del curso. Si se selecciona los links de la sección entonces aparecen en la parte inferior de la pantalla.

Figura 21. Detalle de información de un curso



8.2.3.2. Asignación cursos

En esta opción el usuario podrá asignarse los cursos que necesita. Aparecerán los cursos que actualmente se impartirán en el presente ciclo.

Al seleccionar un curso aparecerá que catedrático la va a impartir y que horario tendrá la sección. También mostrara los cursos que ya se ha asignado en el presente ciclo. El usuario tendrá la opción de desasignarse un curso que previamente se halla asignado.

Figura 22. Asignación de cursos

Inicio Cursos Asignados Asignar Cursos Salir

Curso REDES DE COMPUTADORAS 2 **Seccion** A

Informacion del Curso

Hora Inicio	07:10
Hora Fin	08:50
Catedratico	Manuel Lopez

Asignar

Detalle de Cursos Asignados

Curso	Seccion	Inicio	Fin	Desasignar
SISTEMAS DE BASES DE DATOS 1	B	07:10	08:50	Desasignar
ANALISIS Y DISEÑO DE SISTEMAS 2	B	07:10	08:50	Desasignar
ANALISIS Y DISEÑO DE SISTEMAS 1	B	07:10	08:50	Desasignar

8.2.4. Opciones de administrador o catedrático

Para acceder a esta opción también es necesario autenticarse. Al ingresar correctamente el usuario tendrá disponibles las siguientes opciones:

- **Inicio.** Es la página principal para las opciones del usuario. Muestra el escudo de la universidad y del sitio.

- **Cursos.** En esta opción se podrán crear cursos, asignar cursos al pensum de una carrera y asignar un curso al ciclo actual.
- **Agregar sección.** En esta opción se podrán crear las secciones, para los cursos asignados al ciclo actual.
- **Crear links.** En esta opción se podrán crear e eliminar *links* para las secciones que estén creadas para el ciclo actual.
- **Nuevo ciclo.** En esta opción el usuario podrá crear un nuevo ciclo de actividades. Se muestran los ciclos ya creados en la parte superior derecha.

8.2.4.1. Cursos

Esta opción está dividida en tres partes: La primera es la creación de cursos, la segunda opción es la asignación de un curso a una carrera, y la tercera es la asignación de un curso al ciclo actual.

La primera opción Agregar Curso el usuario podrá crear un nuevo curso que mas adelante podrá asignar al pensum de una carrera.

Figura 23. Agregar cursos

Inicio Cursos Agregar Seccion Crear links

[Agregar Curso](#) [Curso Carrera](#) [Curso Ciclo](#)

Agregar Curso

Nombre Curso

Descripción Curso

La segunda opción Agregar Curso a Carrera el usuario podrá asignar un curso a una carrera. Así mismo se ingresa la cantidad de créditos para el curso.

Figura 24. Agregar curso a carrera

Inicio Cursos Agregar Seccion Crear links Nuevo

[Agregar Curso](#) [Curso Carrera](#) [Curso Ciclo](#)

Agregar Curso a Carrera

Carrera
Ingeniería en Ciencias y Sistemas ▼

Curso
INVESTIGACION DE OPERACIONES I ▼

Créditos
5

Agregar

La tercera opción Agregar Curso a Ciclo actual el usuario agrega un curso al ciclo actual de actividades. Estos cursos son los disponibles para crear una sección.

Figura 25. Agregar curso a ciclo actual

Inicio Cursos Agregar Seccion Crear links Nuevo

[Agregar Curso](#) [Curso Carrera](#) [Curso Ciclo](#)

Agregar Curso a Ciclo Actual

Carrera
Ingeniería en Ciencias y Sistemas ▼

Curso
MATEMATICA BASICA 1 ▼

Agregar

8.2.4.2. Agregar sección

En esta opción se podrá crear y eliminar una sección para un curso que este asignado al ciclo actual de actividades. Entre los datos a ingresar se encuentran el nombre de la sección y la hora de inicio y fin del mismo. También en la parte inferior se muestran las secciones que tiene un determinado curso.

Figura 26. Agregar sección a curso

Inicio Cursos Agregar Seccion Crear links Nueva

Carrera Ingeniería en Ciencias y Sistemas **Catedratico** No asignado

Curso MATEMATICA BASICA 1

Seccion C

Hora Inicio 11:10

Hora Fin 12:50

Detalle de las secciones del Curso

seccion	catedratico	Hora Inicio	Hora Fin	
A	No asignado	09:10	10:50	Eliminar
B	No asignado	07:10	08:50	Eliminar

8.2.4.3. Crear links

En esta opción se podrá crear y eliminar *links* para una sección del ciclo actual de actividades. En la parte inferior aparece el detalle de los *links* que tenga la sección.

Figura 27. Agregar links a sección

Carrera
 Ingeniería en Ciencias y Sistemas ▼

Curso
 SISTEMAS DE BASES DE DATOS 1 ▼

Sección **Tipo link**
 B ▼ EXCEL ▼

Nombre
 Notas Finales

Descripción
 Archivo de notas finales

Ruta Archivo
 notas\notas_finales.xls

Crear

Detalle de los Links del Curso

descripcion	url_path	tipo_link	
link Contenido curso	documentos\contenido.doc	LINK	Eliminar
link Resumen clase1	archivo\resumen1.zip	ZIP	Eliminar
link Resumen clase2	archivo\resumen2.zip	ZIP	Eliminar

8.2.4.4. Nuevo ciclo

En esta opción se podrá crear un ciclo al año actual y ver los ciclos de actividades anteriores.

Figura 28. Nuevo ciclo

Inicio Cursos Agregar Sección Crear links Nuevo Ciclo Salir

Agregar Nuevo Ciclo Al Año Actual

CICLO **CICLOS EXISTENTES**
 1ER SEMESTRE 1ER SEMESTRE 2008 ▼

Agregar

8.3. Beneficios de AJAX

Una de las razones por las que se utilizó *Magic* AJAX para realizar la aplicación *Web* es que permite desactivar la funcionalidad AJAX. Al desactivarlo, la aplicación *Web* se comporta de manera normal.

Para comprobar los beneficios de AJAX se realizan varias gestiones y se toman como indicadores la cantidad de bytes transmitidos y el tiempo necesario para completar la tarea. El tráfico generado es capturado utilizando la herramienta Fiddler, el cual es un depurador de HTTP que permite generar estadísticas relacionadas al tráfico capturado.

8.3.1. Gestiones de la opción catedrático

La finalidad de esta prueba es utilizar las opciones más comunes. Se procede a crear un nuevo ciclo, después se agregan cursos al ciclo, se crea la sección y de último se crea un link.

A continuación se detalla paso a paso las actividades:

- Autenticarse como catedrático
- Ir a la opción Nuevo Ciclo
 - Crear un nuevo ciclo
- Ir a la opción de cursos
 - Curso Ciclo
 - Agregar los cursos
 - Seleccionar la carrera de Ingeniería en Ciencias y Sistemas
 - Agregar los cursos de: Matemática Básica 1, Análisis y diseño de sistemas 1, Teoría de sistemas 1, Sistemas de bases de datos 1, Inteligencia artificial 1

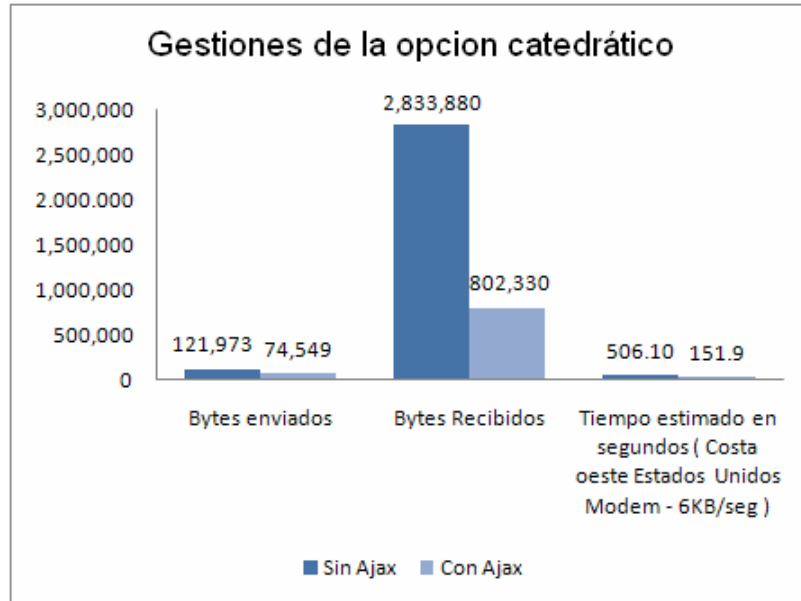
- Seleccionar la opción Agregar sección
 - Carrera: Ingeniería en Ciencias y Sistemas
 - Catedrático: No asignado
 - Agregar los cursos: Sistemas de base de datos 1, Matemática Básica 1, Teoría de sistemas 1.
 - Sección: A
 - Hora Inicio: 08:00
 - Hora fin: 08:45
- Seleccionar la opción de Crear *Links*
 - Carrera: Ingeniería en Ciencias y Sistemas
 - Curso: Sistemas de bases de datos 1
 - Sección: A
 - Tipo link: PDF
 - Nombre: Programa del curso
 - Descripción: Programa del curso
 - Ruta Archivo: /pdf/programa_curso1001.pdf
- Salir de la aplicación *Web*

En la tabla se resume el resultado de las pruebas. Con la funcionalidad de AJAX activada el rendimiento mejora notablemente. Disminuye la cantidad de bytes transmitidos así como el tiempo necesario para llevar a cabo la gestión

Tabla XXIV. Tabla comparativa de la opción catedrático

Indicadores	Sin AJAX	Con AJAX	Diferencia	%
<i>Bytes</i> enviados	121,973	74,549	47,424	38.88%
<i>Bytes</i> Recibidos	2,833,880	802,330	2,031,550	71.69%
Tiempo estimado en segundos (Coste oeste Estados Unidos Modem - 6KB/seg)	506.10 (8.43 minutos)	151.90 (2.53 minutos)	354.20 (5.90 minutos)	69.99%

Figura 29. Gestiones de la opción catedrático



8.3.2. Gestiones de la opción estudiante

Para la prueba de la opción de estudiante se realizan los siguientes pasos. A continuación se detalla paso a paso las actividades:

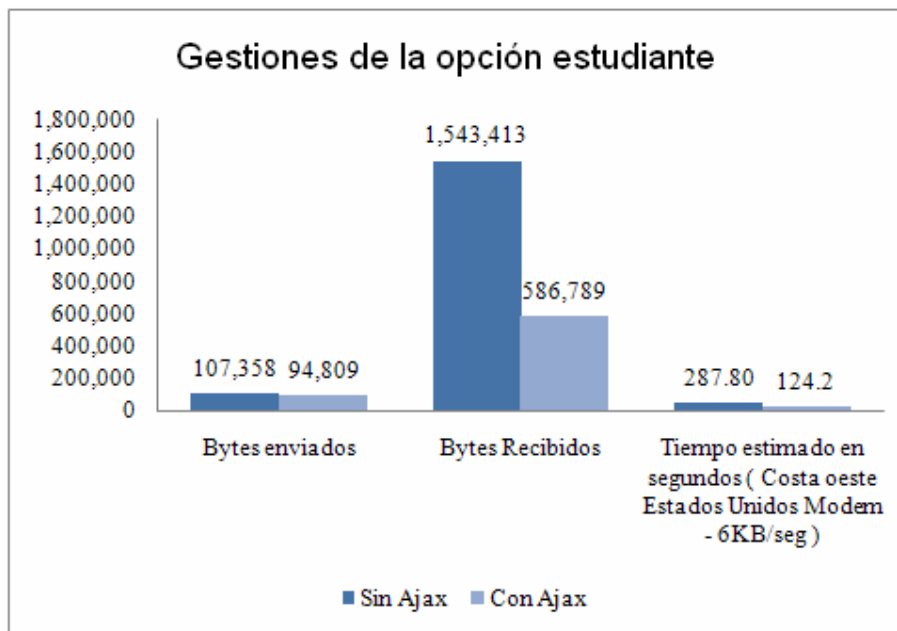
- Autenticarse como estudiante
- Seleccionar la opción Asignar Cursos
 - Asignarse los cursos: Sistemas de base de datos 1, Matemática Básica 1, Teoría de sistemas 1
- Seleccionar la opción Cursos asignados
 - Ver los cursos actuales
 - Ver los *links* del curso Sistemas de base de datos1
 - Ver los cursos por orden alfabético
 - Ver los cursos que empiecen con R
- Salir de la aplicación *Web*

Igual que la prueba anterior, la opción de estudiante mejoró en rendimiento en un 61% al tener activada la funcionalidad AJAX. En la tabla se resume el resultado de las pruebas.

Tabla XXV. Tabla comparativa de la opción estudiante

Indicadores	Sin AJAX	Con AJAX	Diferencia	%
Bytes enviados	107,358	94,809	12,549	11.69%
Bytes Recibidos	1,543,413	586,789	956,624	61.98%
Tiempo estimado en segundos (Costa oeste Estados Unidos Modem - 6KB/seg)	287.80 (4.79 minutos)	124.20 (2.07 minutos)	163.60 (2.72 minutos)	56.85%

Figura 30. Gestiones de la opción estudiante



CONCLUSIONES

1. AJAX es una técnica que ayuda a mejorar ciertos aspectos de una aplicación *Web*. La parte medular, el objeto *XMLHttpRequest* hace posible que el explorador se comunique asíncronamente con el servidor *Web* permitiendo que el explorador no tenga que esperar por la respuesta y así no interrumpir el flujo de trabajo del usuario. Entre los aspectos más importantes se encuentran: el rendimiento, la interactividad y el menor consumo de ancho de banda.
2. La diferencia fundamental entre el modelo tradicional de una aplicación *Web* y una basado en AJAX es el motor AJAX que actúa como intermediario entre el explorador y el servidor *Web*. El funcionamiento general del motor AJAX se resume en: comunicaron asíncrona con el servidor *Web*, pequeños cambios en la interfaz del usuario y la obtención de datos al servidor *Web* cuando sea necesario.
3. Las tecnologías que forman AJAX son estándares *Web* a excepción del objeto *XMLHttpRequest* que aun se encuentra en etapa de revisión. Esto permite cierta uniformidad de las aplicaciones AJAX. Los problemas relacionados con los exploradores se deben en su mayoría el no seguir completamente las especificaciones de dichos estándares.
4. Cada tecnología detrás de AJAX, tiene una función específica, esto ayuda en ciertos aspectos como:
 - a. Javascript. Lenguaje de programación *script* soportado en la mayoría de exploradores modernos.

- b. CSS. Permite separar el contenido de la presentación. Las hojas de estilo definen la apariencia de los elementos visuales de la página HTML.
 - c. DOM. API independiente del lenguaje que provee una estructura lógica parecida a un árbol, para modificar los elementos y contenido de los documentos XML y HTML.
 - d. XML. Estándar para el intercambio de información. Es un lenguaje independiente de la plataforma, que permite definir otros lenguajes o estructuras de datos.
 - e. El objeto *XMLHttpRequest*. Permite la comunicación asíncrona con el servidor. Soporta una gran variedad de formatos de intercambio de datos, por lo que es muy flexible, además de tener soporte en la mayoría de las nuevas versiones de los exploradores del mercado.
5. Visto desde un nivel general, la arquitectura AJAX difiere de la arquitectura *Web* tradicional, con el uso de un motor AJAX encargado de gestionar los eventos de la aplicación y de la comunicación asíncrona con el servidor. Este motor AJAX es un intermediario entre el servidor *Web* y el explorador.
6. Toda la funcionalidad que ofrece AJAX es proporcionada también por otras tecnologías como Java y Flash. La ventaja que tiene AJAX con los demás es que no necesita ningún *plug-in* en el lado del explorador, solo necesita tener activado Javascript y que el explorador soporte el objeto *XMLHttpRequest*.
7. Seleccionar la mejor tecnología para una aplicación *Web*, no es una tarea fácil. Cada tecnología Java, Flash y AJAX tienen características únicas que los hacen la mejor opción dependiendo del tipo y requerimientos de la aplicación.

8. La utilidad de los *frameworks* de AJAX para los lenguajes de programación del lado del servidor (.NET, Java y PHP) ayudan a reducir el tiempo de desarrollo. La elección del mejor *framework* depende de las habilidades del equipo de desarrollo, la documentación, el grado de configuración y los recursos asignados al proyecto.
9. El uso de *frameworks* de lado del cliente ayudan a disminuir el tiempo de desarrollo de cierta manera porque proveen abstracciones de alto nivel y componentes listos para usar. La facilidad de uso y documentación juegan un papel importante para lograr este objetivo. También se encargan de las incompatibilidades de los exploradores y problemas relacionados al utilizar una sola página Web (*bookmarks* y el botón de atrás).
10. La seguridad en las aplicaciones *Web* es muy importante. AJAX no es la excepción. Tomar medidas de seguridad mínimas como la validación de entradas pueden prevenir ataques de inyecciones SQL y secuencias de comandos entre sitios que pueden ser muy desastroso para la organización.
11. Entre las ventajas más importantes al utilizar AJAX se encuentran: el rendimiento, las interfaces de usuario, la interactividad y la portabilidad. Cada una de estas ventajas se verá en una proporción distinta dependiendo del diseño de la aplicación y del tipo de aplicación.
12. La accesibilidad y la usabilidad son dos desventajas derivadas del uso obligatorio de un lenguaje *script* y de la pérdida de cierta funcionalidad del explorador como lo es el botón de atrás y el uso de *bookmarks*. También la falta de herramientas adecuadas como editores y depuradores para el código Javascript pueden afectar el tiempo de desarrollo y la calidad de la aplicación.

13. El uso de mejores prácticas en el desarrollo de una aplicación AJAX ayuda a resolver ciertas cuestiones como:
 - a. Ayuda a resolver algunos problemas, que alguien ya analizó y presenta una posible solución
 - b. El uso de patrones de diseño ayudan a resolver algunos problemas relacionados con los exploradores.
 - c. El uso de patrones AJAX ayudan al desarrollo de la aplicación, porque proveen una guía de cómo resolver un problema y las posibles consecuencias de mismo.
 - d. Javascript al ser un lenguaje interpretado es lento, entonces al aplicar técnicas de optimización del código, hacen que la aplicación sea más rápida brindando una mejor experiencia.
 - e. El manejo de errores en una aplicación AJAX es muy importante, ya que si falla puede ocasionar perdida de información o alteración involuntaria de datos.

14. El objetivo principal del ejemplo presentado en este trabajo es mostrar la funcionalidad que puede tener una aplicación AJAX y mostrar algunas de sus ventajas. El resultado de la comparación entre una aplicación *Web* AJAX y sin AJAX se mostró en el ejemplo. Los resultados fueron satisfactorios ya que disminuyó la cantidad de bytes transmitidos desde el servidor al explorador del cliente, la cantidad de tiempo necesario para completar las tareas disminuyó y el uso de la aplicación *Web* fue más fluida.

RECOMENDACIONES

1. XML es el formato de intercambio de información estándar para la mayoría de aplicaciones *Web*. En el caso de AJAX, XML no es la única opción, JSON es otro formato para el intercambio de información que es mucho más ligero que XML. La elección de la mejor opción dependerá del tipo y requerimientos de la aplicación
2. El objeto *XMLHttpRequest* no está atado exclusivamente a un formato de *intercambio* como el XML, la elección del formato de intercambio depende en gran medida del *tipo* de aplicación. Entre los formatos más conocidos se encuentran: XML, JSON, HTML y texto plano.
3. El objeto *XMLHttpRequest* es un objeto nativo del explorador. El uso de este objeto no está restringido a un lenguaje *script* como Javascript. Cualquier lenguaje *script* que soporte el explorador podrá hacer uso de este objeto. Por ejemplo al utilizar VBscript la aplicación AJAX solo correrá en los exploradores de Microsoft Internet Explorer u otro que soporte este lenguaje *script*.
4. Al momento de elegir AJAX considerar que Flash y Java es más fácil mantener al código. Una de las razones es que ambos son orientados a objetos y proveen herramientas para depurar el código. En el caso de Javascript no es un lenguaje orientado a objetos y las herramientas para depurar el código aún no son lo suficientemente adecuadas.

5. Al momento de elegir una determinada tecnología como Java Applets, Flash o AJAX, es necesario conocer cuáles son las ventajas y desventajas de cada una de estas tecnologías. La opción más adecuada será aquella en la cual los requerimientos de la aplicación serán cubiertos de la forma más adecuada y de la forma más simple.
6. El uso de ciertos *frameworks* del lado del servidor permiten generar automáticamente todo el código Javascript necesario para la aplicación. Esto ayuda en cierta medida a reducir el tiempo de desarrollo, pero también genera grandes cantidades de código Javascript que aumentara el tamaño final de la aplicación.
7. Es necesario testear rigurosamente las aplicaciones AJAX, porque existen una gran variedad de exploradores y alguna funcionalidad puede no funcionar de igual manera o no funcionar en absoluto.
8. Es muy importante la validación de todo tipo de entrada de datos que tenga la aplicación AJAX, porque es la puerta para cualquier ataque. Es necesario que la validación de datos sea en la parte del cliente y en la parte del servidor, ya que la validación que se haga en el lado del cliente puede ser burlada más fácilmente que del servidor.
9. El uso del modelo MVC en el diseño de una aplicación AJAX, es de suma utilidad, porque permite separar la aplicación en tres sistemas, que pueden desarrollarse con cierta independencia. En este contexto ayuda a definir con más claridad, la lógica de presentación y parte de la lógica del negocio que tendrá la aplicación AJAX en el explorador.

10. Tratar de mantener la cantidad de código AJAX lo mínimo posible, con el objetivo reducir el tiempo requerido para la descarga.
11. Utilizar AJAX como valor agregado solo si los beneficios son considerables y si ayuda al flujo de trabajo de la aplicación Web.

REFERENCIAS

- Bosworth, Alex. (2005a), **AJAX Mistakes**.
<http://alexbosworth.backpackit.com/pub/67688>. (15/02/2008)
- Bosworth, Alex. (2006b). **Places To Use AJAX**.
<http://swik.net/AJAX/Places+To+Use+AJAX>. (15/02/2008)
- CERT. (2000), **Understanding Malicious Content Mitigation for Web Developers**. http://www.cert.org/tech_tips/malicious_code_mitigation.html. (20/02/2008)
- Crane Save, Eric Pascarello y Darren James. (2006), **AJAX in Action**. Manning Publications, 650pp, 2006.
- Finnigan, Pete. (2006), **SQL Injection and Oracle, Part One**.
<http://www.securityfocus.com/infocus/1644>. (20/02/2008)
- Frye, Colleen. (2006a), **AJAX alert raises security, scalability issues**.
http://searchsoa.techtarget.com/news/article/0,289142,sid26_gci1162641,00.html. (05/03/2008)
- Frye, Colleen. (2006b), **Eric Pascarello dissects AJAX security vulnerabilities**.
http://searchsoa.techtarget.com/news/interview/0,289202,sid26_gci1164745,00.html. (05/03/2008)
- Garcia, Fernando. (2005), **AJAX, un cóctel de programas, revoluciona el diseño y la navegación por las 'webs'**.
http://www.elpais.es/articulo.html?xref=20051208elpcbpor_1&type=Tes&anchor=elpcbpor. (15/02/2008)

- Garret, Jesse James. (2005), **AJAX: A New Approach to Web Applications**. <http://adaptivepath.com/publications/essays/archives/000385.php>. (08/03/2008)
- Govella, Austin. (2005), **Best Practices: Implementing Javascript for rich internet applications**. <http://thinkingandmaking.com/entries/63>. (20/02/2008)
- Hakman, kevin. (2006), **The Four "Quantum States" of AJAX**. <http://www.AJAXdevelopersjournal.com/read/166503.htm>. (08/03/2008)
- Scott, Bill W. (2006). **Designing for AJAX**. <http://billwscott.com/share/presentations/realworldnyc/DesigningForAJAX-RWA-NYC.pdf>. (22/02/2008)

BIBLIOGRAFÍA

1. Crane dave, Eric Pascarello y Darren James. **AJAX in Action**. Manning Publications, 2006. 650pp.
2. Asleson Ryan, Schutta Nathaniel T. **Foundations of AJAX**. Apress, 2006. 273pp.
3. Zakas Nicholas C, McPeak Jeremy, Fawcett Joe. **Profesional AJAX**. Wiley Publishing, 2006. 380pp.
4. **3 Practical Uses For AJAX**. <http://www.interspire.com/content/articles/33/1/3-Practical-Uses-For-AJAX>. (02/02/2008)
5. **AJAX (programming)**. [http://en.wikipedia.org/wiki/AJAX_\(programming\)](http://en.wikipedia.org/wiki/AJAX_(programming)). (19/04/2006)
6. **AJAX alert raises security, scalability issues**. http://searchwebservices.techtarget.com/originalContent/0,289142,sid26_gci1162641,00.html. (10/01/2008)
7. **AJAX and Flash Compared**. <http://getahead.ltd.uk/AJAX/AJAX-flash-compared>. (20/01/2008)
8. **AJAX Examples**. http://AJAXpatterns.org/AJAX_Examples. (02/02/2008)
9. **AJAX Gotchas**. http://AJAXpatterns.org/AJAX_Gotchas. (02/02/2008)
10. **AJAX Isn't All Purpose Soap**. <http://www.webpronews.com/expertarticles/expertarticles/wpn-62-20060322AJAXIsntAllPurposeSoap.html>. (12/02/2008)
11. **AJAX o el fin del clic y espera**. http://www.alzado.org/articulo.php?id_art=457. (10/01/2008)
12. **AJAX Patterns: Design Patterns for AJAX Usability**. <http://softwareas.com/AJAX-patterns>. (10/01/2008)
13. **AJAX Principles**. http://AJAXpatterns.org/AJAX_Principles. (02/02/2008)

14. **AJAX reconsidered.** <http://adambosworth.net/2005/06/01/AJAX-reconsidered/>. (02/02/2008)
15. **AJAX vai desifentar o Flash?.** http://www.usabilidoido.com.br/AJAX_vai_desifentar_o_flash.html. (15/01/2008)
16. **AJAX y Usabilidad.** <http://usalo.blogspot.com/2005/07/AJAX-y-usabilidad.html>. (20/01/2008)
17. **AJAX, AJAX Everywhere.** <http://www.powazek.com/2005/05/000520.html>. (20/01/2008)
18. **AJAX, promise or hype?.** http://www.quirksmode.org/blog/archives/2005/03/AJAX_promise_or.html. (12/02/2008)
19. **AJAX, un cóctel de programas, revoluciona el diseño y la navegación por las 'webs'.** http://www.elpais.es/articulo.html?xref=20051208elpcibpor_1&type=Tes&anchor=elpcibpor. (10/01/2008)
20. **AJAX, XMLHttpRequest or AHAX.** <http://www.baekdal.com/articles/Technology/AJAX-xmlhttprequest-ahax/>. (10/01/2008)
21. **AJAX.** <http://es.wikipedia.org/wiki/AJAX>. (10/01/2008)
22. **AJAX.** <http://tecncliente.osmosislatina.com/curso/AJAX.htm>. (10/01/2008)
23. **AJAX: A New Approach to Web Applications.** <http://adaptivepath.com/publications/essays/archives/000385.php>. (10/01/2008)
24. **AJAX: Asynchronous Java + XML?.** <http://www.developer.com/design/article.php/3526681>. (05/01/2008)
25. **AJAX: Descubre qué se oculta tras esta vieja-nueva tecnología.** <http://www.mouse.cl/2005/rep/04/15/index.asp>. (15/02/2008)
26. **AJAX: Single-page vs. Multi-page.** <http://getahead.ltd.uk/AJAX/single-page-design>. (15/01/2008)
27. **AJAX: Usable Interactivity with Remote Scripting.** <http://www.sitepoint.com/article/remote-scripting-AJAX>. (20/02/2008)

28. **Best Practices for AJAX development.**
http://www.jonathanboutelle.com/mt/archives/2005/07/best_practices.html.
(01/03/2008)
29. **Best Practices: Implementing javascript for rich internet applications.**
<http://thinkingandmaking.com/entries/63>. 01/03/2008
30. **Cómo: Evitar secuencias de comandos entre sitios en ASP.NET.**
<http://www.microsoft.com/spanish/msdn/articulos/archivo/201205/voices/paght00004.msp.x>. (15/02/2008)
31. **Cómo: Proteger ASP.NET de inyecciones SQL.**
<http://www.microsoft.com/spanish/msdn/articulos/archivo/201205/voices/paght00002.msp.x>. (15/02/2008)
32. **Cross-Domain AJAX. Security Implications in Depth.**
<http://getahead.ltd.uk/AJAX/cross-domain-xhr>. (20/02/2008)
33. **CSS2 Reference.** http://www.w3schools.com/css/css_reference.asp. (12/02/2008)
34. **DHTML en Castellano.**
<http://www.programacion.net/html/dinamico/tutorial/indice.htm>. (12/02/2008)
35. **Document Object Model (DOM) Level 1 Specification.**
<http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/DOM.pdf>.
(20/01/2008)
36. **DotNet AJAX Frameworks.**
http://AJAXpatterns.org/DotNet_AJAX_Frameworks. (15/02/2008)
37. **Dynamic Content with DOM-2.**
<http://developer.apple.com/internet/webcontent/dom2i.html>. (20/02/2008)
38. **Dynamic Html and Xml: The XMLHttpRequest Object.**
<http://developer.apple.com/internet/webcontent/xmlhttpreq.html>. (20/02/2008)
39. **ECMAScript Language Specification.** <http://www.ecma-international.org/publications/files/ecma-st/ECMA-262.pdf>. (20/01/2008)
40. **El Modelo Cliente/Servidor.**
<http://agamenon.uniandes.edu.co/~revista/articulos/cliser.html>. (01/03/2008)

41. **Eric Pascarello dissects AJAX security vulnerabilities.**
http://searchwebservices.techtarget.com/qna/0,289202,sid26_gci1164745,00.html.
(05/03/2008)
42. **Eric Pascarello interviewed about AJAX security vulnerabilities.**
http://AJAX.phpmagazine.net/2006/02/eric_pascarello_interviewed_ab.html.
(20/02/2008)
43. **Errors and AJAX.** <http://www.xml.com/pub/a/2005/05/11/AJAX-error.html>.
(12/02/2008)
44. **Fixing AJAX: XMLHttpRequest Considered Harmful.**
<http://www.xml.com/pub/a/2005/11/09/fixing-AJAX-xmlhttprequest-considered-harmful.html> (02/02/2008)
45. **Flash RIAs vs. Javascript RIAs.**
http://www.jonathanboutelle.com/mt/archives/2005/03/flash_rias_vs_j.html.
(12/02/2008)
46. **How to make XMLHttpRequest calls to another server in your domain.**
<http://fettig.net/weblog/2005/11/28/how-to-make-xmlhttprequest-connections-to-another-server-in-your-domain/>. (12/02/2008)
47. **How To: Protect From Injection Attacks in ASP.NET.**
<http://msdn.microsoft.com/security/default.aspx?pull=/library/en-us/dnpag2/html/paght000003.asp>. (20/02/2008)
48. **Introducing JSON.** <http://www.json.org/>. (20/01/2008)
49. **Java AJAX Frameworks.** http://AJAXpatterns.org/Java_AJAX_Frameworks.
(20/02/2008)
50. **Java Experts Predict AJAX Will Be Huge.**
<http://www.eweek.com/c/a/Application-Development/Java-Experts-Predict-AJAX-Will-Be-Huge/>. (24/03/2006)
51. **Javascript Best Practices.** <http://www.javascripttoolbox.com/bestpractices/>.
(02/02/2008)
52. **JavaScript Multipurpose frameworks.**
http://AJAXpatterns.org/Javascript_Multipurpose_Frameworks. (20/02/2008)

53. **JavaScript: The World's Most Misunderstood Programming Language.**
<http://www.crockford.com/javascript/javascript.html>. (10/01/2008)
54. **JSON como alternativa al XML en AJAX.**
<http://www.obispot.com/2006/01/31/json-como-alternativa-al-xml-en-AJAX/>.
(02/02/2008)
55. **KickStart Tutorial XML version 1.0.**
http://www.spiderpro.com/ebooks/kickstart_tutorial_xml.pdf. (10/01/2008)
56. **Lista de Aplicaciones AJAX.** <http://www.uberbin.net/archivos/web20/lista-de-aplicaciones-AJAX.php>. (22/03/2006)
57. **Listas enlazadas con AJAX y Json.** <http://www.obispot.com/labs/codigo/listas-AJAX-json/>. (20/02/2008)
58. **Meet AJAX: Intelligent Web Applications with AJAX.** <http://AJAX.sys-con.com/read/131768.htm>. (12/02/2008)
59. **Model-View-Controller.** <http://java.sun.com/blueprints/patterns/MVC-detailed.html>. (01/03/2008)
60. **Paper: HTML Code Injection and Cross-site scripting.**
<http://www.technicalinfo.net/papers/CSS.html>. (01/03/2008)
61. **Patrones de Diseño.** <http://siul02.si.ehu.es/~alfredo/iso/06Patrones.pdf>.
(15/02/2008)
62. **PHP AJAX frameworks.** http://AJAXpatterns.org/PHP_AJAX_Frameworks.
(20/02/2008)
63. **Que Cosa Es AJAX Y Para Qué Es Bueno.**
http://www.masternewmedia.org/es/tecnolog%C3%ADas_dise%C3%B1o_de_interfaz/AJAX/Que_Cosa_Es_AJAX_Y_Para_Qu%C3%A9_Es_Bueno_20051101.htm. (04/03/2006)
64. **Realtime Form Validation Using AJAX.**
<https://bpcatalog.dev.java.net/nonav/AJAX/validation/frames.html>. (15/02/2008)
65. **Request Validation - Preventing Script Attacks.**
<http://www.asp.net/faq/requestvalidation.aspx?tabindex=0&tabid=1>. (12/02/2008)
66. **Rich Internet Applications and AJAX - Selecting the best product.**
<http://www.javalobby.org/articles/AJAX-ria-overview/>. (10/01/2008)

67. **The Four "Quantum States" of AJAX.** <http://www.AJAXdevelopersjournal.com/read/166503.htm>. (08/03/2008)
68. **The XMLHttpRequest Object.** <http://www.w3.org/TR/XMLHttpRequest/>. (20/01/2008)
69. **Tuning AJAX.** <http://www.xml.com/pub/a/2005/11/30/tuning-AJAX-performance.html>. (01/03/2008)
70. **Understanding Malicious Content Mitigation for Web Developers.** http://www.cert.org/tech_tips/malicious_code_mitigation.html. (15/02/2008)
71. **Usable XMLHttpRequest in Practice.** <http://www.baekdal.com/articles/Usability/usable-XMLHttpRequest/>. (10/01/2008)
72. **Using AJAX in Web Applications.** http://www.telerik.com/documents/Telerik_and_AJAX.pdf. (12/02/2008)
73. **Using JavaServer Faces Technology with AJAX.** <https://bpcatalog.dev.java.net/nonav/AJAX/jsf-AJAX/frames.html>. (15/02/2008)
74. **Very Dynamic Web Interfaces.** <http://www.xml.com/pub/a/2005/02/09/xml-http-request.html>. (20/02/2008)
75. **Web Application Solutions: A Designer's Guide.** <http://www.lukew.com/resources/WebApplicationSolutions.pdf>. (15/02/2008)
76. **What kind of language is XSLT?.** <http://www-128.ibm.com/developerworks/xml/library/x-xslt/?article=xr>. (10/01/2008)
77. **Why AJAX Patterns.** http://AJAXpatterns.org/Why_AJAX_Patterns. (01/03/2008)
78. **Why use XUL?.** <http://www.xulplanet.com/tutorials/whyxul.html>. (01/03/2008)
79. **XAML.** <http://www.xaml.net/dictionary.html>. (01/03/2008)
80. **XAML Overview.** <http://msdn2.microsoft.com/en-us/library/ms752059.aspx>. (01/03/2008)

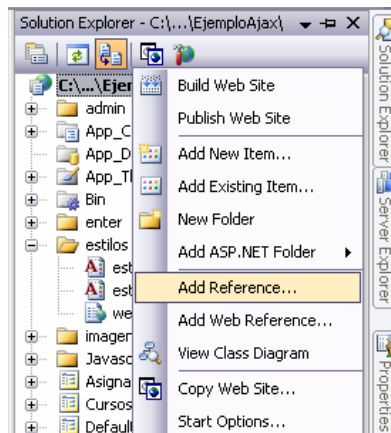
81. **Xml Transformations with Css and Dom.**
<http://developer.apple.com/internet/webcontent/xmltransformations.html>.
(10/01/2008)
82. **XMLHttpRequest Usability Guidelines.**
<http://www.baekdal.com/articles/Usability/XMLHttpRequest-guidelines/>.
(20/02/2008)
83. **XUL.** <http://developer.mozilla.org/en/docs/XUL>. (01/03/2008)

APÉNDICES

Apéndice 1: Pasos para agregar *Magic AJAX* al proyecto

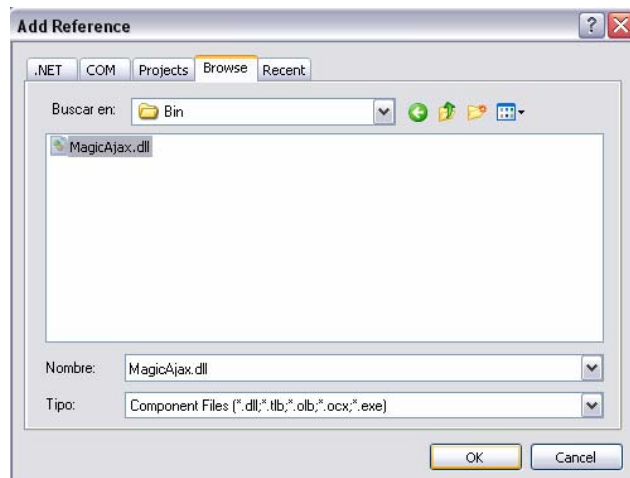
Primero crear el proyecto en Visual Estudio. Al tener creado el proyecto, se agrega la referencia a la librería. Para agregarle la referencia presionar el botón derecho en el explorador de soluciones en el nombre del proyecto. Luego seleccionar agregar referencia (add reference).

Figura 31. Menú agregar referencia en Visual Studio 2005



Luego de seleccionar la opción de agregar referencia. Seleccionar la pestaña de explorar (*Browse*). Al estar en esta opción se busca el archivo dll y se presiona el botón *Ok*.

Figura 32. Agregar la referencia en Visual Studio 2005



Luego de agregar la referencia es necesario modificar el archivo web.config. Agregar en la sección de “configSections” la siguiente línea:

```
<configSections>
<section name="magicAJAX"
type="MagicAJAX.Configuration.MagicAJAXSectionHandler, MagicAJAX"/>
</configSections>
<magicAJAX outputCompareMode="HashCode" tracing="false">
<pageStore mode="NoStore" unloadStoredPage="false" cacheTimeout="5"
maxConcurrentPages="5" maxPagesLimitAlert="false"/>
</magicAJAX>
```

En la sección de “system.web” agregar las siguientes líneas

```
<httpModules>
<add name="MagicAJAXModule" type="MagicAJAX.MagicAJAXModule,
MagicAJAX"/> </httpModules>
```

Para utilizar *Magic AJAX* en una determinada pagina agregar la siguiente línea al principio de la página.

```
<% @ Register Assembly="MagicAJAX" Namespace="MagicAJAX.UI.Controls"
TagPrefix="AJAX" %>
```

Ahora ya esta lista la configuración para poder utilizar *Magic AJAX* en el sitio *Web*. Para utilizar *Magic AJAX* solo se define un panel y dentro de este se agregan los controles que tendrán la funcionalidad AJAX. Ejemplo:

```
<AJAX:AJAXpanel id="AJAXPanel1" runat="server">
<asp:DropDownList id="dwl_carrera" runat="server" AutoPostBack="True"
DataSourceID="SqlID_carrera" DataTextField="nombre" DataValueField="carrera"
CssClass="estilo_comboBox1"></asp:DropDownList>
</AJAX:AJAXpanel>
```

Apéndice 2: Pasos para agregar RICO a una página *Web*.

Primero agregar las referencias de la librería en la página *Web*. Agregar las siguientes líneas de código.

```
<script type='text/javascript' src="Javascript/rico/prototype.js"></script>
<script type='text/javascript' src="Javascript/rico/rico.js"></script>
```

Dentro de la página *Web* se agrega el siguiente código HTML para definir un control con pestañas verticales. Para definir más opciones se agregan más opciones de la misma manera que la primera opción.

```

<div id='Acordion_cursos'> <!-- contenedor -->
  <div>                                <!-- primera opción -->
    <div>Cursos Actuales<br /></div>
    <div>contenido <div>
  </div>
</div>

```

Agregar el siguiente código Javascript en la página *Web* para crear el control gráfico.

```

<script type='text/javascript'>
window.onload=function(){
  var outer=$('#Acordion_cursos');
  outer.style.width='245px';
  new Rico.Accordion(
    outer,
    { panelHeight:200,
      expandedBg:'#909090',
      collapsedBg:'#404040',
      onShowTab: onShowTab
    }
  );
}
</script>

```

Página oficial de RICO:

Rico. <http://openrico.org/>. (01/03/2008)

Apéndice 3: Pasos para agregar YUI a una página Web.

Primero agregar las referencias de la librería en la página Web. Agregar las siguientes líneas de código.

```
<script src="Javascript/yui/build/yahoo/yahoo.js"></script>
<script src="Javascript/yui/build/dom/dom.js"></script>
<script src="Javascript/yui/build/event/event.js"></script>
<script src="Javascript/yui/build/animation/animation.js"></script>
<script src="Javascript/yui/build/autocomplete/autocomplete.js"></script>
```

Dependiendo de lo que se va a utilizar del *framework* el número de referencias puede variar. En este caso solo se incluyen las requeridas para utilizar el control de autocompletación.

Se agrega el siguiente código HTML para definir el control base para la autocompletación.

```
<input id="cursos_auto" type="text" style="width: 294px">
<div id="sombra_cursos">
<div id="contenedor_cursos">
</div>
</div>
```

Luego se define una función que será la encargada de inicializar y crear el nuevo control.

```

<script>
function autoCompInit() { // instancia de la fuente de datos
    oACDS = new YAHOO.widget.DS_JSArray(arrayDatos); // Instancia de la
interfaz de auto completacion    oAutoComp = new
YAHOO.widget.AutoComplete('cursos_auto','contenedor_cursos', oACDS);
    oAutoComp.queryDelay = 0;
    oAutoComp.textboxFocusEvent.subscribe(myOnTextboxFocus);
    oAutoComp.textboxBlurEvent.subscribe(myOnTextboxBlur);
    oAutoComp.itemSelectEvent.subscribe(myOnItemSelect);
    function myOnTextboxFocus(sType, aArgs) {

                                }
    }
}
</script>

```

Luego se crea el control y está listo para ser utilizado

```

<script>
YAHOO.util.Event.addListener(this,'load',autoCompInit);
</script>

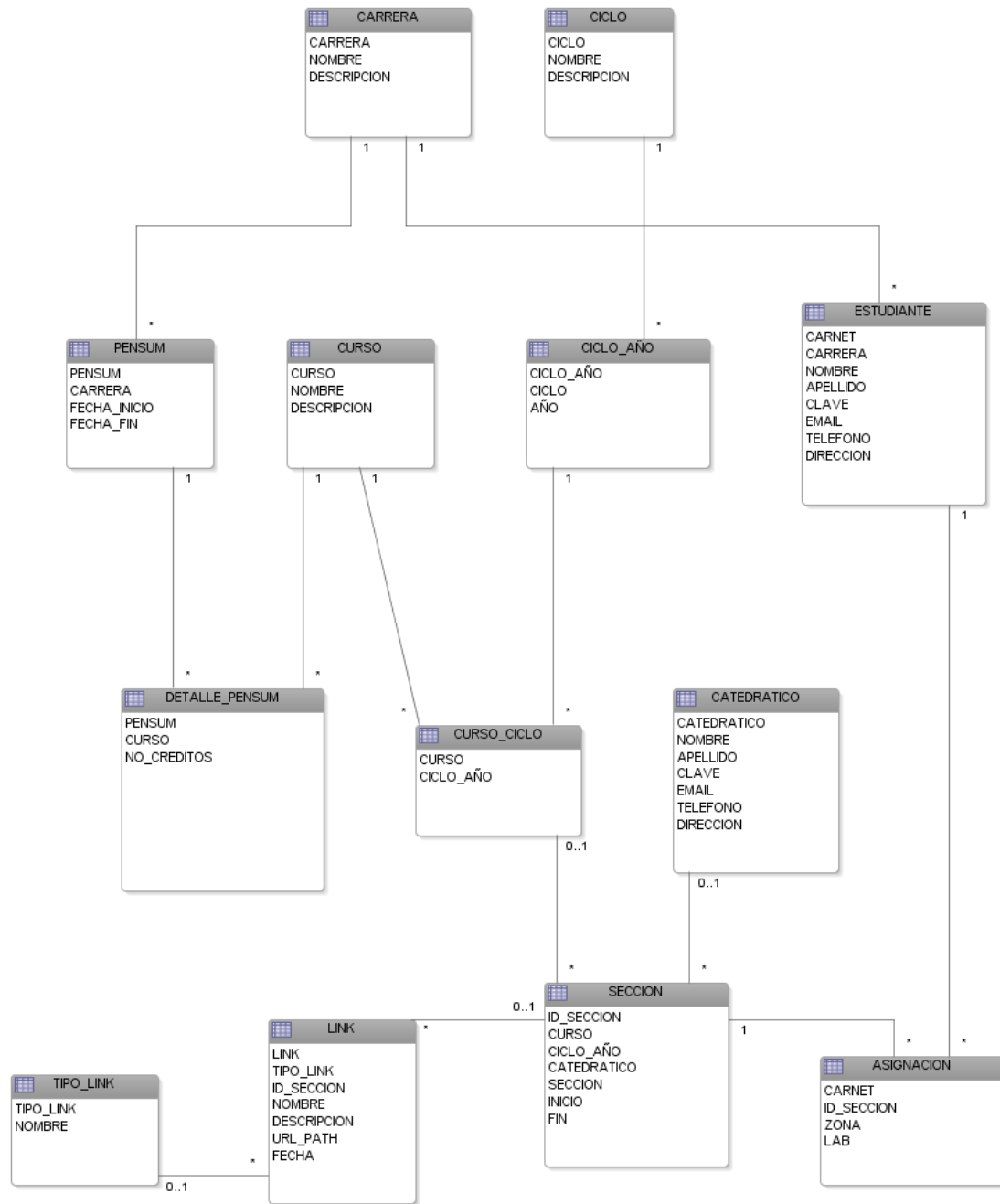
```

Página oficial de YUI:

Yahoo UI Library. <http://developer.yahoo.com/yui/>. (01/03/2008)

Apéndice 4: Modelo de datos del ejemplo AJAX

Figura 33. Diagrama del modelo de datos del ejemplo AJAX



ANEXOS

Anexo 1. Crear el objeto *XMLHttpRequest* en cualquier explorador.

Fuente: Brett McLaughlin. *Mastering AJAX, Part 1: Introduction to AJAX*.

<http://www-128.ibm.com/developerworks/java/library/wa-AJAXintro1.html>.

(01/03/2008)

```
/* Crear un Nuevo objeto XMLHttpRequest para comunicarse con el servidor */
var xmlhttp = false;
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
} catch (e) {
    try {
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e2) {
        xmlhttp = false;
    }
}
@end @*/
if (!xmlhttp && typeof XMLHttpRequest != 'undefined') {
    xmlhttp = new XMLHttpRequest();
}
```

Explicación:

1. Crear la variable, xmlhttp, para referenciar al objeto *XMLHttpRequest* que se creará.

2. Tratar de crear el objeto en los exploradores Microsoft:
 - a. Tratar de crear el objeto usando el *Msxml2.XMLHTTP*.
 - b. Si falla, tratar de crear el objeto usando *Microsoft.XMLHTTP*.
3. Si no se puede crear después de los 2 primeros pasos entonces crear el objeto de la forma estándar.

Al final del proceso, la variable `xmlHttp` tendrá la referencia de un objeto *XMLHttpRequest* válido, sin importar en que explorador se ejecute.