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

APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS

Miguel Alejandro Catalán López

Asesorado por la Inga. Erika Yesenia Corado Castellanos de Lima

Guatemala, enero de 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS

TRABAJO DE GRADUACIÓN

PRESENTADO A JUNTA DIRECTIVA
DE LA FACULTAD DE INGENIERÍA
POR

MIGUEL ALEJANDRO CATALÁN LÓPEZ
ASESORADO POR LA INGA. YESENIA CORADO CASTELLANOS DE LIMA

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN CIENCIAS Y SISTEMAS

GUATEMALA, ENERO DE 2012

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Enrique Alfredo Beber Aceituno
VOCAL II	Ing. Pedro Antonio Aguilar Polanco
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Juan Carlos Molina Jiménez
VOCAL V	Br. Mario Maldonado Muralles
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Murphy Olympo Paiz Recinos
EXAMINADOR	Ing. Juan Álvaro Díaz Ardavin
EXAMINADOR	Ing. Edgar Josué González Constanza
EXAMINADOR	Ing. José Ricardo Morales Prado
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

HONORABLE TRIBUNAL EXAMINADOR

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

APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS

Tema que me fuere asignado por la Dirección de la Escuela de Ingeniería en Ciencias y Sistemas, con fecha marzo de 2011.

Miguel Alejandro Catalán López

Guatemala, 13 de junio de 2011

Ingeniero
Marlon Antonio Pérez Turk
Director
Escuela de Ciencias y Sistemas
USAC

Señor Director:

Por medio de la presente, informo a usted que he revisado el trabajo de graduación titulado: **"APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS"**, el cual fue realizado por el estudiante MIGUEL ALEJANDRO CATALÁN LÓPEZ, el cual encuentro satisfactorio.

Sin otro particular, me es grato suscribirme de usted.

Atentamente,

Inga. Erika Yesenia Corado de Lima
Col. 8418

Inga. Erika Yesenia Corado Castellanos de Lima
Colegiado 8418
Asesor

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

Guatemala, 13 de Julio de 2011

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 **MIGUEL ALEJANDRO CATALAN LOPEZ** carné **2003-13133,** titulado: **"APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS",** 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

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
INGENIERIA EN CIENCIAS Y SISTEMAS
COORDINACION

ESCUELA DE CIENCIAS Y SISTEMAS

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÌA
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 ***"APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS"****, presentado por el estudiante MIGUEL ALEJANDRO CATALÁN LÓPEZ, aprueba el presente trabajo y solicita la autorización del mismo.*

"ID Y ENSEÑAD A TODOS"

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

Guatemala, 19 de enero 2012

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

DTG. 028.2012

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: APLICACIONES ENRIQUECIDAS PARA INTERNET: ESTADO ACTUAL Y TENDENCIAS, presentado por el estudiante universitario Miguel Alejandro Catalán López, autoriza la impresión del mismo.

IMPRÍMASE:

Ing. Murphy Olympo Paiz Recinos
Decano

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

Guatemala, 23 de enero de 2012.

/gdech

ACTO QUE DEDICO A:

Dios Por hacerme una persona bendecida, por regalarme la familia que tengo, por darme las oportunidades que me ha dado y por cada segundo de mi vida en los cuales nunca me ha desamparado. Por darme la oportunidad de estar aquí hoy. Toda la gloria y la honra sean para Él.

Mis padres Por todo el esfuerzo que realizaron para ayudarme a cumplir esta meta, por sus palabras, sus consejos y el tiempo que me regalaron cuando lo necesité. Por el ejemplo que me han dado, por corregirme y apoyarme siempre, aunque a veces las circunstancias no fueran las mejores. Por aguantarme en las mañanas y por buscar siempre mi bienestar y el de mis hermanos. Este objetivo alcanzado es suyo también.

Mi abuela Alicia Por la sabiduría que me brindó con sus palabras, por guiarme desde pequeño por un buen camino, por corregirme y regalarme de su amistad y su conocimiento. Muchas gracias.

Mis hermanos Luis y Jorge, porque de sus actitudes he podido tomar ejemplo para mi vida, por cada plática con ustedes en las que siempre he aprendido algo nuevo, porque hasta en las situaciones más difíciles he podido bromear y soltar carcajadas a su lado.

Mi esposa Karla Por tu compañía durante todo este proceso. Por ser la voz que me da fuerza cuando creo no poder, por ser el equilibrio de mi vida, por estar a mi lado en los momentos más difíciles y en los más alegres. Gracias por regalarme de tu amor y sabiduría, por ser mi amiga y compañera de vida.

Mis amigos y amigas Por el apoyo y amistad que me han brindado y por hacer más llevaderos los momentos difíciles de la carrera. Gracias por sus consejos y por el conocimiento que me permitieron adquirir de ustedes.

Inga. Yesi Corado Por la amistad, consejo y apoyo que me ha brindado en todo momento, gracias por compartir su conocimiento y regalarme de su tiempo, ya que gracias a ello hoy culmina una etapa más en mi vida.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

GLOSARIO

Ancho de banda Capacidad de transmisión de datos que pueden ser enviados a través de una red en un período de tiempo dado.

API Acrónimo de *Application Programing Interface.* Es un conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

API REST Es una librería de funciones, a la que se accede utilizando el protocolo HTTP, por medio de direcciones de Internet o URL en la que se envían los datos que se desea consultar.

Applet Componente de una aplicación que se ejecuta en el contexto de otro programa.

Arquitectura de *software* Conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del *software* en un sistema de información.

ASP Acrónimo de *Active Server Pages*. Tecnología de Microsoft para la escritura de páginas *web,* generadas dinámicamente del lado del servidor.

Assembly En el contexto informático, es una colección de uno o más archivos agrupados juntos para formar una unidad lógica.

Binding En informática, se refiere a una "ligadura" o referencia a otro símbolo más largo y complicado. Este otro símbolo puede ser un valor de cualquier tipo.

Bytecode En informática, es un código intermedio más abstracto que el código de máquina. Habitualmente es tratado como un archivo binario que contiene un programa ejecutable similar a un módulo objeto, que es un archivo binario producido por el compilador, cuyo contenido es el código objeto o código máquina.

CGI Acrónimo de *Common Gateway Interface*. Es una tecnología que permite a un navegador de Internet solicitar datos de un programa ejecutado en un servidor de Internet. Especifica un estándar para transferir datos entre el cliente y el programa.

CLR Acrónimo de *Common Language Runtime*. Es un entorno de ejecución para los códigos de los programas que corren sobre la plataforma Microsoft .NET. La responsabilidad de este es compilar una forma de código intermedio llamada *Common Intermediate Language* (CIL) al código de máquina nativo mediante un compilador en tiempo de ejecución.

Code-behind	Este modelo recomienda que para realizar una programación dinámica, se coloque el código en un archivo separado, o en una etiqueta *script* especialmente diseñada. Los nombres de los archivos *code-behind* están basados en el nombre del archivo ASPX tales como MiPagina.aspx.cs o MiPagina.aspx.vb.
Código administrado	Código fuente que no se ejecuta directamente sobre el sistema operativo de un dispositivo, sino que en vez de ello se apoya en un ambiente de ejecución. Este ambiente de ejecución se encarga de la asignación de recursos, seguridad, etc.
Código fuente	Conjunto de líneas de texto que representan instrucciones que debe de seguir la computadora para ejecutar un programa informático.
Código no administrado	Código fuente que al compilarse genera un archivo binario que es ejecutado directamente por la computadora.
Compilador	Programa informático encargado de traducir un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que pueda ser ejecutado por la computadora.
Comunicación asíncrona	Comunicación establecida de forma diferida en el tiempo, es decir sin coincidencia temporal.

Comunicación síncrona Comunicación que permite a uno o varios emisores y receptores intercambiar mensajes en el mismo instante en que se transmiten.

Cross-browser Término utilizado para describir aplicaciones *web* que son compatibles con todos los navegadores para Internet, sin importar la versión que se esté utilizando.

Desacoplado Se llama así a los programas de *software* que no se encuentran fuertemente atados a otras piezas de *software* para su funcionamiento.

DHTML Acrónimo de *Dynamic HyperText Markup Language*. Es un conjunto de técnicas que permiten crear sitios *web* interactivos, utilizando una combinación de HTML estático y *scripting* como por ejemplo JavaScript.

DirectX Es una colección de APIs desarrolladas para facilitar las tareas relacionadas con multimedia, especialmente programación de juegos y video en la plataforma Microsoft Windows.

DOM Acrónimo de *Document Object Model*. Es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

ECMA	Organización internacional basada en membresías de estándares para la comunicación y la información. Estandariza los sistemas computarizados.
Encapsulamiento	Principio de la Programación Orientada a Objetos que indica que se debe de ocultar el estado de un objeto, de manera que solo pueda ser cambiado mediante las operaciones definidas por ese objeto.
Experiencia de uso	Conjunto de factores y elementos relativos a la interacción del usuario con un entorno o dispositivo concreto, cuyo resultado es la generación de una percepción positiva o negativa de dicho servicio, producto o dispositivo.
Framework	Estructura conceptual y tecnología de soporte, definida normalmente con módulos de *software* concretos, con base en lo cual otro proyecto de *software* puede ser organizado y desarrollado.
GNU	Proyecto iniciado por Richar Stallman para crear sistemas operativos completamente libres.
Hiperenlace	También llamado hipervínculo. Es un elemento en un documento electrónico que hace referencia a otro recurso que puede ser accedido en distintas formas, como visitarlo con un agente de navegación, mostrarlo como documento referenciado o guardarlo localmente.

HTTP	Acrónimo de *HyperText Transfer Protocol*. Protocolo de comunicación utilizado en la *World Wide Web* orientado a transacciones que sigue el esquema petición-respuesta entre un cliente y un servidor.
IDE	Acrónimo de *Integrated Development Environment*. Es un programa informático compuesto por un conjunto de herramientas de programación. Normalmente incluye un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.
Interfaz de usuario	Medio por el cual el usuario puede comunicarse e interactuar con una máquina, un equipo o una computadora y comprende todos los puntos de contacto entre el usuario y el equipo.
Intérpretes	Programa informático capaz de analizar y ejecutar programas en tiempo de ejecución, realizando la traducción a medida que es necesaria.
Intranet	Red de computadoras privadas que utiliza tecnología de Internet para compartir dentro de una organización parte de sus sistemas de información y sistemas operacionales.
IoC	Acrónimo de *Inversion of Control*. Principio abstracto utilizado en la arquitectura de *software* en la cual el flujo del control de un sistema es invertido en comparación, respecto de su funcionamiento en la programación procedural.

Java Virtual Machine	Es un programa que se ejecuta desde una plataforma de *hardware* específica, capaz de ejecutar e interpretar instrucciones expresadas en código binario especial (Java bytecode) el cual es generado por el compilador del lenguaje Java.
JRE	Acrónimo de *Java Runtime Environment*. Es un conjunto de utilidades que permite la ejecución de programas Java, este se encuentra conformado por una Java Virtual Machine, un conjunto de bibliotecas Java y otros componentes necesarios para la ejecución de programas Java.
Manejador de eventos	Es una función dentro de un programa informático que dado un conjunto de eventos, realiza una acción específica.
Mashup	Es un sitio *web* o aplicación *web* que usa y combina datos y contenido de otras aplicaciones *web* para crear un nuevo contenido completo, consumiendo servicios directamente, a través del protocolo HTTP.
Metadata	Son datos que describen otros datos.
Motor de búsqueda	Sistema informático que busca archivos almacenados en servidores *web,* gracias a un rastreador (*crawler*).

Multimedia	Utilizado para referirse a cualquier objeto o sistema que utiliza múltiples medios de expresión para presentar o comunicar información.
ORM	Acrónimo de *Object Relationship Model*. Técnica de programación para convertir datos entre el sistema de tipos, utilizado en un lenguaje orientado a objetos y el utilizado en una base de datos relacional.
Paradigma	Es un modelo o patrón en cualquier disciplina científica u otro contexto epistemológico.
Protocolo	En informática, es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red.
Proxy	En informática, es un programa o dispositivo que realiza una acción en representación de otro.
SDK	Acrónimo de *Software Development Kit*. Es un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto.
Servidor	En informática, representa una computadora que forma parte de una red y que provee servicios a otras computadoras denominadas clientes.
Stand-alone	Se utiliza para referirse a aplicaciones que son autónomas o independientes.

Streaming	Consiste en la distribución de audio o vídeo por Internet de forma continua.
Subversion	Sistema de control de versiones en el que al realizar el versionado de un archivo, todo el repositorio se incrementa, no únicamente las versiones de los archivos.
TI	Acrónimo de Tecnologías de la Información. Se define como el estudio, diseño, desarrollo, implementación, soporte o dirección de los sistemas de información computarizados, en particular del *software* de aplicación y *hardware* de computadoras.
3D	Acrónimo de tres dimensiones. Utilizado para referirse a programas que pueden representar objetos y ambientes simulando su largo, alto y ancho.
URI	Acrónimo de *Uniform Resource Identifier*. Es una cadena corta de caracteres que identifica inequívocamente un recurso que es accesible en una red o sistema.
WebKit	Plataforma para el renderizado HTML de los navegadores *web*.
World Wide Web	Es un sistema de distribución de información basado en hipertexto o hipermedios enlazados y accesibles, a través de la Internet.

XML	Acrónimo de *eXtensible Markup Language*. Es un metalenguaje extensible de etiquetas que permite definir lenguajes para diferentes necesidades.

RESUMEN

Conforme el paso del tiempo las aplicaciones *web* han evolucionado en función de distintas situaciones, como la accesibilidad que han encontrado mediante nuevos dispositivos que proveen conexión a la Internet, el uso de navegadores para dispositivos móviles, el creciente número de usuarios, la mejora en cuanto a conectividad que poseen los usuarios para acceder a la red y las mejoras en cuanto a *hardware* y *software* que han facilitado el acceso a dicha red, en donde las aplicaciones *web* síncronas han dejado de ser suficientes debido a las dificultades que proveen para la gestión de datos y en la usabilidad e interfaces poco atractivas y amigables que ofrecen a los usuarios.

Debido a esta deficiencia que presentaban las aplicaciones *web* se da el surgimiento de las Aplicaciones Enriquecidas para Internet, que satisfacen las necesidades que las aplicaciones *web* síncronas no habían podido satisfacer, dadas las limitantes que poseen en cuanto a su comportamiento y arquitectura interna.

Las Aplicaciones Enriquecidas para Internet han permitido satisfacer estas necesidades a través de características propias que las diferencian de las aplicaciones *web* síncronas, estas capacidades son descritas comúnmente como la emulación del ambiente de una aplicación de escritorio en un ambiente *web*.

Gracias a la aceptación que ha tenido este paradigma en miles de aplicaciones publicadas en la Internet, se han desarrollado distintas herramientas para la construcción de Aplicaciones Enriquecidas para Internet. Entre estas herramientas se encuentran Adobe Flash Builder, Microsoft Silverlight y Oracle JavaFX, además de alternativas no tan populares como lo son OpenLaszlo, Mono Moonlight y el recientemente liberado HTML5.

A medida que el nuevo paradigma fue ganando terreno se creó el contexto en el cual pudo ser implementado en conjunto con otros paradigmas que se encuentran actualmente en la Internet. Gracias a las mejoras y el aprovechamiento de los beneficios de las Aplicaciones Enriquecidas para Internet, se ha logrado adaptar de forma muy versátil a las necesidades de los nuevos paradigmas. Esta adaptación y uso se puede encontrar en la gran mayoría de las aplicaciones *web* 2.0, en la *web* 3.0 y en el traslado que han tenido recientemente los sistemas informáticos a la Internet para facilitar y reducir costos de implementación a las empresas, como ocurre en el caso del *Cloud Computing*, además de otras tendencias que actualmente se encuentran aún en desarrollo tales como la *web* 4.0.

Como comúnmente ocurre, este paradigma ha ido implementándose en distintos países, cada uno con un ritmo de crecimiento y adaptación propio, el cual se ve marcado normalmente por la curva de aprendizaje que todo cambio implica. Este proceso existe también entre los profesionales que están involucrados en alguna parte de todo el proceso de la construcción del *software* en Guatemala, en donde existen razones por las cuales este paradigma es utilizado y razones por las cuales no, estableciendo así el estado actual dentro del *software* creado en Guatemala.

OBJETIVOS

General

Presentar un análisis del paradigma de las Aplicaciones Enriquecidas para Internet como una tecnología emergente para la creación de aplicaciones *web* en el ámbito nacional.

Específicos

1. Definir el paradigma de las Aplicaciones Enriquecidas para Internet y presentar un análisis comparativo contra aplicaciones *web* tradicionales.

2. Listar al menos cuatro características que identifiquen la arquitectura de las Aplicaciones Enriquecidas para Internet.

3. Exponer al menos seis herramientas que puedan ser utilizadas para la creación de Aplicaciones Enriquecidas para Internet e identificar sus características y fortalezas.

4. Evaluar el impacto de las Aplicaciones Enriquecidas para Internet en el desarrollo de aplicaciones *web* a través de un análisis de tendencias.

5. Evaluar la utilización del paradigma de Aplicaciones Enriquecidas para Internet en Guatemala, a través de un estudio de mercado realizado en empresas del sector público y privado.

INTRODUCCIÓN

La Internet ha sido desde su creación la red más grande y utilizada a nivel mundial, contando con millones de usuarios en todo el mundo, transmitiendo enormes cantidades de datos entre distintos sitios geográficos y almacenando en toda la infraestructura que la conforma una inmensa cantidad de información, la cual es accedida, creada y utilizada por los usuarios.

Mediante la búsqueda de proveer herramientas para facilitar la creación y consumo de información almacenada en la Internet, se han creado sistemas informáticos capaces de publicarse en redes de computadoras y hacerlos accesibles a los usuarios. Esta necesidad a través del tiempo ha ido evolucionando, desde la creación de *software* instalado en una computadora cliente que se comunica a un servidor de datos, hasta las ya conocidas aplicaciones *web*.

Esta evolución ha seguido un proceso de mejora buscando facilitar la creación, publicación y distribución de estas aplicaciones *web*, hasta llegar a crear infraestructuras en las que existen servidores especializados en donde se ofrecen servicios tales como el de acceso a una aplicación *web*, el de datos o de seguridad, los cuales son consumidos por computadoras cliente. Estos servicios necesitan únicamente de un *software* cliente capaz de conectarse a una red y que permita la comunicación en doble vía utilizando el protocolo de transferencia de datos HTTP; actualmente este *software* es conocido como navegador *web*.

Las aplicaciones *web* fueron mejorando poco a poco, iniciándose como clientes instalados localmente en una computadora personal que se conectaba a un servidor que proveía únicamente los datos que necesitaba. Estas aplicaciones fueron reemplazadas debido a las dificultades que se presentaban al desear actualizar la aplicación, ya que se debía instalar las actualizaciones en cada cliente. Esto se dificultaba cuando las ubicaciones geográficas se encontraban distantes.

Posterior a estas fueron creados los sitios *web*, en donde en un servidor con acceso a la red eran publicadas páginas estáticas que eran administrables únicamente accediendo y cambiando manualmente el contenido de dichas páginas. Estas fueron descartadas debido a la poca usabilidad e interacción que proveían a los usuarios.

Más adelante fueron creadas las aplicaciones *web*, las cuales ya eran capaces de hacer uso de código fuente de alto nivel para ofrecer una mejor interacción con el usuario, creando páginas que podían cambiar su contenido en función de las acciones que ejecutara el usuario. Posteriormente estas acciones eran procesadas por un servidor. Estas páginas eran publicadas en un servidor *web* el cual hacía uso de un API que ejecutaba el código fuente que contenía la página y que recibía la acción que había realizado el usuario y enviaban de vuelta al cliente la página *web* con la respuesta a dicha acción. Además de esto ofrecía conexión a bases de datos con lo cual le permitía la gestión de información dentro de la aplicación *web*.

Este paradigma fue muy útil ya que se podían crear aplicaciones complejas que fueran accesibles desde la Internet, proveyendo de muchos beneficios a empresas y usuarios, como por ejemplo la disminución de costos de implementación, la facilidad de realizar actualizaciones a la aplicación de

forma transparente al usuario y la amplia accesibilidad que ofrecían. El beneficio de la accesibilidad era posible gracias a que se podía ingresar a la aplicación desde cualquier lugar, siempre y cuando se tuviera acceso a la red en la que se encontrara disponible la aplicación, por lo que tampoco era necesaria una instalación local de la aplicación en las computadoras clientes.

A pesar de los grandes avances que se habían logrado aún era necesario el poder brindar mayor usabilidad y enriquecer la interfaz de usuario de forma que se pudiera ofrecer una experiencia de uso similar a las aplicaciones de escritorio en donde no se tuvieran pantallas en blanco al enviar una solicitud al servidor, en las que se pudieran utilizar elementos multimedia sin restricciones y en donde la información pudiera ser manipulada de una mejor manera por el usuario, haciendo de las aplicaciones *web* aplicaciones intuitivas y atractivas al usuario; es debido a estas necesidades que se crea el paradigma de Aplicaciones Enriquecidas para Internet.

En el presente trabajo, se incluyen, en su orden, la historia, ventajas y desventajas de las Aplicaciones Enriquecidas para Internet, sus herramientas de desarrollo, los diferentes paradigmas y las tendencias del desarrollo web, tomando en cuenta, las interfaces de usuario, la gestión de información y la expansión hacia otros dispositivos.

Se da a conocer también un estudio de mercado, cuyo análisis e interpretación, permite fortalecer el paradigma de las Aplicaciones Enriquecidas para Internet, cuya importancia radica en ofrecer a los usuarios una interfaz que permita aprovechar las bondades de las aplicaciones web, proveyendo las interfaces más potentes o enriquecidas en multimedia.

1. APLICACIONES ENRIQUECIDAS PARA INTERNET (RIA)

Las aplicaciones Enriquecidas para Internet, también conocidas como RIA por las iniciales en inglés del término Rich Internet Applications, son un nuevo paradigma en cuanto al desarrollo de aplicaciones *web*. Este nuevo paradigma se ha visto potenciado debido a la importancia de ofrecer a los usuarios una interfaz por medio de la cual se puedan aprovechar las bondades que brindan las aplicaciones *web*, proveyéndoles de interfaces más potentes o enriquecidas en multimedia e intuitivas.

En este capítulo se presenta una introducción a lo que son las aplicaciones *web* y la evolución que han sufrido para llegar a convertirse en lo que se conoce hoy como Aplicaciones Enriquecidas para Internet. Se realiza un recorrido a través de la evolución de las aplicaciones *web* y los paradigmas que han formado parte de ella, para posteriormente conocer qué provocó la aparición de las Aplicaciones Enriquecidas para Internet.

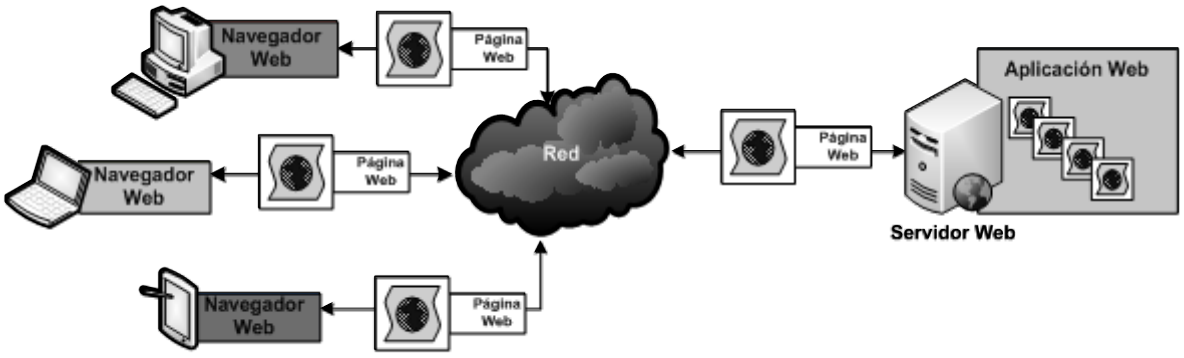
1.1. Aplicaciones *web*

Se les llama aplicaciones *web* a todo *software* que se encuentra disponible para un grupo de usuarios definidos a través de la Internet o de una Intranet, permitiendo de esta forma, acceder a la aplicación por medio de las facilidades de comunicación que proveen las redes de computadoras. Este *software* se encuentra alojado en un servidor *web* el cual administra y gestiona el uso de dicho *software*.

Las aplicaciones *web* se encuentran normalmente constituidas por archivos *web*, los cuales se presentan al usuario mediante lenguajes interpretados, como HTML y XHTML. Estos archivos son interpretados por un navegador de Internet, el cual transforma el código fuente en una interfaz entendible para el usuario.

Las aplicaciones *web* hacen uso de los clientes livianos (*thin-clients*), los cuales interactúan con el servidor *web*; el servidor *web* comparte sus recursos con todos los clientes conectados a él, mientras que los clientes reservan sus recursos para ser utilizados única y exclusivamente por ellos mismos. La característica que distingue a los clientes livianos es que no requieren de una gran cantidad de recursos de la computadora para poder interactuar con el usuario y con el servidor *web*, sino que es el servidor *web* quien maneja las mayores cargas de procesamiento, de almacenamiento y de memoria RAM. En la figura 1 se representa el funcionamiento de una aplicación *web*.

Figura 1. **Funcionamiento de las aplicaciones *web***



Fuente: elaboración propia.

Las aplicaciones *web* se han convertido, con el paso del tiempo y el crecimiento de la Internet, en una solución eficiente para los problemas de gestión de información de pequeñas, medianas y grandes empresas, proveyéndoles de beneficios como alta disponibilidad, descentralización, accesibilidad, mantenimiento y actualización de una forma rápida, sencilla y transparente para el usuario.

1.1.1. Historia de las aplicaciones *web*

El uso del *software* como una herramienta para facilitar el trabajo y hacerlo más rápido se inició con las aplicaciones de escritorio, éstas eran instaladas en el disco duro de una computadora de forma local. Las configuraciones necesarias, los archivos ejecutables y demás requerimientos de aplicación, eran copiados a la computadora cliente, así como las modificaciones que el instalador del *software* necesitaba realizar en muchos casos a los archivos del registro del sistema operativo de la computadora, para poder funcionar correctamente.

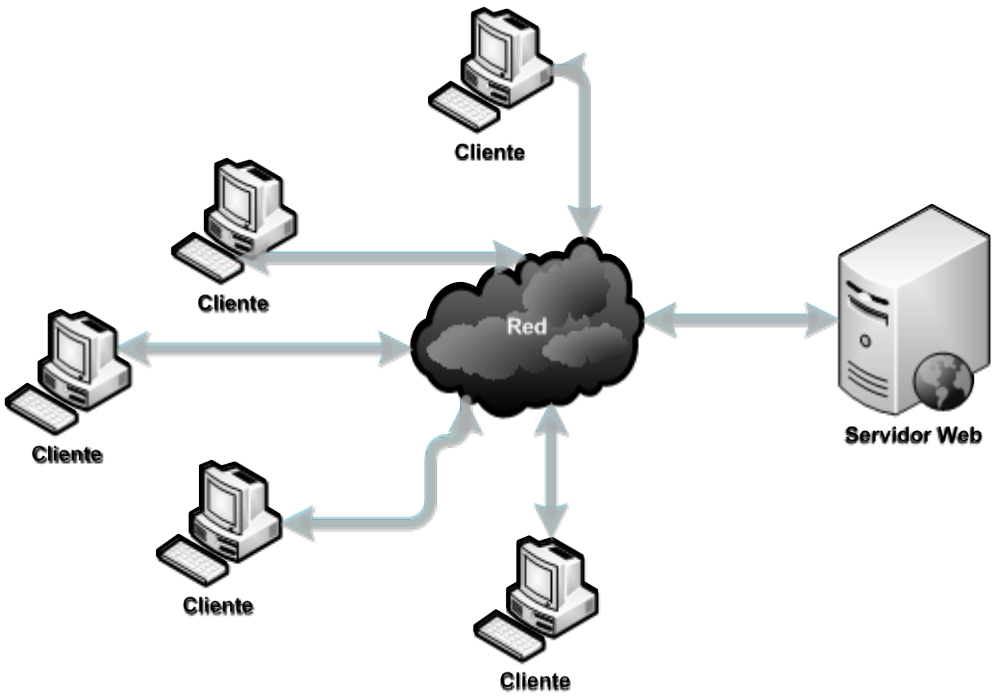
El tener aplicaciones de escritorio representaba una pérdida de espacio en disco duro, uso de procesador y de memoria RAM, debido a que el mismo *software* era ejecutado en muchas computadoras. Adicionalmente, se encontraban complicaciones al intentar obtener toda la información almacenada en cada una de las computadoras personales.

Con la implementación de las redes de computadoras, se empezó a utilizar un nuevo tipo de aplicaciones basadas en un modelo cliente-servidor, en donde una computadora personal que funciona como cliente, solicita información a una computadora especializada llamada servidor. El servidor

recibe y procesa la petición del cliente y envía una respuesta, la cual es recibida por el cliente y mostrada al usuario.

En el inicio del modelo cliente-servidor se utilizaba *software* cliente, el cual era instalado en la computadora cliente. Éste proveía de una interfaz de usuario, de comunicación, de entrada y salida de la computadora cliente. Por otro lado, se encontraba el servidor de aplicaciones, el cual contenía una aplicación que era la encargada de recibir, procesar y responder a las solicitudes de todos los clientes conectados a él. En la figura 2 se muestra cómo se encuentra conformado el modelo cliente-servidor.

Figura 2. **Modelo cliente-servidor**



Fuente: elaboración propia.

Este modelo respondía de mejor manera a las empresas debido a que la gestión de datos e información se realizaba de manera centralizada; sin embargo al realizar actualizaciones al *software,* era necesario hacerlo tanto en el cliente como en el servidor, lo cual representaba una desventaja.

Para mejorar este tipo de aplicaciones se crearon las aplicaciones *web*, las cuales están basadas en documentos *web* que son escritos en lenguaje HTML. Estos documentos son interpretados por una interfaz que reside del lado del cliente, conocida como navegador de Internet. Con este tipo de *software* únicamente se debía tener como instalación local cualquier navegador, el cual hace las solicitudes de documentos *web* al servidor de aplicaciones y éste responde con un documento *web* que contiene dentro de sí la respuesta a la solicitud realizada por el cliente.

Esta comunicación entre cliente y servidor es llevada a cabo mediante un protocolo estándar para la transferencia de información, llamado HTTP. Este protocolo divide los mensajes enviados y recibidos en paquetes, los cuales son decodificados y ordenados en función de la información que es enviada en el encabezado de dicho paquete. También provee instrucciones para poder enviar información al servidor y *metadata*, que incluye un código que indica el resultado de la respuesta del servidor *web*.

Véase la tabla I para encontrar el significado de cada uno de los códigos de respuesta del protocolo HTTP.

Conforme fue avanzando el paradigma de las aplicaciones *web* se fueron buscando nuevas tecnologías que pudieran ayudar a brindar una mejor experiencia de usuario por medio de la reducción de tiempos de respuestas, creación de una comunicación más segura y optimización del envío y recepción

de mensajes entre cliente y servidor entre otras mejoras. Es así como en 1995 Netscape introduce el primer lenguaje de *scripting* del lado del cliente, llamado JavaScript, el cual permite a los desarrolladores de aplicaciones *web* brindar cierta dinamicidad en las interfaces de usuario, de manera que se redujera la cantidad de solicitudes enviadas al servidor. Este lenguaje de *scripting* permite trabajar y validar datos sin necesidad de crear una nueva solicitud al servidor.

Tabla I. **Códigos de respuesta HTTP**

1xx	**Respuesta informativa: petición recibida, continúa el proceso**	
	100	Continúa
	101	Conmutando protocolos
	102	Procesando (*WebDAV - RFC 2518*)
2xx	**Petición correcta: petición recibida correctamente, entendida y aceptada**	
	200	OK
	201	Creado
	202	Aceptado
	203	Información no autoritativa (desde HTTP/1.1)
	204	Sin contenido
	205	Recargar contenido
	206	Contenido parcial
	207	Estado múltiple (*Multi-Status, WebDAV*)
3xx	**Redirección: se necesita de la interacción con el cliente para completar la petición**	
	300	Múltiples opciones
	301	Movido permanentemente
	302	Movido temporalmente
	303	Vea otra (desde HTTP/1.1)

Continuación tabla I.

	304	No modificado
	305	Utilice un *proxy* (desde HTTP/1.1)
	306	Cambie de *proxy*
	307	Redirección temporal (desde HTTP/1.1)
4xx	**Error del cliente: la solicitud lleva errores o no puede procesarse**	
	400	Solicitud incorrecta
	401	No autorizado
	402	Pago requerido
	403	Prohibido
	404	No encontrado
	405	Método no permitido
	406	No aceptable
	407	Autenticación *proxy* requerida
	408	Tiempo de espera agotado
	409	Conflicto
	410	Ya no disponible
	411	Requiere longitud
	412	Falló precondición
	413	Solicitud demasiado larga
	414	URI demasiado larga
	415	Tipo de medio no soportado
	416	Rango solicitado no disponible
	417	Falló expectativa
	421	Hay muchas conexiones desde esta dirección de Internet
	422	Entidad no procesable (*WebDAV - RFC 4918*)
	423	Bloqueado (*WebDAV - RFC 4918*)
	424	Falló dependencia (*WebDAV - RFC 4918*)

Continuación tabla I.

	425	Colección sin ordenar
	426	Actualización requerida
	449	Reintente con
5xx	**Error del servidor: el servidor no pudo responder a una petición aparentemente válida**	
	500	Error interno
	501	No implementado
	502	Pasarela incorrecta
	503	Servicio no disponible
	504	Tiempo de espera de la pasarela agotado
	505	Versión de HTTP no soportada
	506	Variante también negocia (*RFC 2295*)
	507	Almacenamiento insuficiente (*WebDAV - RFC 4918*)
	509	Límite de ancho de banda excedido
	510	No extendido (*RFC 2774*)

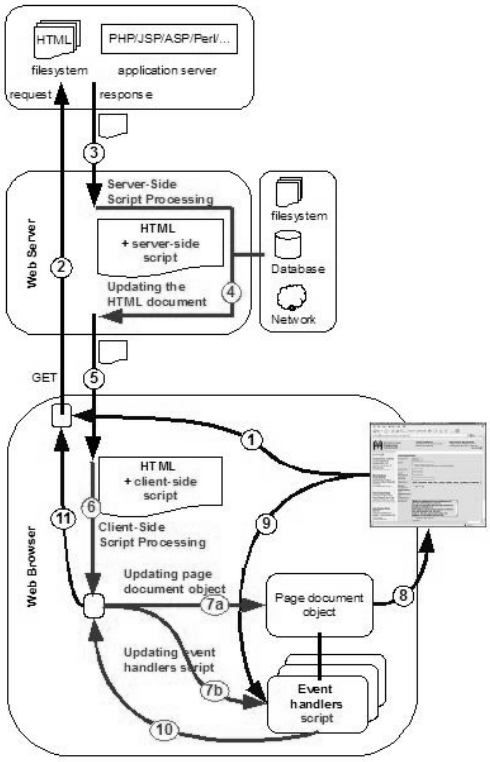
Fuente: http://es.wikipedia.org/wiki/Anexo:C%C3%B3digos_de_estado_HTTP. 20 de febrero de 2011.

Véase la figura 3 para apreciar cómo es el funcionamiento del *scripting*, tanto del lado del cliente como del lado del servidor.

En el paso 1 que se muestra en la figura 3, la aplicación *web* realiza una solicitud mediante el navegador a través de la instrucción *GET* del protocolo HTTP. En el paso 2, la solicitud es recibida por el servidor *web* y redireccionada al servidor de aplicaciones que está ejecutando la aplicación *web,* creando un *request* para dicha aplicación.

La solicitud es recibida por el servidor de aplicaciones, se recopila la información del archivo HTML y se procesa cualquier tipo de *scripting* asociado al archivo HTML (JSP, ASP, PHP, etc.). En el paso 3, se envía una respuesta al servidor *web*, creando un *response.* En el paso 4, es procesado el archivo HTML con cualquier modificación o actualización de datos que haya provocado el *scripting* en el servidor de aplicaciones, accede a cualquier recurso que sea necesario y que esté disponible para la aplicación *web*; esto puede ser una base de datos, un sistema de archivos u otros sitios dentro de la red. En el paso 5, al haber recopilado la información necesaria, es enviada al navegador *web* nuevamente.

Figura 3. **Arquitectura del *scripting***



Fuente: http://www.herongyang.com/JavaScript/Web-Scripting-Architecture-Overview.html. 05 de abril de 2011.

En el paso 6, se interpreta y ejecuta cualquier instrucción o acción que haya sido enviada por el servidor *web*, para ser ejecutada en el lado del cliente. En el paso 7, es ejecutada o interpretada cada una de las instrucciones y/o acciones. En el paso 8, se actualiza el objeto de la página *web* mediante el *scripting* del lado del cliente y es mostrada la respuesta al usuario, en un formato gráfico mediante el navegador *web*.

Es probable que la aplicación *web* ejecute acciones que no requieran de comunicación con el servidor *web*, sino que sean manejadas por *scripts* del lado del cliente. En estas circunstancias, la aplicación *web* pasará por el paso 9, a ejecutar un manejador de eventos construido a través de un *script*, para posteriormente, ser procesado en el paso 10 y regresar al paso 7, a actualizar el objeto de la página *web* modificado por los *scripts* del lado del cliente, y mostrará el resultado al usuario en el paso 8, a través del navegador *web*.

Es de esta forma como se desenvuelve el uso del *scripting* en las aplicaciones *web*, para brindar una dinamicidad que antes no era posible brindar al usuario, mediante el uso de HTML.

Posterior al surgimiento de JavaScript, una empresa adquirida por Adobe en abril de 2005[1], crea una herramienta con el fin de superar las capacidades que proveía hasta entonces JavaScript, y es así como nace Flash. Este funciona a través de un *plug-in* que debe de ser instalado en los navegadores de Internet, el cual provee compatibilidad *cross-browser*. Flash, al igual que JavaScript, provee de un lenguaje de programación que permite una mejora en la interfaz de usuario, proveyendo una disminución de solicitudes enviadas al

[1] BBC News. *Adobe buys Macromedia for $3.4bn.* <http://news.bbc.co.uk/2/hi/business/4456895.stm>

servidor. Su principal diferenciador es el uso de animaciones que potencian la interfaz del usuario para hacerla más amigable.

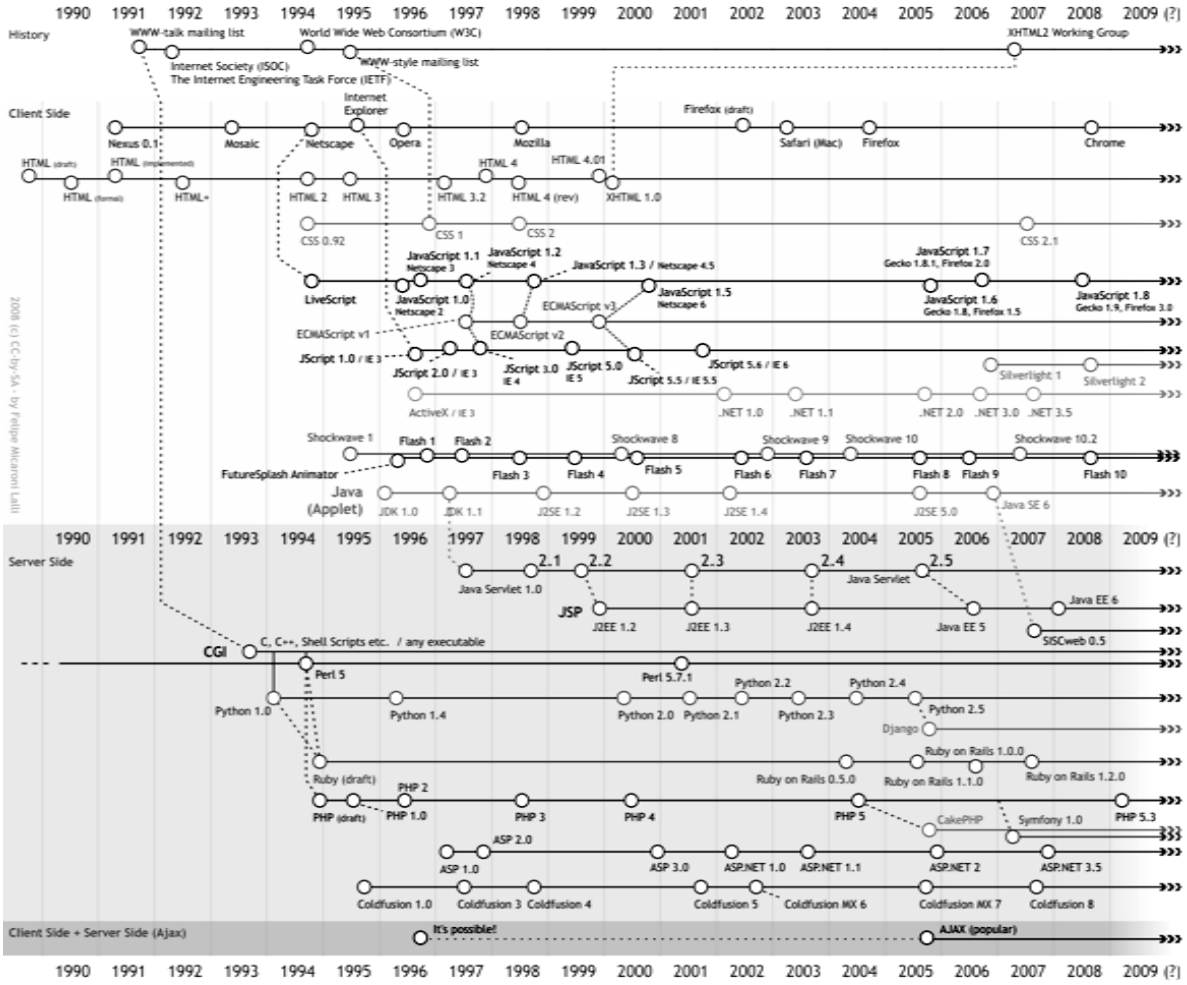
En 1999 una empresa, adquirida por Oracle en abril de 2009[2], lanza oficialmente el término aplicaciones *web*, con el lenguaje Java en la *Servlet Specifications* versión 2.2. Este documento define por primera vez el término de aplicaciones *web*, de la siguiente manera:

"Las aplicaciones *web* son una colección de *servlets*, *JavaServer Pages*, documentos HTML, y otros recursos tales como imágenes, archivos comprimidos, y otros datos. Una aplicación *web* puede estar empacada dentro de un archivo o estar organizada en una estructura de directorios abiertos. Todos los contenedores de *servlets* que sean compatibles, deben de permitir aplicaciones *web* y permitir despliegues de contenido en tiempo de ejecución. Esto puede implicar que los contenedores permitan ejecutar la aplicación directamente utilizando un archivo de la aplicación *web* o bien puede significar que este moverá el contenido de la aplicación *web* dentro de las locaciones apropiadas para ese contenedor en particular."[3]

El proceso de evolución del desarrollo de aplicaciones *web* se ha visto acompañado de una evolución de herramientas y otras tecnologías que han permitido brindar el entorno para crear las Aplicaciones Enriquecidas para Internet.

[2] TILLMAN, Karen. *Oracle Buys Sun*. <http://www.oracle.com/us/corporate/press/018363>.
[3] DUNCAN DAVIDSON, James, COWARD, Dany. *Java Servlet Specification, v2.2.*.

Figura 4. **Evolución del desarrollo *web***



Fuente: http://en.wikipedia.org/wiki/Web_development#Timeline. 04 de abril de 2011.

En la figura 4 se muestra el proceso de evolución del desarrollo de aplicaciones *web,* en el que se distinguen específicamente cuatro líneas de tiempo:

- Historia**:** muestra la formación de los grupos o consorcios creados para la regulación y establecimiento de estándares de la *web*.

- Lado del cliente: abarca las tecnologías utilizadas del lado del cliente, en este caso el surgimiento de los navegadores de Internet y los lenguajes de programación que pueden ser ejecutados del lado del cliente.

- Lado del servidor: abarca las tecnologías utilizadas para la creación de aplicaciones *web* que son ejecutadas del lado del servidor.

- Lado del cliente + Lado del servidor: es aquí donde encajan las Aplicaciones Enriquecidas para Internet, en este caso indicado por el término AJAX, por las capacidades de comunicación asíncrona que esta herramienta provee.

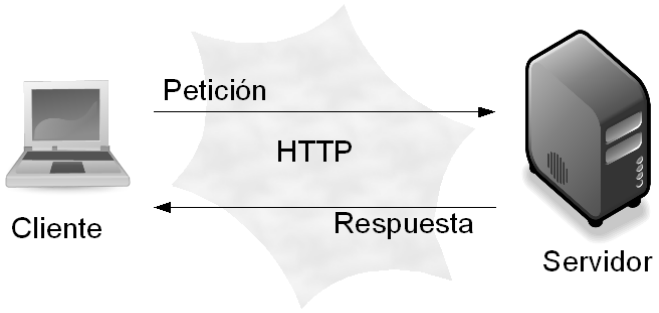
1.1.2. Arquitectura de las aplicaciones *web*

1.1.2.1. Modelo cliente-servidor

Las aplicaciones *web* son comúnmente desplegadas sobre servidores de aplicaciones desde donde son consumidas por múltiples clientes. Esta interacción es posible debido a que se encuentran conectados mediante un modelo cliente-servidor.

El modelo cliente-servidor se encuentra basado en la existencia de una computadora principal o servidor la cual se encarga de recibir las solicitudes de los clientes, procesarlas y enviar la respuesta al cliente, como se muestra en la figura 5.

Figura 5. **Estructura del modelo cliente-servidor**



Fuente: http://www.di.uniovi.es/~labra/cursos/Web20/images/WWW.png. 05 de abril de 2011.

1.1.2.1.1. Ventajas

- Debido a que las aplicaciones se encuentran contenidas dentro de una misma computadora, las actualizaciones o cambios realizados en la aplicación *web* son aplicados únicamente en una computadora que es el servidor, y se realiza completamente transparente al usuario.

- Siendo una única computadora la que provee la información a los clientes, la seguridad puede ser gestionada de mejor manera, ya que se puede restringir el acceso a cierta información dentro del servidor, a un grupo seleccionado de clientes.

1.1.2.1.2. Desventajas

- El servidor es un punto crítico dentro de la estructura de la aplicación *web*, ya que al perderse conectividad con él, la aplicación *web* no se encontrará disponible para los clientes.

- Toda la carga de procesamiento debe de ser soportada por el servidor, lo cual puede ocasionar un desempeño lento de la aplicación *web*.

- Tiempos de respuesta más lentos, debido al consumo de ancho de banda por todas las solicitudes de los clientes hacia el servidor.

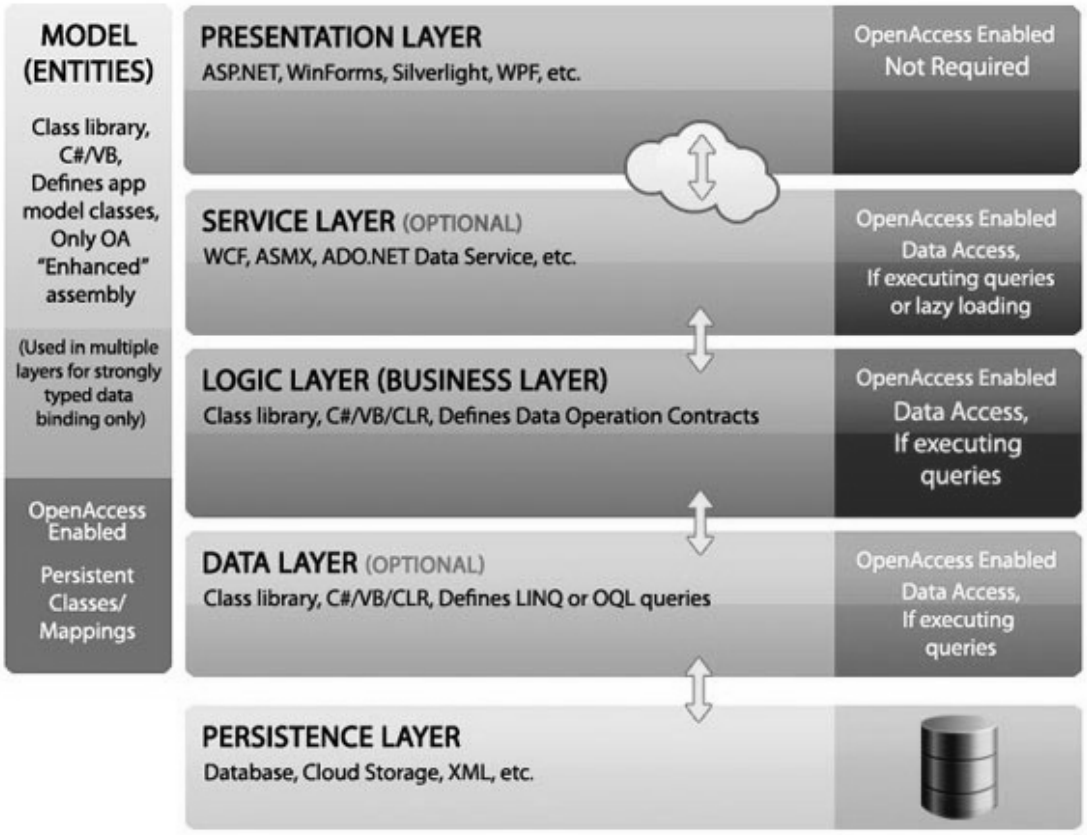
A pesar de las desventajas que presenta este modelo, el modelo cliente-servidor continúa siendo la base para la presentación de la mayoría de aplicaciones *web* que se encuentran hoy día; siendo la excepción más representativa, las aplicaciones que se encargan de las descargas de *software* o compartición de datos como los *Torrents*, que utilizan un modelo punto-a-punto para aprovechar el ancho de banda que provee este modelo.

1.1.2.2. Arquitectura de n-capas

La arquitectura de las aplicaciones *web* ha ido evolucionando con el tiempo. Esta evolución ha buscado la forma de aislar mejor los comportamientos de las aplicaciones *web* de forma que sean lo más desacopladas posible.

El modelo arquitectónico más conocido es el modelo de “n” capas. Cada capa representa un nivel de la aplicación, el cual tiene una responsabilidad y comportamiento único, que es accesible únicamente a través de una interfaz que se expone entre las capas. Un ejemplo de este modelo se muestra en la figura 6.

Figura 6. **Ejemplo de una arquitectura de n-capas con tecnologías Microsoft**



Fuente: http://thinkerclub.wordpress.com/2009/07/30/n-tier-architecture/. 08 de abril de 2011.

Las capas más comunes de encontrar dentro de este modelo arquitectónico son las siguientes:

- Capa de presentación: esta capa es la encargada de presentar la interfaz gráfica al usuario. Se presenta mediante archivos HTML que son interpretados por el navegador de Internet y mostrados al usuario.

- Capa de negocio: capa que contiene toda la lógica del negocio, es decir las reglas, procesos, validaciones y verificaciones que el negocio necesita cumplir. Dentro las aplicaciones *web* esta capa está compuesta de archivos alojados en el servidor.

- Capa de datos: como su nombre lo indica, esta capa es la encargada de la gestión de datos. Normalmente, está compuesta por una o varias bases de datos. En aplicaciones empresariales muy grandes se suele extender el modelo cliente-servidor, tanto del lado del servidor en donde se cuenta tanto con un servidor *web*, como con un servidor de aplicación y un servidor de base de datos.

En la búsqueda del desacoplamiento y mejor manejo de las aplicaciones, han surgido nuevas capas para crear las aplicaciones de n-capas. Estas buscan poner capas intermedias entre las tres capas antes mencionadas para mejorar algún aspecto de la aplicación *web* que sea de importancia para los usuarios, o bien satisfacer un requerimiento de la aplicación. Como ejemplo de las capas intermedias se pueden mencionar:

- Capa de persistencia de datos: normalmente compuesta por una herramienta llamada ORM, la cual permite utilizar objetos de la capa de datos mediante objetos creados en la capa de lógica del negocio, creando de esta forma una especie de interfaz de comunicación entre la capa del negocio y la capa de datos.

- Capa de servicios: compuesta por una serie de funciones comunes hacia toda la aplicación *web*, que pueden ser consumidas en cualquier momento por cualquier componente que lo necesite. Esta capa forma parte de la capa de lógica del negocio.

1.2. Aplicaciones enriquecidas para internet (RIA)

Las Aplicaciones Enriquecidas para Internet son aplicaciones *web* que tienen como objetivo brindar al usuario las características de las aplicaciones de escritorio, de tal manera que el usuario pueda tener una mejor experiencia de uso y una adaptación a las aplicaciones *web* mucho más rápida y sencilla.

Las Aplicaciones Enriquecidas para Internet proveen de una infraestructura que permite el acceso a multimedia y a datos, como si se tratara de una aplicación de escritorio, esto es posible gracias al uso de *frameworks* RIA que son instalados en los navegadores de Internet en forma de *plug-in*, *sandboxes* o máquinas virtuales. Estas instalaciones son las encargadas de funcionar como interfaces entre el cliente o navegador de Internet y el proveedor de datos o servidor.

A diferencia de las aplicaciones *web*, una de las características más relevantes de las Aplicaciones Enriquecidas para Internet, es que estas no proveen únicamente de páginas *web* basadas en HTML al cliente, sino en vez de esto proveen documentos *web* que contienen lógica del negocio dentro de estos, y sobre la cual el documento *web* puede realizar ciertos procesos sin establecer comunicación con el servidor.

No existe una definición concreta y puntual para las RIA, ya que comúnmente son definidas a través de sus características, sin embargo existe una definición descrita por Marianne Busch y Nora Koch de la Universidad de *Ludwig-Maximilians-Universität* de Munich, Alemania, que es una descripción técnica respecto de lo que es una Aplicación Enriquecida para Internet y la descripción de lo que el usuario encontrará, utilizando una de estas aplicaciones.

"Las Aplicaciones Enriquecidas para Internet son aplicaciones *web*, las cuales utilizan datos que pueden ser procesados tanto por el servidor como por el cliente. Además, el intercambio de datos toma lugar de una forma asíncrona de tal forma que el cliente puede quedar en espera de una respuesta del servidor mientras que continuamente calcula o actualiza partes de la interfaz de usuario. Del lado del cliente, las Aplicaciones Enriquecidas para Internet proveen un *look-and-feel* similar a una aplicación de escritorio y la palabra 'Enriquecida' significa particularmente, las diferencias entre la generación anterior de aplicaciones *web*." [4]

Las RIA son caracterizadas básicamente por una variedad de interacciones con controles de operación, la posibilidad del uso de las aplicaciones en línea o fuera de línea y el uso transparente del poder de procesamiento que existe del lado del cliente y del lado del servidor y la conexión a las redes de computadoras.

De esta definición se puede hacer énfasis en que el poder de procesamiento de los datos y de la aplicación, se encuentra divido tanto del lado del cliente como del lado del servidor, lo cual representa una ventaja para la carga de trabajo de los servidores *web*, ya que parte de esta es transferida al cliente. Además, se encuentra el uso de la comunicación asíncrona, que es el comportamiento que permite a las Aplicaciones Enriquecidas para Internet brindar una mejor experiencia de usuario, permitiendo realizar acciones del lado del cliente, mientras se envía una solicitud al servidor y se espera por la respuesta.

[4] BUSCH, Marianne, KOCH, Nora. *Rich Internet Applications State-of-the-Art.*

1.2.1. Historia

Las aplicaciones *web* dieron un nuevo giro en cuanto al uso del *software* como una herramienta, debido a que mediante el uso de la infraestructura de las redes de computadoras mostraron las bondades que podían ofrecer a los usuarios, como la descentralización de la información, la fácil actualización de los sistemas informáticos, la accesibilidad a la aplicación y a la información que esta posee desde distintos puntos de acceso o bien en la mayoría de casos, el acceso multiplataforma que estas permiten.

En función de estas bondades dio inicio la búsqueda de formas para poder ofrecer a los usuarios una mejor experiencia de uso, en donde el proceso para realizar actividades a través de las aplicaciones *web* no fuera lento, que las interfaces de usuario fueran más amigables e intuitivas y que se pudiera optimizar la transmisión de datos entre el cliente y el servidor.

En 1999 Microsoft buscó la forma de crear aplicaciones *web* que suplieran las necesidades de los usuarios, e introdujo la tecnología *remote scripting* para las aplicaciones *web* basadas en ASP.[5]

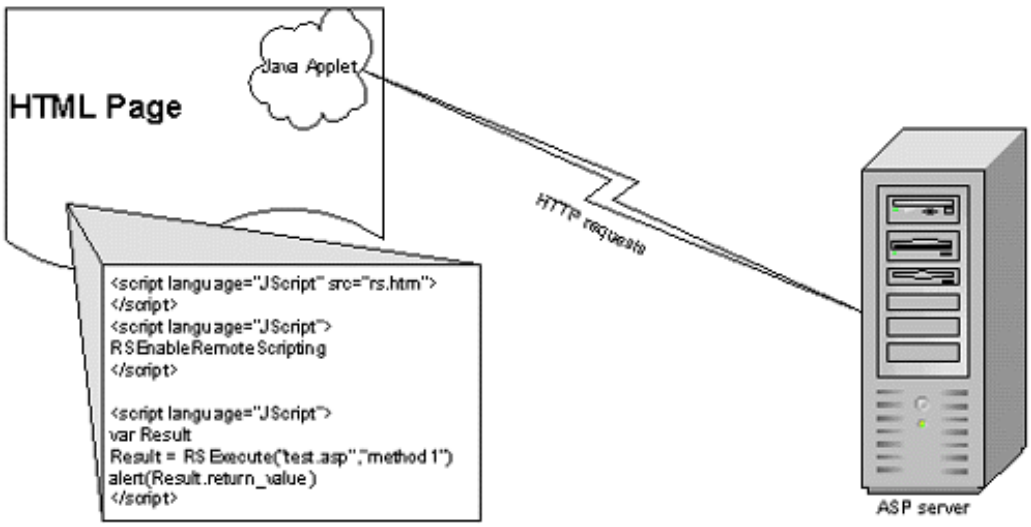
El *remote scripting* hace uso de llamadas síncronas y asíncronas al servidor *web*. Son las llamadas asíncronas las que permiten dar una mejor experiencia de usuario y optimizar el desempeño de las aplicaciones, ya que no es necesario enviar completamente la página *web* al servidor, esperar a que este procese la información, genere nuevamente el código HTML de la página y lo retorne al cliente, para que sea interpretado por el navegador *web*, sino que

[5] CLINICK, Andrew. *Remote Scripting*. <http://msdn.microsoft.com/en-us/library/ms974566.aspx>.

en vez de esto realiza la llamada a una función ASP en el servidor mediante el uso de una *Applet* de Java que se ejecuta del lado del cliente.

Este envío de la solicitud a través del *Applet* de Java es interpretado por el servidor *web* como cualquier otra solicitud HTTP, por lo que la procesa y la responde, no retornando el código HTML sino únicamente el valor del resultado de la ejecución de la función que se deseaba ejecutar. Esta información es devuelta en formato XML y es transformada en un objeto JavaScript del lado del cliente. Es la capacidad de la ejecución de llamadas asíncronas al servidor lo que permite brindar interfaces de usuario más poderosas teniendo como principal beneficio el no congelar la pantalla de la interfaz de usuario en espera de una respuesta por parte del servidor. Su funcionamiento es representado en la figura 7.

Figura 7. **Funcionamiento del *remote scripting* de Microsoft**



Fuente: http://msdn.microsoft.com/en-us/library/ms974566.aspx. 16 de marzo de 2011.

El *remote scripting* tenía las siguientes desventajas:

- Era únicamente accesible para páginas ASP, y estaba disponible únicamente para el navegador de Internet de Microsoft Internet Explorer versiones 4.0 y 5.0, además de Netscape, lo cual representaba una limitante al desear crear aplicaciones *cross-browser.*

- Necesitaba de las librerías de JScript de Microsoft y también de la máquina virtual de Java, para la ejecución del *Applet* encargado de la comunicación con el servidor.

En mayo de 2001, la compañía Forrester Research, dedicada a la investigación de mercados y tecnología independiente, publicó una nota de prensa en la cual se predecía el fin de los sitios *web* y de la Internet si no se realizaba un cambio que era necesario para los usuarios.[6]

En esta nota de prensa, Forrester Research indicaba que la *web* en ese momento no brindaba ningún beneficio a los usuarios, ya que las páginas estáticas que se publicaban con información del clima, del tipo de cambio o de la bolsa de valores, se referían a la misma información que podía ser consultada en cualquier medio de noticias impreso; además, que la actualización de la información de los sitios que la presentaban no era automática, sino que era necesario que una persona la actualizara, tal como en los periódicos; por lo que no se encontraba ningún valor agregado en el uso de la *web*.

[6] ZAPF, Mariko. *The Death Of The Web Is Inevitable, According To Forrester Research*. <http://www.forrester.com/ER/Press/Release/0,1769,567,00.html>.

En esta misma nota de prensa se planteaba el concepto de X-Internet (*Extended Internet*) el cual abarca dos áreas principales:

- La primera de ellas es el contar con una evolución en las páginas *web* que se encontraban en ese momento, para que estas brindaran información en tiempo real al usuario; que pudieran ofrecer información actualizada a los usuarios en el momento en que estuviera ocurriendo, así como crear una interfaz de usuario útil y llamativa, esto a través de aplicaciones "desechables" que fueran descargadas a sus computadoras o dispositivos portátiles y que posterior a su uso se desecharan.

- La segunda área es el acceso a Internet desde dispositivos electrónicos que utilizaran electricidad, a través de microprocesadores de bajo costo que permitieran la conexión tanto alámbrica como inalámbrica.

En el 2002, Macromedia, quien fue comprada en el 2005 por Adobe, utilizó por primera vez el término de Aplicaciones Enriquecidas para Internet (Rich Internet Applications) en el ensayo "*Macromedia Flash MX – A next-generation rich client*"[7]. A través de este ensayo, el autor establece las características que debe de tener un Cliente Enriquecido (*Rich Client*) y una Aplicación Enriquecida para Internet, de modo que pueda satisfacer las necesidades de los usuarios, y respecto a cómo la herramienta Flash provee de los medios para llevarlo a cabo.

Es a partir de la creación de Flash, en donde Macromedia Flash empieza a ser la herramienta preferida para la construcción de Aplicaciones Enriquecidas para Internet, que provee de un lenguaje dinámico para la creación de dichas aplicaciones, lo cual le brinda gran ventaja sobre el lenguaje

[7] ALLAIRE, Jeremy. *Macromedia Flash MX-A next-generation rich client.*

HTML ya que este último estaba diseñado únicamente para presentar información en los navegadores de Internet mediante páginas estáticas, las cuales a pesar de hacer uso del *scripting* en ese momento, aún se encontraban con algunos problemas debido a la compatibilidad *cross-browser*. Macromedia Flash no tenía este problema debido al *plug-in* que utilizaba en los navegadores de Internet.

Las características con las que debe de cumplir una aplicación *web* para poder ser catalogada como una Aplicación Enriquecida para Internet son las siguientes:

- Proveer un medio de ejecución eficiente y de alto desempeño para la ejecución de código, contenido y comunicación. Debido a que el lenguaje HTML no proveía ninguna forma de ejecutar código en tiempo real, mostrar contenido multimedia o la capacidad de almacenar datos del lado del cliente, se establece como una de las características que debe de ofrecer las Aplicaciones Enriquecidas para Internet.

- Integrar contenido, comunicación y las interfaces de la aplicación de un ambiente común. Debido a la falta de integración de medios para la presentación de información basada en HTML que pudieran ser mostrados en los navegadores de Internet, las distintas interfaces de comunicación existentes y la dependencia de los reproductores de multimedia que se tuvieran instalados en la computadora cliente para la ejecución de dicho contenido, demuestran que para que el usuario tenga una mejor experiencia de uso, estas características deben de estar integradas a la aplicación.

- Proveer de modelos de objetos fuertes y extensibles para la interactividad. Esto es proveer una mejora al *scripting* y el modelo de objetos DOM y DHTML, ya que estas tecnologías no son lo suficientemente enriquecidas para ofrecer verdaderas Aplicaciones Enriquecidas para Internet.

- Permitir el rápido desarrollo de aplicaciones a través de componentes y la reutilización de las mismas y el uso de *frameworks* de trabajo, para que los desarrolladores puedan construir herramientas rápidamente.

- Permitir el uso de servicios *web* y de datos proveídos por los servidores de aplicación. Satisfacer la necesidad de la separación de la capa de presentación, debido a que al utilizar una interfaz exclusiva para los datos, se optimiza también la transferencia de información entre ambas partes.

- Adoptar los clientes conectados y desconectados. Las Aplicaciones Enriquecidas para Internet deben de permitir su funcionamiento conectado a una red y desconectado de ella.

- Permitir la distribución de forma sencilla para distintas plataformas y dispositivos. Aprovechar el beneficio primario de las aplicaciones *web*, el cual es el alcance que estas brindan, de forma que las aplicaciones puedan ser accedidas y actualizadas de forma transparente al usuario.

1.2.2. Arquitectura

Las Aplicaciones Enriquecidas para Internet utilizan una estructura basada en la optimización de recursos tanto del lado del servidor *web*, como del cliente y de la conexión de red; esto con el fin de brindar al usuario una mejor

experiencia de uso a través de las interfaces gráficas enriquecidas que estas aplicaciones ofrecen, así como la disponibilidad que una aplicación *web* ofrece.

Existen distintas herramientas para el desarrollo de este tipo de aplicaciones, algunas son nuevas herramientas y otras con algunos años de existir, que han sido combinadas para ofrecer una innovación evolutiva.

Para algunos autores[8] utilizar herramientas que no son nuevas para la construcción de RIAs no es más que el uso de tecnologías viejas, como JavaScript, de una forma distinta para crear un nuevo comportamiento, opinión que actualmente se encuentra sin fundamentos debido al reciente lanzamiento de HTML5, el cual permite crear aplicaciones que ofrecen contenido multimedia sin depender de un reproductor de video, o un reproductor de música específico, así como una interfaz de usuario atractiva e intuitiva sin necesidad de herramientas como Flash; de igual manera HTML5 seguirá utilizando las herramientas como JavaScript y CSS para ofrecer una experiencia completa al usuario, creando verdaderas Aplicaciones Enriquecidas para Internet.

1.2.2.1. Comunicación asíncrona

Una de las características de las Aplicaciones Enriquecidas para Internet que la diferencia de las demás aplicaciones *web,* es el uso de la comunicación asíncrona con el servidor *web*.

En las aplicaciones *web* que no utilizan tecnologías RIA, al momento de realizar una acción en una página *web* es enviado un *request* utilizando el protocolo HTTP, al servidor *web*. Este es recibido y procesado para,

[8] NASCIMBENE, Carlos. *¿Qué son las Rich Internet Applications?*. <http://www.canal-ar.com.ar/noticias/noticiamuestra.asp?Id=2639>.

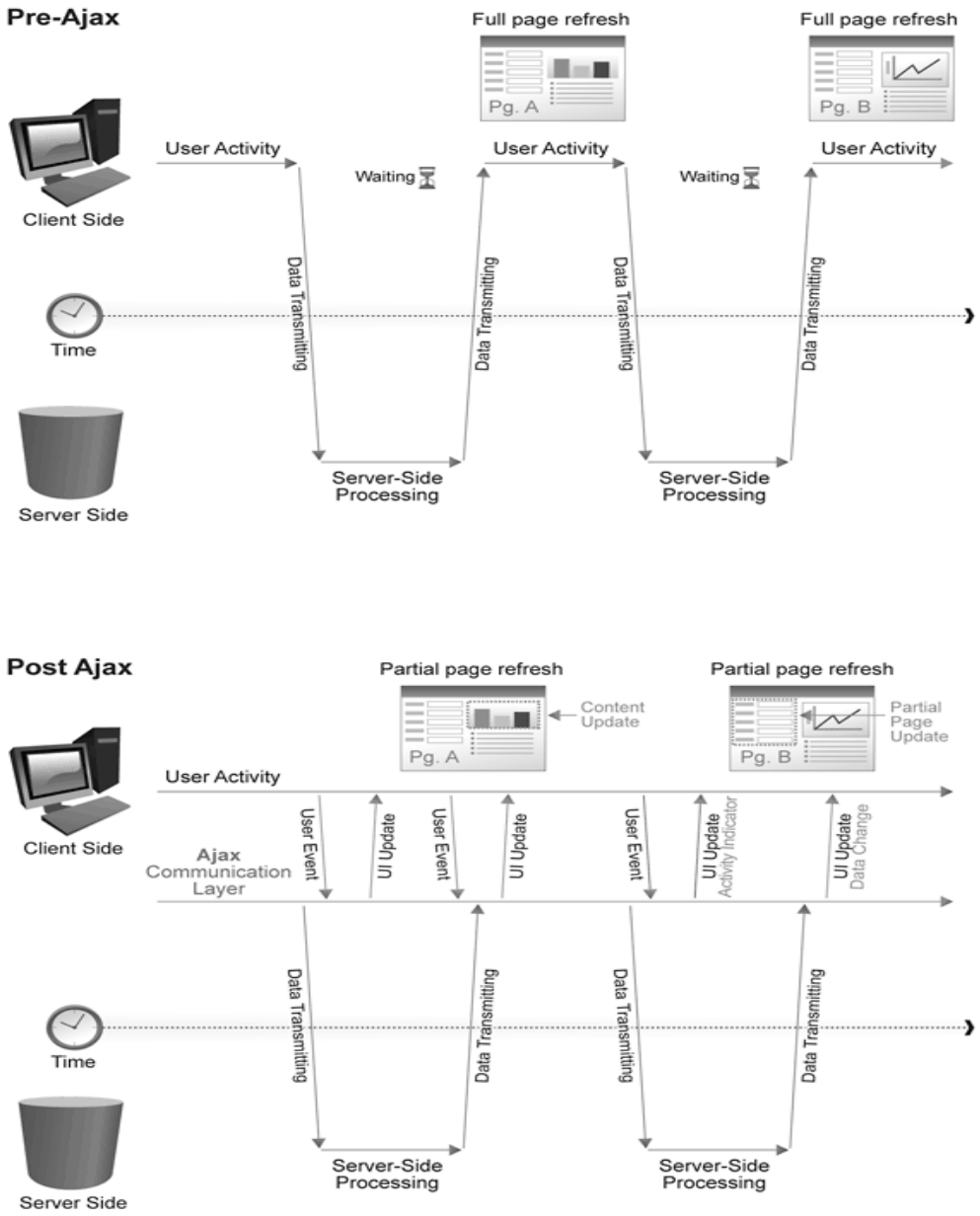
posteriormente, ser respondido mediante un *response*, retornando una nueva página *web*, completando así una transacción HTTP. Esto implica que al realizar una petición al servidor *web,* la interfaz de usuario queda inhabilitada en espera de la respuesta del servidor *web*.

La comunicación asíncrona con el servidor *web* que proveen las RIA, se logra debido al encapsulamiento de la comunicación en un componente de transacción e inserción de objetos (*Transaction & Embedding Component*), el cual es el encargado del envío y recepción de las transacciones HTTP realizadas por la aplicación *web*, adquiriendo completamente esta responsabilidad de la interfaz de usuario y evitando de esta forma que la página *web* sea actualizada en su totalidad.

La comunicación entre el componente y el servidor *web* seguirá siendo una comunicación síncrona, pero la comunicación entre la interfaz de usuario y el componente será asíncrona.

Para lograr la comunicación asíncrona entre la aplicación y el servidor *web,* se hace uso de la instrucción XMLHttpRequest del protocolo HTTP, el cual es implementado nativamente en los navegadores *web*. En la figura 8, se muestra un ejemplo comparando la comunicación síncrona con la comunicación asíncrona, utilizando la tecnología AJAX.

Figura 8. **Comparación de comunicación asíncrona utilizando AJAX con la comunicación síncrona**



Fuente: http://www.websiteoptimization.com/secrets/ajax/8-1-ajax-pattern.png. 05 de abril de 2011.

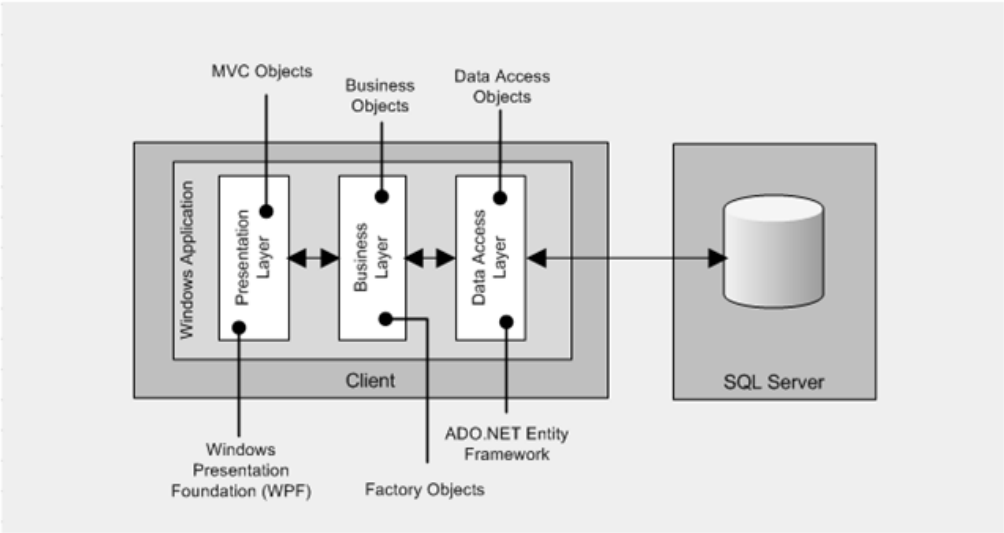
1.2.2.2. Rich clients

Las aplicaciones que se encuentran en redes de computadoras se basan comúnmente en una arquitectura cliente-servidor; pero el cliente toma distintas responsabilidades para las Aplicaciones Enriquecidas para Internet. En las aplicaciones *web*, el cliente tiene la única responsabilidad de recibir los datos ingresados por el usuario y enviarlos al servidor, así como recibir las respuestas del servidor, interpretar el código HTML y mostrar la respuesta al cliente. Las Aplicaciones Enriquecidas para Internet hacen uso de *Rich Clients* (clientes enriquecidos).

Los clientes enriquecidos toman más responsabilidades, las cuales son definidas por los creadores de la aplicación. Lo que le permite el poder tomar estas responsabilidades es la capacidad que tienen de mantener información en la memoria del navegador de Internet para que pueda ser procesada posteriormente por medio de *scripting*, por ejemplo realizar validaciones, cálculos con la información que ha ingresado el usuario, carga automática de filtros o pagineo de resultados de información, esto sin necesidad de establecer comunicación con el servidor.

En la figura 9, se muestra un ejemplo sobre cómo puede estar estructurado un cliente enriquecido. En este caso el cliente enriquecido se encuentra construido sobre tecnologías Microsoft, por lo que se puede apreciar que del lado del cliente se tendrá la capa de presentación, la capa de negocio y la capa de acceso a datos, dejando fuera de la responsabilidad del cliente únicamente la capa de datos. Este es un modelo que podría ser fácilmente adoptado también por tecnologías de Adobe.

Figura 9. **Estructura de un cliente enriquecido**



Fuente: http://blogs.msdn.com/b/jmeier/archive/2009/01/21/rich-client-application-pattern.aspx. 27 de marzo de 2011.

El uso de los clientes enriquecidos ayudan a la disminución del tráfico de red, por lo que originalmente fue muy utilizado en redes que contaban con un ancho de banda muy bajo, ya que los únicos datos enviados y recibidos son los de carácter estrictamente necesario, optimizando los mensajes que eran enviados a través de la red.

Cuando se inició el movimiento de la creación de clientes enriquecidos, se les llamaba de esta forma a las aplicaciones que son instaladas en los clientes y que accesan a la información provista por un servidor, como por ejemplo los clientes de bases de datos[9].

[9] Bright Software Pty. Ltd. *Rich Clients vs. Thin Clients.*

Algunas de las ventajas que proveen los clientes enriquecidos son:

- Menor cantidad de solicitudes a los servidores *web*
- Capacidad de trabajar con la aplicación cuando está fuera de línea
- Mejoras notables en cuanto al desempeño con contenido multimedia
- Permite aumentar el rendimiento del servidor *web*

1.2.2.3. Aislamiento de procesos

Con el paradigma de las Aplicaciones Enriquecidas para Internet se inició la creación de herramientas creadas específicamente para el desarrollo de las aplicaciones RIA; entre estas se puede mencionar Adobe Flash, Microsoft Silverlight y Oracle JavaFX. Estas herramientas proveen de la estructura necesaria para brindar a los usuarios una interfaz enriquecida, que cumpla con el propósito de brindar al usuario un ambiente similar al de las aplicaciones de escritorio y la capacidad de reproducir contenido multimedia, haciendo uso únicamente de las mismos componentes que forman parte de la herramienta, sin necesidad de hacer uso de *software* externo, es decir de *software* instalado en el cliente.

Conforme al uso de las herramientas para RIAs se encontró que estas poseen ciertas vulnerabilidades que pueden ser utilizadas para perjudicar al usuario, por lo que los proveedores de las herramientas para desarrollar RIAs implementaron un medio por el cual se disminuye significativamente el efecto de utilizar las vulnerabilidades de cada una de las herramientas dentro del sistema operativo del cliente, de forma que la vulnerabilidad no pueda afectar significativamente el sistema operativo cliente.

Durante el tiempo que tome la corrección de la vulnerabilidad detectada, el medio utilizado para proteger el sistema operativo cliente es el aislamiento de procesos (*Sandbox*).

El aislamiento de procesos consiste en contextos de ejecución de un programa cliente dentro del navegador *web* que permite el acceso, únicamente a una parte de los recursos del sistema operativo, como pueden ser memoria RAM y disco duro, esto con el fin de no proveer acceso a todo el sistema operativo, sino únicamente a un área restringida ya que si en algún momento se intentara explotar una vulnerabilidad de la aplicación RIA, el *sandbox* no le permitirá acceder más allá de los recursos que tiene asignados la aplicación.

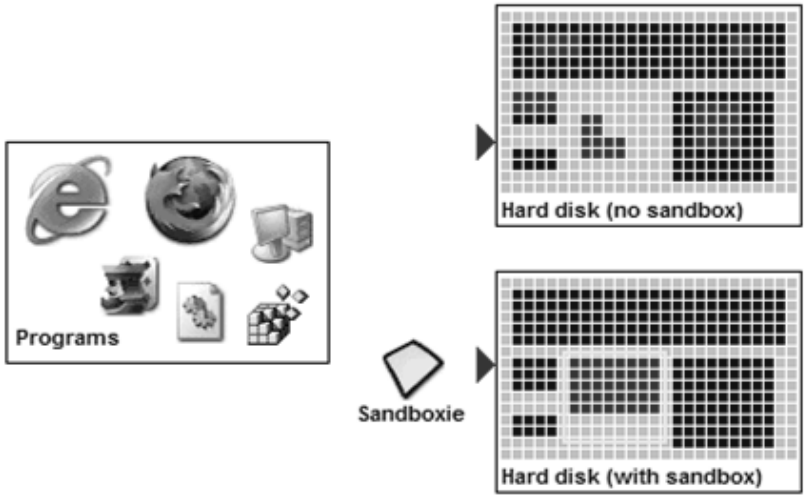
Tanto Adobe Flash, Microsoft Silverlight y Oracle JavaFX han creado por sus propios medios, estructuras de aislamiento de procesos, esto con el fin de brindar mayor seguridad y confiabilidad sobre las aplicaciones desarrolladas con estas tecnologías para los usuarios de las mismas.

En la figura 10 se puede apreciar cómo funciona el aislamiento de procesos en un disco duro.

En el lado izquierdo de la imagen están los programas, los cuales pueden ser *software*, ejecutado en un navegador *web* como Microsoft Internet Explorer o bien Mozilla Firefox. En el cuadro que se encuentra en la esquina superior derecha de la imagen se muestran los cuadros de color más oscuro, los cuales son sectores ocupados del disco duro; los cuadros de color gris claro son sectores libres del disco duro y los de tono gris intermedio, son los sectores escritos por la aplicación *web*. En el cuadro inferior derecho se encuentra la misma codificación de colores para indicar el estado de los sectores, pero se

encuentra también una serie de cuadros blancos formando un cuadro, lo cual representa el *sandbox*.

Figura 10. **Funcionamiento del aislamiento de procesos en un disco duro**



Fuente: http://3.bp.blogspot.com/_7sYlkd_i4DY/TRt83Hu5yNI/AAAAAAAAAro/1pnAkWshme0/s1600/Sandboxie%2Bv3.51.08%2BBeta%2BMultiLanguage.gif. 27 de marzo de 2011.

Como se puede observar, cuando no existe el aislamiento de procesos, la aplicación tiene acceso a cualquier parte del disco duro, por lo que puede leer y escribir en donde desee; dando así un acceso libre al sistema operativo cliente, permitiéndole explotar vulnerabilidades del sistema.

En el caso del aislamiento de procesos, la aplicación *web* únicamente tiene acceso al sector delimitado y establecido que le ha sido asignado, sin permitir el acceso al resto del disco duro para que no exploten vulnerabilidades del sistema.

1.2.2.4. *Plug-in*

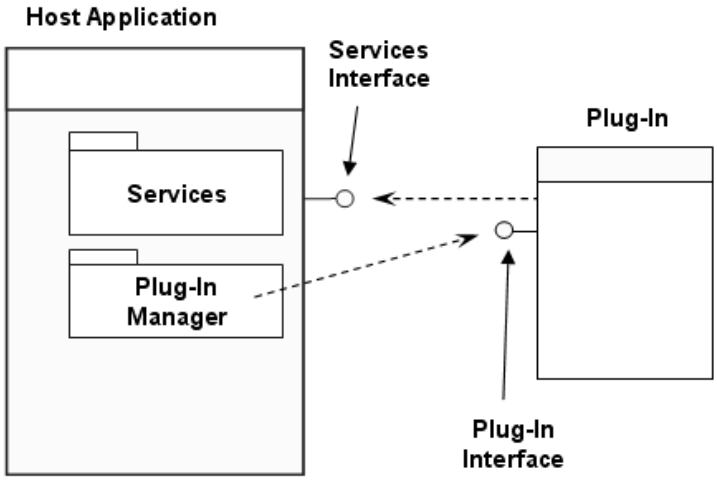
Para poder crear las interfaces de usuario enriquecidas, era necesario realizar una extensión a las capacidades de los navegadores de Internet, debido a que estos únicamente pueden interpretar código HTML y ejecutar *scripts*; pero esto no basta para las necesidades de las Aplicaciones Enriquecidas para Internet.

Para permitir a los navegadores *web* que pudieran brindar interfaces de usuario enriquecidas, se hizo uso de los *plug-in*. Los *plug-in* son pequeños componentes de *software* que se comunican con un *software* más grande, en este caso el navegador *web*. Esta comunicación es posible gracias al uso de interfaces comunes, las cuales sirven para enviar y recibir instrucciones entre sí. Estos componentes de *software* extienden el funcionamiento del navegador *web* proveyendo de nuevas funcionalidades, las cuales son necesarias para las Aplicaciones Enriquecidas para Internet.

Entre los *plug-in* más comunes se puede mencionar Adobe Flash Player *plug-in*, Microsoft Silverlight *plug-in* y el JavaFX *plug-in*. Cabe mencionar que no todas las herramientas para crear Aplicaciones Enriquecidas para Internet hacen uso de *plug-in* para poder funcionar, siendo el caso más relevante el HTML5, el cual puede ser ejecutado sobre cualquier navegador *web* de forma nativa que posea soporte para HTML5, sin necesidad de instalar ningún *plug-in*.

En la figura 11 se puede ver representada la forma en que se comunica el *software* principal, en este caso identificado como *Host Application* con el *plug-in*. El *software* principal posee un administrador de *plug-in* el cual presenta una interface para que el mismo pueda comunicarse con el *software* principal, y viceversa.

Figura 11. **Comunicación de un *plug-in* con una aplicación**



Fuente: http://upload.wikimedia.org/wikipedia/en/4/46/Plug-InExample.png. 29 de marzo de 2011.

1.2.3. Aplicaciones *web* vs. Aplicaciones enriquecidas para internet

Para la creación de aplicaciones *web* que buscan aprovechar los beneficios que ofrecen las redes de computadoras, es básico conocer cuándo realmente es necesario hacer uso del paradigma de las Aplicaciones Enriquecidas para Internet. Para esto se debe conocer cuáles son las características de cada paradigma; esta es la base sobre la que se tomará la decisión respecto de qué paradigma utilizar para satisfacer las necesidades de los usuarios.

En el desarrollo de *software* es común que los requerimientos no sean claros, ya sea porque no se obtienen directamente del usuario final, o la persona que recopila dichos requerimientos no los orienta a lo realmente

necesario. Esto ocasiona que se entregue como producto final, un *software* muy completo pero que no satisface las necesidades del usuario, como por ejemplo cuando se necesita solucionar un problema y la solución óptima sugiere utilizar una aplicación *web*, pero en vez de esto se crea una Aplicación Enriquecida para Internet, la cual puede ofrecer mucha funcionalidad, pero no resuelve eficientemente el problema inicial.

Las aplicaciones *web*, como ya se ha explicado, presentan información mediante HTML y *scripting*, que funcionan en su mayoría de manera síncrona y que necesitan recargar toda la interfaz del usuario para poder presentar la respuesta del servidor al usuario.

Las Aplicaciones Enriquecidas para Internet proveen una gran capacidad multimedia, utilizan comunicación asíncrona con el servidor *web* y *rich clients*, y únicamente entablan comunicación con el servidor cuando se necesita realizar alguna gestión de datos.

Al analizar ambos paradigmas en cuanto a desempeño, se puede concluir que las Aplicaciones Enriquecidas para Internet hacen uso más eficiente del ancho de banda de la red, ya que transmiten únicamente la información necesaria para la aplicación *web*; no es necesario que sea enviada información que no es de utilidad y que en la respuesta de la aplicación retorne la misma sin haber sufrido algún cambio.

Una de las grandes facilidades que proveen las aplicaciones *web* es liberar la carga de procesamiento del cliente y concentrarlas en el servidor, lo cual al inicio del paradigma de las aplicaciones *web* era una ventaja que tenían respecto de los primeros clientes del modelo cliente-servidor.

Con el tiempo esto ha cambiado, debido a que las capacidades de la Internet y de las redes de computadoras se han ido expandiendo, proveyendo de una mejor infraestructura para las aplicaciones *web* y permitiendo así el acceso de más usuarios de tal forma que las capacidades de procesamiento, memoria y disco duro de los servidores han crecido exponencialmente.

Un ejemplo de esto es la Ley de Moore, la cual indica que cada dieciocho meses se duplicará la capacidad de transistores existentes dentro del microprocesador más potente del día de hoy; por lo que las ventajas que proveían las aplicaciones *web* originalmente, han empezado a representar un costo, debido a que todo el procesamiento de las aplicaciones está concentrado en el servidor *web*; esto es resuelto por las Aplicaciones Enriquecidas para Internet, puesto que en este paradigma se busca liberar un poco de la carga de procesamiento a los servidores *web*.

A pesar de las capacidades que proveen las Aplicaciones Enriquecidas para Internet, existe un área en la que este paradigma provee algunas inconformidades, entre las cuales se encuentra la estandarización. Como ya se ha comentado el paradigma las Aplicaciones Enriquecidas para Internet provocó la creación de *frameworks* de desarrollo de RIAs, los cuales necesitan de la instalación de *plug-ins* en los navegadores *web*. Esto implica que el cliente debe de instalar dicho componente en el navegador *web*, pero en algunas ocasiones no es tan sencillo como parece.

En el 2009 Apple anunció que el iPhone 3G no soportaría el *plug-in* de Flash[10], lo cual representaba la pérdida del mercado de usuarios que utilizaban este dispositivo móvil para poder ser utilizado en RIAs. Según declaraciones, la

[10] IONESCU, Daniel. *3 Reasons Why iPhone Won't Get Adobe Flash*. <http://www.pcworld.com/article/173092/3_reasons_why_iphone_wont_get_adobe_flash.html>.

razón era que las capacidades del iPhone no permitían la ejecución de dicho *plug-in*. Posteriormente, se comentó respecto de las posibilidades que Apple quisiera implementar un nuevo estándar para el sistema operativo del iPhone.

De esta cuenta, se puede indicar que existe la posibilidad que algunos *plug-ins* no se encuentren disponibles para ser instalados en algún navegador *web*, limitando así el acceso a la aplicación *web*.

Este tipo de inconvenientes se ha presentado en las aplicaciones *web* desde su inicio, y fue este desacuerdo entre estándares lo que provocó la creación del W3C (*World Wide Web Consortium*), para poder crear y regular la estandarización de HTML, JavaScript y CSS, de forma que se puedan crear aplicaciones *cross-browser*, aunque esta organización ha funcionado hasta el momento como una guía para la creación de las librerías que son utilizadas en los distintos navegadores.

Uno de los ejemplos que está ocurriendo actualmente, es que con la aparición del HTML5 y el CSS3, se encuentran en desarrollo las librerías de CSS3 para la animación de figuras en ambientes 3D. Para esto *Web Kit* ha desarrollado las librerías necesarias, las cuales al ser utilizadas en navegadores como Google Chrome o Apple Safari, son accesibles a través de la instrucción

–web-kit [propiedad]

Mientras que en los navegadores Mozilla, como Firefox, estas funcionalidades son accesibles a través de la instrucción

-moz-kit [propiedad]

Esto indica que inclusive para tecnologías y lenguajes que tienen mucho más tiempo de existir que los *plug-in* que se utilizan para ejecutar las Aplicaciones Enriquecidas para Internet aún existe una falta de estandarización.

A pesar de esto el uso de las librerías de CSS3 para la creación de animaciones de figuras en 3D, son funcionalidades que pueden ser utilizadas en todos los navegadores *web* sin necesidad de la instalar ningún *plug-in*, lo cual puede llegar a representar una ventaja respecto de los *frameworks* para la construcción de las Aplicaciones Enriquecidas para Internet.

1.2.4. Ventajas y desventajas de las aplicaciones enriquecidas para internet

Las Aplicaciones Enriquecidas para Internet ofrecen una serie de ventajas sobre las aplicaciones *web*, pero como toda tecnología y como todo paradigma, estas poseen ciertas características que pueden representar una desventaja, por lo que es necesario conocer cuáles son y evaluar si el paradigma es el adecuado para ser implementado.

A lo largo del capítulo se han mostrado cuáles son las capacidades que tienen las Aplicaciones Enriquecidas para Internet, pero es necesario conocer en dónde se encuentran sus desventajas, ya que solamente conociendo el estado real de este paradigma se podrá discernir respecto del uso de este paradigma para la construcción de una aplicación *web*.

1.2.4.1. Ventajas

- Mejora del consumo de ancho de banda para la comunicación con el servidor *web*. Esta es una de las ventajas más importantes de las

Aplicaciones Enriquecidas para Internet, ya que utiliza el ancho de banda de la red de forma óptima, puesto que únicamente se envían y reciben los paquetes de datos que son completamente necesarios para el funcionamiento de la aplicación, pues existe una parte de la aplicación que ya se encuentra cargada en el navegador *web* del cliente.

- Comunicación asíncrona con el servidor *web*. Esto permite brindar a los usuarios una mejor experiencia de uso que las aplicaciones *web*, debido a que se optimiza el tiempo de uso de la aplicación para el usuario, ya que al realizar una acción dentro de la aplicación que necesite entablar comunicación con el servidor, no congela toda la aplicación, sino que utiliza una interfaz de comunicación transparente al usuario que es la que se encarga de solicitar la información, recibirla y presentarla de forma asíncrona, permitiendo así la optimización de tiempo.

- Amplia mejora en interfaces multimedia. Brinda grandes capacidades multimedia al usuario, permitiendo reproducir *streaming* de datos sin necesidad de utilizar ninguna otra utilidad en la computadora cliente, sino que utiliza reproductores embebidos en la herramienta utilizada para construir la aplicación. Esto permite también brindar al usuario un ambiente familiar y cómodo para realizar el tipo de actividades que se realizan comúnmente en aplicaciones de escritorio.

1.2.4.2. Desventajas

- Tiempo de carga inicial. Con la finalidad de las Aplicaciones Enriquecidas para Internet de disminuir el tiempo de respuesta al usuario, de no presentar pantallas en blanco en espera de la respuesta del servidor *web*, comúnmente estas aplicaciones tienen una carga inicial prolongada, ya

que en la primera solicitud que se realiza al servidor se carga todo el contexto necesario para que la aplicación pueda ser ejecutada en el navegador *web*, esto puede llegar a incluir un alto porcentaje del tamaño total de la aplicación.

- Falta de estandarización. Al igual que con HTML las tecnologías *web* tienen la desventaja que no existe un estándar bien definido o regulado respecto del uso de *plug-in* en el caso de las tecnologías para construcción de RIAs, o de los intérpretes en el caso del HTML. Como se mencionó anteriormente, actualmente Apple en sus dispositivos móviles no soporta la reproducción de aplicaciones Flash, lo cual representa una desventaja en cuanto a la capacidad de la aplicación de ser accesible a los usuarios.

1.3. Aplicación RIA en Intranet o Internet

Las aplicaciones RIA están basadas en un paradigma que provee de muchos beneficios a los usuarios y de un medio por el cual se pueden crear aplicaciones *web* con capacidades impresionantes para la gestión de datos en pequeñas, grandes y medianas empresas para los desarrolladores de *software*. Las RIA cuentan con muchos beneficios, tales como corto tiempo de respuesta en las peticiones de una aplicación y la estandarización del *software* cliente que se conecta a las redes para acceder a ellas.

Las RIA son una excelente opción para la construcción de aplicaciones empresariales, debido a como ya se ha mencionado antes, la capacidad de gestionar datos en la carga de los mismos, a las interfaces de usuario y la forma en que son estructurados y mostrados dichos datos, brindan una experiencia de uso muy similar a la de las aplicaciones de escritorio.

Una aplicación RIA puede ser utilizada de igual forma en Internet que en una Intranet, esto gracias a que ambos ambientes proveen del contexto necesario para que la aplicación pueda ser utilizada por los usuarios.

De igual forma, al poseer las mismas características que una aplicación *web*, no se encuentra restringida a ser utilizada por un tipo de *software* cliente en específico, ni tampoco posee alguna dependencia de este, proveyendo de esta forma una excelente accesibilidad a la aplicación desde cualquier tipo de red.

2. HERRAMIENTAS DE DESARROLLO PARA APLICACIONES RIA

Con el apogeo de las Aplicaciones Enriquecidas para Internet se han desarrollado muchas herramientas para la creación de aplicaciones que utilizan este paradigma, algunas son *frameworks* completos de trabajo que contienen librerías y entornos de desarrollo integrados (IDE, por sus siglas en inglés) y otros son únicamente librerías para la creación de las aplicaciones. Entre las distintas herramientas que existen actualmente se tiene una amplia variedad que se diferencia en ciertos aspectos tales como: funcionalidad, tipo de tecnología que utiliza para presentar la interfaz gráfica al usuario, capacidades para la gestión de datos, comunicación con el servidor o también si es un *software* propietario o gratuito.

En este capítulo se describen algunas herramientas para el desarrollo de RIAs que existen actualmente, tanto propietarias como gratuitas.

2.1. *Software* propietario y *software* gratuito

Existe actualmente una gran variedad de herramientas que permiten la elaboración y desarrollo de Aplicaciones Enriquecidas para Internet, muchas que dominan gran porcentaje del mercado y otras que buscan ser una alternativa diferente; entre esta variedad de herramientas se pueden encontrar *software* propietario y *software* gratuito.

Cada uno de estos tipos de herramientas proveen beneficios y utilidades, que pueden funcionar como diferenciadores, para seleccionar entre un tipo de

software y otro para el desarrollo de RIA. Para seleccionar una herramienta será necesario evaluar distintos factores tales como: precio, soporte que se puede obtener, documentación existente para consulta y la compatibilidad que tienen los lenguajes en los cuales serán escritas las aplicaciones. Además de evaluar estos factores se deberá comprender cuáles son las diferencias entre el *software* propietario y el *software* gratuito para poder realizar la selección de la herramienta correcta.

2.1.1. *Software* propietario

Es todo aquel *software* por el cual se debe de pagar para tener derecho a utilizarlo legalmente. Normalmente este *software* incluye una serie de restricciones que deben de ser acatadas para poder hacer uso de él.

Este tipo de *software* no permite ante la ley el acceso al código fuente (código fuente cerrado), para realizar modificaciones, mejoras o arreglos que el *software* como tal necesite. El usuario tiene únicamente la opción de informar a la empresa que posee los derechos del *software*, sobre cualquiera de los problemas con que se encuentre o mejora que se desee sugerir.

Este tipo de *software* se encuentra protegido por una o varias patentes que le brinda a los creadores de la herramienta todos los derechos sobre el uso, manipulación y distribución.

Como ejemplo de herramientas que pertenecen al grupo de *software* propietario para la creación de RIAs se puede mencionar Adobe Flash Builder y Microsoft Silverlight.

2.1.2. *Software* gratuito

Es el *software* por el cual no se debe de pagar para poder hacer uso de este y que puede ser utilizado durante un tiempo ilimitado. No debe de confundirse el término de *software* gratuito con el término *software* libre, ya que este último se caracteriza por cumplir con cuatro libertades[11] las cuales no necesariamente se cumplen en el *software* gratuito.

En cuanto al uso de licencias, el *software* gratuito puede venir acompañado de una licencia que permite su redistribución, pero que puede no permitir el acceso y modificación al código fuente del mismo, así como prohibir la venta del mismo o el uso comercial.

Como ejemplo de herramientas que pertenecen al grupo de *software* gratuito para la creación de RIAs se puede mencionar Open Laszlo o Adobe Flex SDK entre otros.

2.2. Herramientas de *software*

A continuación se muestra un pequeño conjunto de herramientas para la construcción de Aplicaciones Enriquecidas para Internet, entre las cuales se han seleccionado aquellas que son las más utilizadas para la construcción de RIAs, como es el caso de Adobe Flash Builder, Microsoft Silverlight y Oracle JavaFX, así como otras opciones que se han mantenido en el mercado gracias al potencial que poseen para la creación de RIAs, como es el caso de Mono Moonlight y OpenLaszlo, para finalmente presentar el lanzamiento del HTML5,

[11] Free Software Foundation. *The Free Software Definition.* <http://www.gnu.org/philosophy/free-sw.html>.

el cual es el ya conocido HTML con capacidades para la creación de aplicaciones *web* con contenido enriquecido.

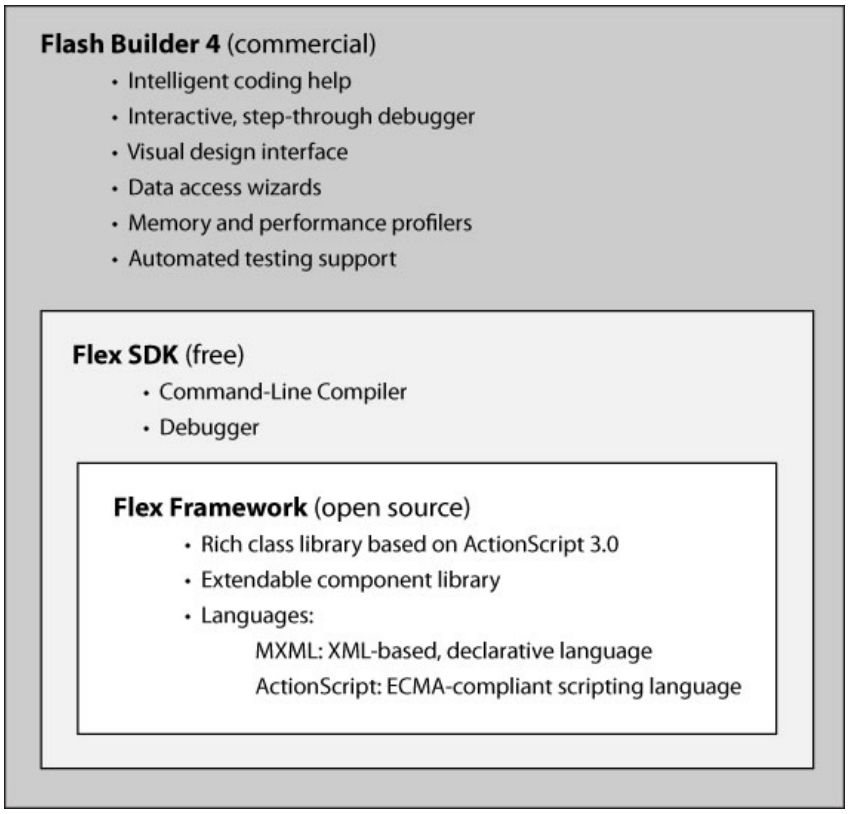
2.2.1. Adobe Flash Builder

La herramienta emblema de la compañía Adobe Flash Builder, la cual es la versión 4 de la herramienta Flex Builder, posee un IDE basado en Eclipse para la construcción rápida de aplicaciones móviles, *web* y de escritorio utilizando ActionScript y MXML, así como el *framework* de código fuente abierto Flex.

El *framework* de Flex provee dos librerías para la creación de interfaces gráficas llamadas Spark y MX. Los componentes MX, los cuales también son conocidos como componentes Halo, fueron introducidos originalmente en Flex 3, cada uno posee una lógica de funcionamiento que permite la integración de su comportamiento con el diseño de la presentación y el *look-and-feel* de la aplicación. Los componentes Spark son nuevos en Flex 4 y han sido creados para separar el comportamiento, el diseño, los estilos y la presentación en la interfaz de usuario. Esta separación permite un mejor control de todos los aspectos de los componentes y mejora la reutilización de componentes.

Actualmente, existen componentes que son equivalentes entre Spark y MX; al momento de la creación de este trabajo de graduación es permitido utilizar los dos tipos de componentes en una misma aplicación sin ningún problema, aunque por parte de Adobe se recomienda utilizar únicamente componentes Spark, ya que son estos últimos los incorporados a Flash Builder 4. En la figura 12 se muestra el diagrama de componentes de la herramienta Flash Builder 4.

Figura 12. **Diagrama de componentes de Flash Builder 4**



Fuente: http://www.adobe.com/devnet/flex/articles/fcf_flex_client_server.html. 17 de abril de 2011.

2.2.1.1. Entorno de desarrollo

El entorno de desarrollo o IDE (*Integrated Development Environment*) de la herramienta Flash Builder se encuentra construido sobre la distribución Java Eclipse. Al adquirir la herramienta, comercialmente esta instala el IDE de Eclipse junto con el *framework* Flex y Adobe AIR.

Se pueden conseguir en Internet otros IDE para el desarrollo con la librería Flex SDK, pero ninguna de estas herramientas provee de la visualización previa

que el IDE propietario de Adobe para Eclipse provee, por lo que el desarrollo utilizando uno de estos IDEs, implica una inversión mayor de tiempo ya que la única forma de ver el resultado gráfico es compilando los cambios realizados y accediendo a la nueva compilación de la aplicación desde un navegador *web*.

Por ser un IDE construido en Eclipse, este posee una gran cantidad de extensiones para ser utilizadas, tales como extensiones para la realización de pruebas unitarias, integración continua, o para desarrollar proyectos que permiten la integración con otros lenguajes de desarrollo, tales como PHP con la extensión PHPEclipse.

2.2.1.2. Actionscript

Es un lenguaje de programación, el cual en conjunto con el lenguaje MXML es utilizado para construir Aplicaciones Enriquecidas para Internet utilizando la herramienta Flash Builder. El lenguaje de programación ActionScript cumple con los estándares establecidos por el ECMA, utilizando una sintaxis similar a la utilizada en JavaScript y Java. En las aplicaciones creadas con Flash Builder, el lenguaje ActionScript es utilizado para escribir la lógica del negocio de la aplicación; posee toda su funcionalidad encapsulada en clases por lo que en su versión estable más reciente llamada ActionScript 3.0, está construida con todas las características para ser clasificada como un lenguaje orientado a objetos.

2.2.1.3. MXML

Es un lenguaje de etiquetas para la creación de interfaces de usuario basado en XML, fue creado por Macromedia en marzo de 2004. Las letras que conforman su nombre significan *Macromedia eXtensible Markup Language*.

Este lenguaje es utilizado junto con ActionScript y Adobe Flex para la creación de Aplicaciones Enriquecidas para Internet.

Las capacidades del lenguaje MXML le permiten la creación de la interfaz de usuario además de la escritura de código fuente ActionScript, por medio de una etiqueta especial. El código fuente, escrito en la capa de presentación de la aplicación, permite realizar validaciones de la información obtenida por la interfaz de usuario y escribir código fuente que contenga lógica del negocio.

El lenguaje MXML al momento de ser compilado por el SDK de Flex, pasa por una fase en que es convertido a lenguaje ActionScript y posteriormente, genera como resultado un archivo binario Flash con extensión SWF; la generación de este archivo binario no depende en ningún momento del IDE sino del SDK. El archivo binario que se obtiene como resultado es el que contiene dentro de sí la interfaz gráfica junto con la lógica del negocio, implícita a la capa de presentación de la aplicación.

2.2.1.4. Flash Player

Flash Player es un reproductor de multimedia y de aplicaciones que pueden ser ejecutadas desde un navegador de Internet. Flash Player fue creado por Macromedia y actualmente es desarrollado y distribuido por Adobe. Flash Player ejecuta archivos SWF que son creados mediante la herramienta Adobe Flash o Adobe Flash Builder.

No se debe de confundir el Flash Player con la herramienta Adobe Flash, ya que la herramienta Adobe Flash es utilizada para crear gráficas y animaciones en base a vectores, utilizando como lenguaje nativo ActionScript

pudiendo ejecutar también *streaming* de audio y video; y el Flash Player es la máquina virtual utilizada para ejecutar archivos de Flash.

Una de las ventajas que provee el uso de Flash en la Internet, es que todas las gráficas que son reproducidas en él están representadas como vectores, lo cual permite minimizar el tamaño del archivo y de este modo minimiza también el consumo de ancho de banda de una red y el tiempo de carga del archivo desde un navegador de Internet.

Flash Player es instalado en los navegadores de Internet como un *plug-in*, a excepción del navegador Google Chrome, el cual posee soporte para Flash, integrado sin necesidad de hacer uso de dicho *plug-in*.

Actualmente Flash Player se encuentra en su versión 10.2, liberada en febrero de 2011 la cual posee aceleración para la reproducción de video para Internet Explorer 9, cursores de *mouse* nativos, soporte para múltiples pantallas y una ampliación a la reproducción de texto.

2.2.1.5. Adobe Air

Esta herramienta viene incluida dentro del paquete de Adobe Flash Builder. El significado de su nombre se deriva del inglés *Adobe Integrated Runtime*. Es utilizada para construir aplicaciones de escritorio que puedan hacer uso de recursos publicados y accesibles en una red. Su principal fortaleza es que además de poder acceder y consumir los recursos publicados en una red, posee también un mayor acceso a los recursos locales del cliente, por lo que una aplicación desarrollada con Adobe Air es capaz de escribir y leer archivos del sistema operativo cliente.

El poder acceder al sistema de archivos local abre una serie de posibilidades que no es posible lograr con aplicaciones *web*, pero esta ventaja exige tener un excelente manejo de seguridad a nivel de la aplicación, para no permitir la explotación de vulnerabilidades de seguridad en la aplicación.

Actualmente se encuentra en su versión 2.7, la cual fue liberada en junio de 2011, ésta brinda soporte para dispositivos con sistema operativo Android[12].

2.2.2. Microsoft Silverlight

Microsoft Silverlight fue creado para la construcción de aplicaciones empresariales y para brindar una mejor experiencia en el manejo multimedia del usuario. La ejecución de estas aplicaciones se lleva a cabo mediante la instalación de un *plug-in* gratuito, que es soportado por la mayoría de navegadores de Internet que existen actualmente y está respaldado por el .NET *Framework*. Este *plug-in* habilita y expande las capacidades de la experiencia de usuario que se puede ofrecer en la Internet, en aplicaciones de escritorio y en aplicaciones *web*, en línea o fuera de línea.

Al igual que Flash Builder y el Flex SDK, Microsoft Silverlight provee el Silverlight SDK de forma gratuita, por lo que para poder desarrollar aplicaciones con Silverlight basta con un procesador de texto, el SDK de Silverlight y el conocimiento para compilar aplicaciones desde consola.

Silverlight provee de un modo gráfico similar al de *Windows Presentation Foundation*, e integra aspectos de multimedia, gráficos, animación e interactividad en un único ambiente de ejecución. En Silverlight, las interfaces

[12] *Adobe Integrated Runtime*. <http://en.wikipedia.org/wiki/Adobe_Integrated_Runtime>.

de usuario son declaradas en lenguaje XAML, y permiten la creación de aplicaciones RIA de modo más sencillo a través del uso del .NET RIA Services.

Las aplicaciones Silverlight son construidas utilizando lenguajes de la *suite* .NET siendo estos C#, C++ y VB.NET.

Actualmente, se encuentra en versión estable Microsoft Silverlight 4, y ya se encuentra liberado el Microsoft Silverlight 5 Beta.

2.2.2.1. Entorno de desarrollo

Microsoft ofrece dos entornos de desarrollo para la escritura de aplicaciones, utilizando Microsoft Silverlight, estas están orientadas a distintos tipos de usuarios en función de las actividades y las responsabilidades que puedan tener dentro de un proyecto de desarrollo de *software*. Los entornos de desarrollo son los siguientes:

- Microsoft Expression Studio: está creado principalmente para diseñadores *web*, por medio de éste entorno los diseñadores pueden crear elementos visuales para las aplicaciones creadas en Silverlight. También brinda una gran cantidad de opciones sobre los elementos que conforman la interfaz visual de una RIA, tales como el color del formulario o pantalla, tipo de letra, etc.

- Microsoft Visual Studio: es utilizado por Microsoft para el desarrollo de aplicaciones .NET. Los desarrolladores de *software* pueden utilizar este para la creación de aplicaciones Silverlight que requieren de codificación, ya sea como parte de la lógica de negocio o bien como validaciones propias de la interfaz de usuario. Visual Studio permite a los

desarrolladores de *software* crear aplicaciones Silverlight sofisticadas, utilizando lenguajes de la herramienta .NET.

Para determinar el entorno de desarrollo que se va a utilizar, únicamente es necesario definir cuál es la función a realizarse.

Si se necesita de una herramienta que permita la creación de una interfaz de usuario sin utilizar nada de código, es recomendable utilizar Microsoft Expression Studio.

Si se necesita crear una interfaz de usuario que utilice código fuente, es recomendable utilizar Microsoft Visual Studio.

2.2.2.2. XAML

El nombre XAML significa *eXtensible Application Markup Language*. Es un lenguaje declarativo basado en XML creado por Microsoft para describir objetos representados por medio de etiquetas XML. Como ya se ha mencionado, este lenguaje representa objetos visuales que serán presentados y animados en páginas HTML que utilicen Silverlight; el mapeo de objetos es realizado directamente a instancias de objetos del *Common Language Runtime* (CLR) incluyendo sus eventos y atributos.

Dado que XAML utiliza el modelo de programación de .NET *framework*, es posible simplificar la creación de interfaces de usuario para aplicaciones basadas en dicho *framework*, permitiendo la separación de la definición de la interfaz de usuario de la lógica del negocio, utilizando archivos *code-behind*. XAML representa la instancia de objetos, lo cual lo diferencia de otros lenguajes de etiquetas que comúnmente son interpretados y no compilados,

mientras que XAML es compilado. Este es utilizado con el .NET *framework* versiones 3.0 y 4.0.

Los archivos XAML pueden ser leídos por el *framework* WPF o Silverlight para ser compilados y renderizados. Los archivos que contienen el lenguaje de etiquetas de XAML, tienen la extensión .xaml, los cuales pueden ser codificados con cualquier codificador XML, pero para decodificarlos comúnmente se utiliza UTF-8.

2.2.2.3. .NET RIA Services

Silverlight provee de un *framework* flexible que le permite trabajar con datos distribuidos disponibles desde distintos tipos de servicios, por medio de comunicación asíncrona y manejo de operaciones *callback*.

.NET RIA Services proporciona un *framework* para el intercambio distribuido de datos, este se encuentra construido sobre *Windows Communications Foundation* (WCF) el cual es una parte del *framework* de .NET que provee de un modelo unificado de programación, para la construcción rápida de aplicaciones orientadas a servicios que se comunican a través de la internet y de redes empresariales[13] .

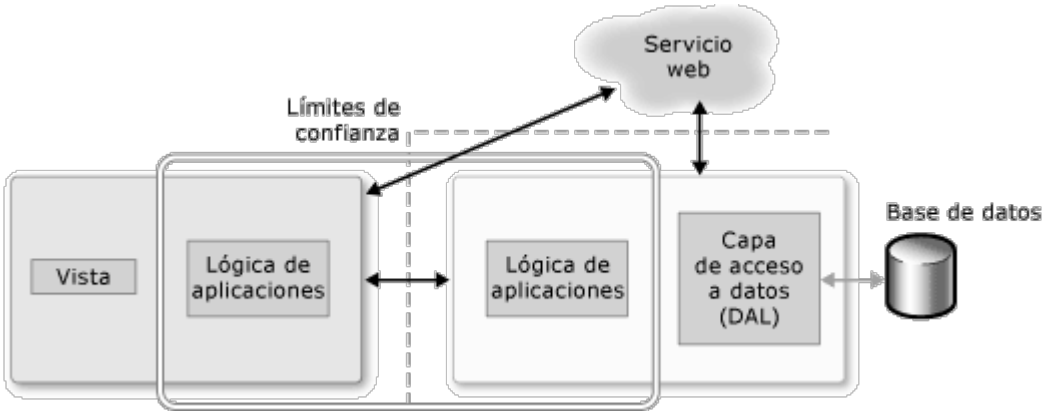
WCF permite el intercambio de datos entre el cliente de Silverlight y el servidor. Este servicio provee a los desarrolladores una forma simple de trabajar con el modelo de programación asíncrona, rastrear el estado de objetos dentro de las aplicaciones Silverlight, compartir códigos de validación entre

[13] Microsoft Corporation. *Windows Communication Foundation is…* <http://msdn.microsoft.com/en-us/netframework/aa663324>.

proyectos del lado del cliente y del lado del servidor y construir objetos *proxy* del lado del cliente, entre otros beneficios.

.NET RIA Services proporciona componentes, herramientas y servicios de la lógica del negocio de la aplicación RIA para ser consumidos por el cliente, lo cual brinda el beneficio de la no duplicación de código. En la figura 13, se muestra una RIA de n-niveles. El lugar que ocupa RIA Services en este modelo, es el cuadro entre la capa de presentación y la capa de acceso a datos, funcionando como una interfaz común para la gestión de datos de la aplicación.

Figura 13. **Aplicación de .NET RIA Services**



Fuente: http://msdn.microsoft.com/es-es/library/ee707344(vs.91).aspx. 23 de abril de 2011.

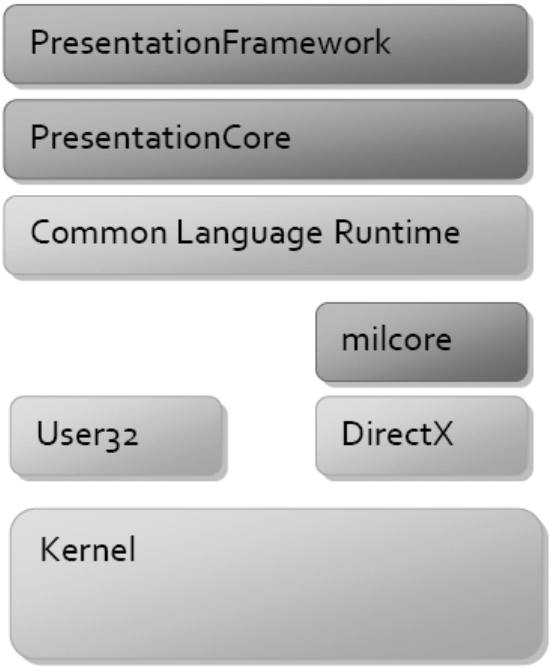
2.2.2.4. Windows Presentation Foundation

Windows Presentation Foundation (WPF) es un subsistema gráfico para el renderizado de interfaces de usuario, en aplicaciones basadas en Microsoft Windows. WPF proporciona un modelo consistente para construir aplicaciones y fomentar la separación entre las interfaces de usuario y la lógica del negocio.

Es utilizado por Microsoft Silverlight para proveer controles *web* embebidos con un enfoque orientado a un modelo de objetos de interfaces de usuario y no a animación, como es el caso de Adobe Flash.

WPF está diseñado para permitir crear sistemas de presentación guiados por datos. Cada parte del sistema está diseñada para crear objetos a través de propiedades establecidas que indican su comportamiento. Las aplicaciones tradicionales crean una pantalla para ser visualizada por el usuario y posteriormente enlazan los datos a dicha pantalla. En WPF, todo lo relacionado con el control y todo aspecto del despliegue en pantalla, es generado por medio del enlazado de datos (*data binding*).

Figura 14. **Componentes principales de WPF**



Fuente: http://msdn.microsoft.com/en-us/library/ms750441.aspx. 23 de abril de 2011.

En la figura 14 se muestran los componentes principales de *Windows Presentation Foundation*. WPF se encuentra constituido por código administrado y no administrado.

Las partes que se muestran en el color más oscuro en la figura 14 son las estructuras de WPF con las mayores porciones de código que lo conforman; de estas partes únicamente una está escrita en código no administrado, la cual es milcore. Milcore está escrito en código no administrado para permitir una mejor integración con el motor de DirectX. Toda la capa de presentación construida con WPF se realiza utilizando el motor de DirectX, para lograr un renderizado eficiente en cuanto al uso de *software* y *hardware*. Milcore necesita también un desempeño extremadamente sensitivo a la ejecución de WPF por lo que utilizará únicamente lo necesario para su ejecución, lo cual implica renunciar a ventajas que puede ofrecer el *Common Languaje Runtime* (CLR).

2.2.3. JavaFX

JavaFX es una herramienta para la creación de aplicaciones con contenido enriquecido y expresivo; permite la unión de características tales como la ubicuidad, capacidad, expresividad y desempeño en las aplicaciones *web*. Proporciona además un modelo unificado para el desarrollo e implementación en la construcción de RIAs, que integran medios enriquecidos de multimedia como el audio, vídeo, gráficas, texto enriquecido y servicios *web*.

El núcleo central de la tecnología JavaFX, es al igual que Microsoft Silverlight y Adobe Flex, su SDK. El SDK de JavaFX contiene el conjunto esencial de tecnologías, herramientas y recursos requeridos por los desarrolladores y diseñadores de *software*, para crear e implementar

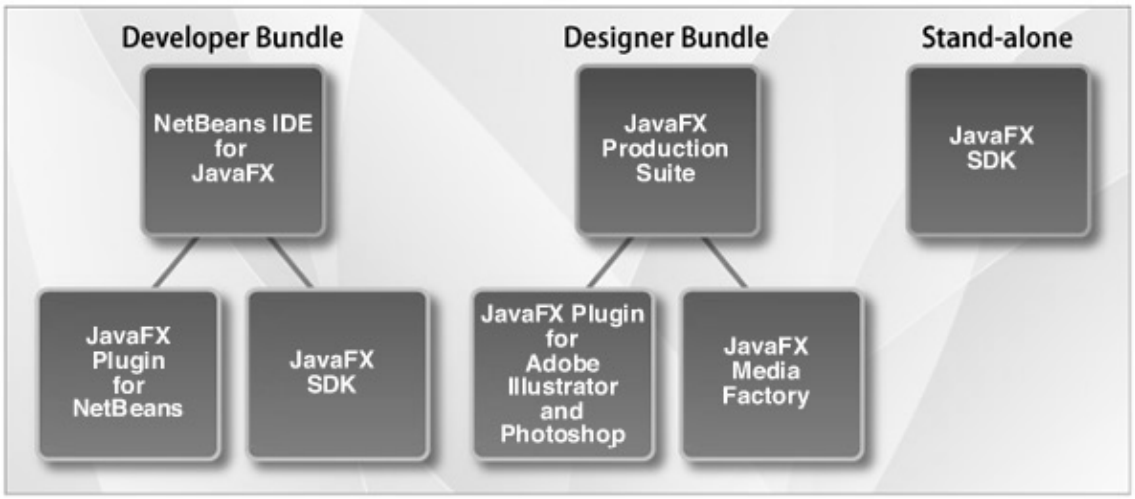
aplicaciones expresivas y potentes. Para crear estas aplicaciones se utiliza un lenguaje declarativo estático tipificado llamado JavaFX Script.

JavaFX puede ser adquirido en tres distintas presentaciones: la primera de ellas es una distribución para desarrolladores, la cual incluye el entorno de desarrollo NetBeans y el SDK de JavaFX. Esta distribución es especial para quienes desean agilizar el proceso de desarrollo de aplicaciones, por medio de un entorno de desarrollo; la segunda opción es la distribución para diseñadores. Esta permite a los diseñadores de interfaces de usuarios interactuar directamente con Adobe Illustrator o Adobe Photoshop para la exportación de gráficas o recursos, para la creación de la interfaz de usuario y convertirlo a un formato compatible con las aplicaciones JavaFX; por último, una distribución para aquellas personas que gusten desarrollar en un bloc de notas y que estén acostumbradas a trabajar con instrucciones desde consola.

Véase la figura 15 para comprender cómo están clasificadas las distribuciones de JavaFX.

JavaFX es una de las herramientas que posee una de las capacidades de mayor ubicuidad, debido al uso de la *Java Virtual Machine* (JVM), plataforma que gracias al tiempo que tiene de existir ha sido probada, ha madurado y se ha convertido en un medio de ejecución para aplicaciones Java de alto desempeño. Las aplicaciones JavaFX pueden tomar ventaja de los beneficios de desempeño de Java, tales como la *Virtual Machine HotSpot*, el recolector de basura y un avanzado conjunto de librerías. Proporciona además capacidades avanzadas de gráficas, audio y video renderizado con soporte para aceleración de *hardware*.

Figura 15. **Distribuciones de la plataforma JavaFX**



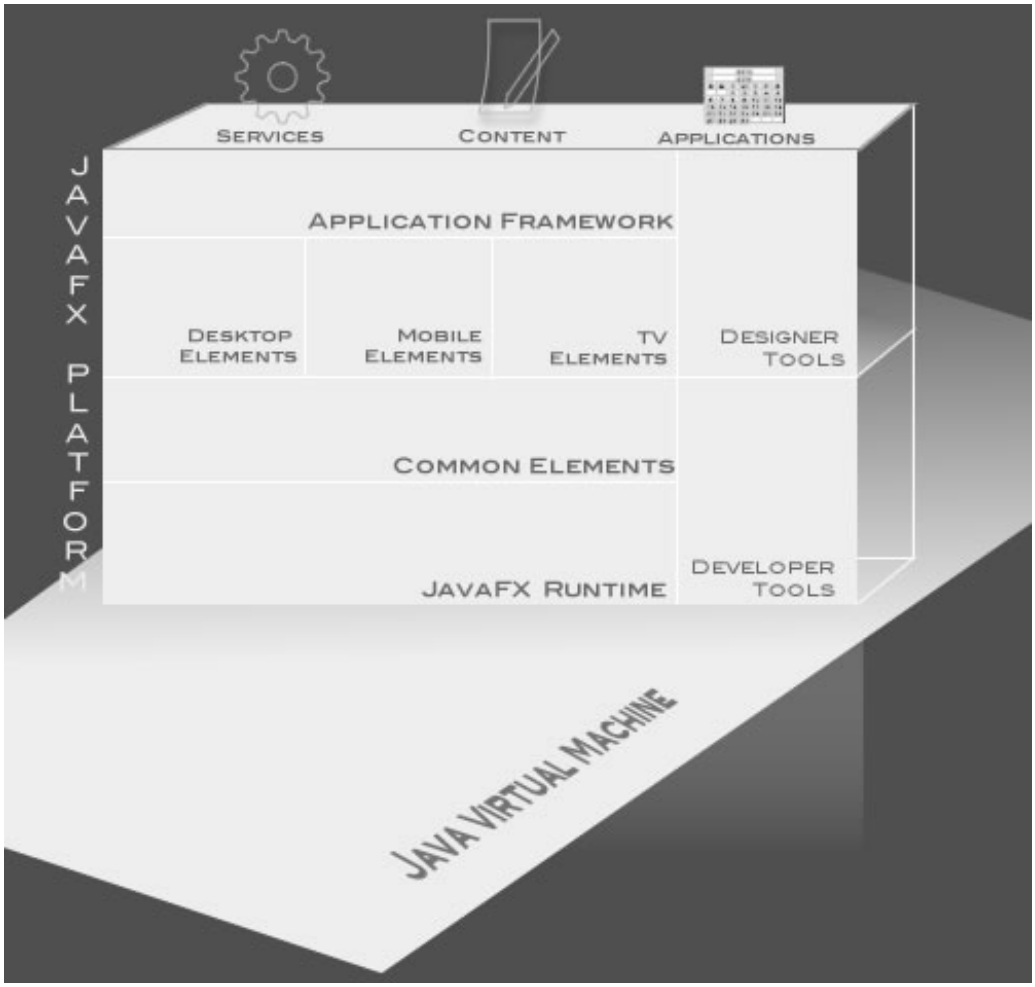
Fuente: http://javafx.com/about/overview/. 28 de abril de 2011.

En la figura 16, se muestra una simplificación de la arquitectura de la plataforma JavaFX. Como base se encuentra la *Java Virtual Machine*, que es la encargada de convertir el *bytecode* generado por la compilación de la aplicación Java a un lenguaje de bajo nivel, el cual dependerá de la arquitectura de la computadora sobre la cual se está ejecutando la aplicación. Sobre la JVM se encuentra el JavaFX *Runtime*, el cual es el conjunto de APIs utilizado para la ejecución de las aplicaciones JavaFX; estos se encuentran dentro del *Java Runtime Environment* (JRE) de Java específico para cada plataforma. Sobre esta capa se encuentran los Elementos Comunes.

Estos elementos comunes contienen APIs que son estables para cualquier plataforma, esto quiere decir que son todos aquellos APIs que no dependen de la plataforma sobre la que se está ejecutando la aplicación.

Sobre la capa de Elementos Comunes se encuentran los elementos de escritorio, móviles y de TV, que no son más que los APIs específicos para cada uno de los ambientes en que pueden ser desplegadas las aplicaciones JavaFX, y sobre estos APIs se encuentra el *framework* de la aplicación, que son bloques pre-construidos de código que pueden ser utilizados para facilitar la construcción de una aplicación JavaFX.

Figura 16. **Arquitectura de la plataforma JavaFX**



Fuente: http://javafx.com/about/overview/. 28 de abril de 2011.

Para la construcción de las aplicaciones se tienen dos tipos de herramientas:

- Las herramientas de desarrollo (*Developer Tools*): las cuales como su nombre lo indica, son las que ayudan a los desarrolladores a crear el contenido y los servicios de las aplicaciones JavaFX.

- Las herramientas de diseño: tienen como objetivo ayudar a los diseñadores con la conversión de medios, para ser utilizados en la construcción de aplicaciones JavaFX, y poder visualizar dichos medios ya convertidos, para ser utilizados por la aplicación.

JavaFX actualmente es mantenido por la empresa Oracle Corporation, y su versión estable actual es la 1.3.

2.2.3.1. Entorno de desarrollo

JavaFX, al estar basada en Java, cuenta con todo el respaldo que ofrece esta tecnología, por lo que existen distintos entornos de desarrollo para poder desarrollar aplicaciones que se han ido adaptando para soportarla.

Antes que Oracle comprara la compañía Sun Microsystems, esta última poseía un entorno de desarrollo el cual era utilizado para el desarrollo de aplicaciones Java. El entorno de desarrollo llevaba y conserva hasta hoy el nombre de NetBeans, el cual es un entorno de desarrollo de código fuente abierto (*open-source*) gratuito, y soporta el la ejecución de todos los tipos de aplicaciones Java.

El entorno de desarrollo NetBeans actualmente se encuentra en su versión 7.0, permite el desarrollo de aplicaciones de escritorio, móviles y aplicaciones *web.* Puede ser ejecutado sobre sistemas operativos Windows, Linux, Mac OS X y Solaris; esto es posible debido a que está construido sobre la misma herramienta Java, utilizando para su ejecución la JVM.

El que NetBeans utilice la JVM de Java para su ejecución, le permite ser una herramienta *cross-platform*. Este entorno de desarrollo puede ser extendido por terceros, para ofrecer nuevas funcionalidades a los usuarios. Soporta el desarrollo de aplicaciones con JavaFX a partir de la versión 6.5, en la cual además ofrece soporte completo de Java SE (*Standard Edition*), y posee un editor de JavaFX y el JavaFX *Composer*, el cual permite la creación, visualización previa y descripción de aplicaciones JavaFX de escritorio o móviles.

Otro entorno de desarrollo bastante reconocido es Eclipse. Este permite el desarrollo con múltiples lenguajes y posee un sistema de extensión mediante el uso de *plug-ins*. Es distribuido bajo la licencia *Eclipse Public License*[14], lo cual le permite ser distribuido como un *software* gratuito y de código fuente abierto. El sistema de extensión que posee Eclipse le da la fortaleza de no tener funcionalidades limitadas en el entorno de desarrollo, sino que permite el agregar funcionalidades nuevas, otorgándole de esta manera la oportunidad de ser utilizado con una variedad de lenguajes de desarrollo.

El SDK de Eclipse incluye el *Eclipse Java Development Tools* (JDT), ofreciendo un entorno de desarrollo con un compilador de Java incremental. Esta característica le proporciona el poder de trabajar con técnicas avanzadas

[14] The Eclipse Foundation. *Eclipse Public License – v 1.0.* <http://www.eclipse.org/org/documents/epl-v10.php>.

de refactorización y análisis de código. Este entorno de desarrollo trabaja utilizando *workspaces*, que no es más que un conjunto de metadatos sobre un archivo plano, que permite la modificación externa del archivo.

El *plug-in* de JavaFX para Eclipse, se encuentra disponible para sistemas operativos Microsoft Windows y Apple Mac, y posee una versión Beta que soporta Linux Ubuntu.

2.2.3.2. JavaFX Script

Las aplicaciones de JavaFX son escritas en JavaFX Script, un lenguaje diseñado por Sun Microsystems para desarrolladores *web* y diseñadores técnicos quienes gustan de trabajar en un contexto visual. Es un lenguaje de *scripting* declarativo que permite a los desarrolladores escribir código fuente que esté asociado a la estructura de la interfaz gráfica del usuario. JavaFX Script, al igual que otros lenguajes para la construcción de RIAs, posee capacidades para el *data binding* por medio de una sintaxis simple para la sincronización de estados de múltiples objetos, que permiten a las interfaces gráficas de usuarios cambiar de estado automáticamente, cuando se detecte el cambio de este en un objeto relacionado.

Es un lenguaje de tipos estáticos, esto quiere decir que el tipo de dato que retorna cada variable, parámetro y método, es conocido en tiempo de compilación. A la vez es también un lenguaje declarativo, describiendo cómo es la aplicación en vez de cómo crearla. El algoritmo que determina cómo mostrar la aplicación en la pantalla, es administrado por el *software* de soporte, el cual puede ser un API como el Java 2D de la librería *Swing*.

2.2.3.3. JavaFX TV

JavaFX TV provee de un ambiente para desarrollar y compilar aplicaciones JavaFX, que serán ejecutadas y utilizadas en una televisión.

La naturaleza de la plataforma de un televisor afecta la forma como se diseñan interfaces efectivamente. Existen distintas características por las cuales la plataforma de televisión difiere de la plataforma de aplicaciones de escritorio.

Con JavaFX TV es posible desarrollar aplicaciones que interactúen con el control remoto de la televisión, y permite que las interfaces sean colocadas sobre la imagen que muestra la televisión, haciendo la experiencia de uso modificable e interactiva, ya que funciona como una capa superpuesta a la imagen de la televisión.

2.2.3.4. JavaFX Mobile

JavaFX *Mobile* tiene como objetivo aprovechar las capacidades existentes en la actualidad en dispositivos móviles para el acceso a contenido enriquecido, utilizando la Internet por medio de JavaFX y Java ME (*Micro Edition*). JavaFX *Mobile* busca impactar a los usuarios por medio de Java ME ya que existen, según encuestas realizadas por Oracle Corporation[15], dos billones de dispositivos ejecutando Java ME hoy en día, por lo que pretenden ejecutar aplicaciones JavaFX *Mobile* en todos estos dispositivos que utilicen Java ME.

[15] Oracle Corporation. *JavaFX Moblie – At a Glance.* <http://java.sun.com/javafx/mobile/>.

2.2.4. Mono Moonlight

Mono Moonlight es una implementación de código fuente abierta y gratuita de Microsoft Silverlight, fue desarrollada buscando brindar la funcionalidad que brinda Silverlight para sistemas operativos basados en Linux y Unix. El proyecto Mono, el cual posee también una implementación para poder ejecutar el Microsoft .NET *Framework* en sistemas operativos basados en Linux y Unix, fue anunciado por Microsoft y Novell, en diciembre de 2007.

En la actualidad Novell ha sido adquirida por Attachmate, junto con todos los proyectos que Novell manejaba, incluyendo el proyecto Mono, el cual fue cancelado por parte de Attachmate, acto que provocó que el fundador del proyecto Mono y por lo tanto de Mono Moonlight, Miguel de Icaza, creara una nueva compañía llamada Xamarin para dar continuidad al proyecto Mono. A pesar de mantener el proyecto activo, se ha encontrado con el problema que Xamarin no tiene los mismos permisos que Novell para trabajar con el .NET *Framework*, por lo que este no tiene acceso al código no *open-source* del *Framework* de .NET.[16]

Actualmente, se encuentra disponible la última versión estable de Moonlight 2, la cual fue liberada el 17 de diciembre de 2009, aún bajo el mando de Novell.

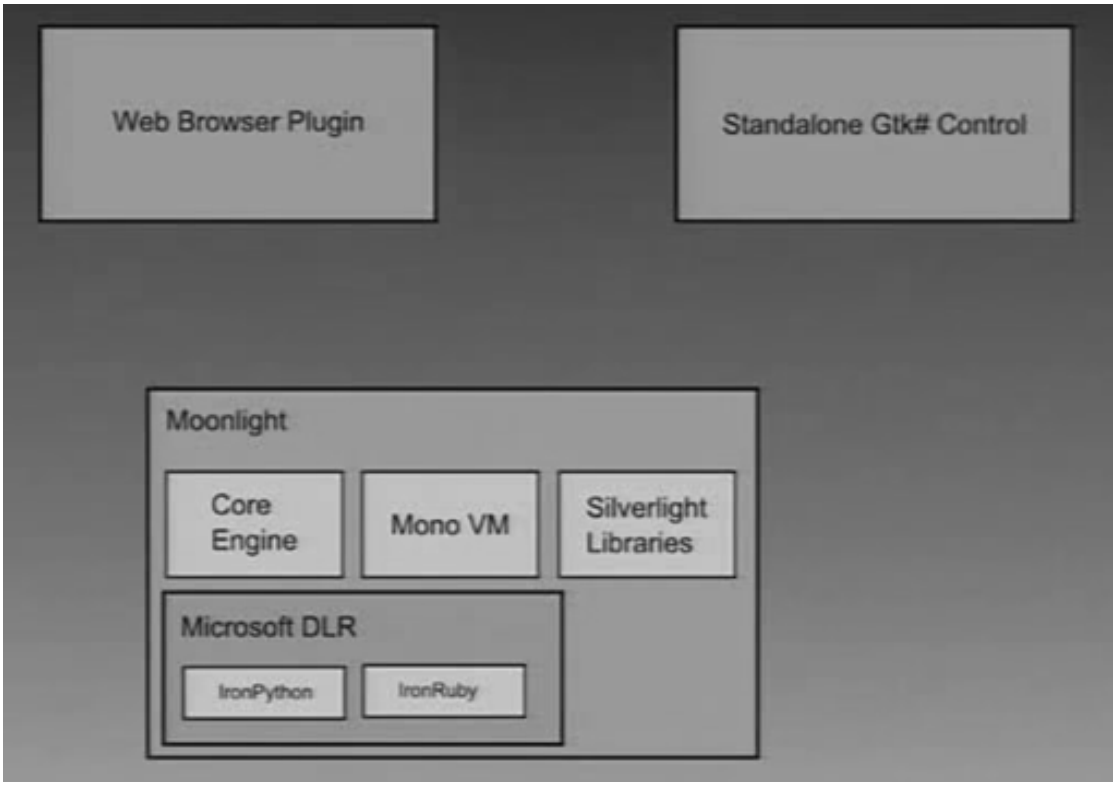
Los objetivos del proyecto Mono, y de Moonlight en específico, son el poder ejecutar aplicaciones Silverlight sobre el sistema operativo Linux, proveer un SDK para Linux que proporcione las herramientas necesarias para construir

[16] ALLEN, Jonathan. *The Death and Rebirth of Mono*. <http://www.infoq.com/news/2011/05/Mono-II>.

aplicaciones Silverlight y poder reutilizar el motor de Silverlight, para poder crear aplicaciones de escritorio.

En la figura 17 se muestra el entorno de ejecución de Mono Moonlight. En la parte superior izquierda se muestra el *plug-in* de Moonlight para navegadores *web*, el cual habilita la ejecución de aplicaciones Moonlight en navegadores de Internet.

Figura 17. **Entorno de Mono Moonlight**



Fuente: http://images.wikia.com/mono/images/a/a0/Moonlight_today.jpg. 29 de abril de 2011.

En la parte superior derecha se muestra el control *standalone* de Gtk#, es sobre esta estructura que se encuentra construido Mono Moonlight y es la interfaz que habilita la ejecución de aplicaciones Moonlight en aplicaciones de escritorio. Por último, en la parte inferior, se muestra la estructura de Moonlight como tal, en donde se puede ver la existencia del Motor Principal (*Core Engine*), la *Virtual Machine* de Mono y las librerías de Silverlight, que son utilizadas para la construcción de aplicaciones, además de encontrar el *Dynamic Language Runtime* de Microsoft, desde donde es posible ejecutar aplicaciones escritas en IronRuby e IronPython.

Mono Moonlight, por ser una implementación de Microsoft Silverlight para sistemas Unix, es programado en los lenguajes del .NET Framework, soportando inclusive la construcción de interfaces gráficas utilizando XAML.

Mono Moonlight puede ser ejecutado sobre sistemas de 32 bit y de 64 bit, utilizando navegadores de Internet Mozilla Firefox 2.0 o posterior y Google Chrome 4.x o posterior. El *plug-in* para ejecutar las aplicaciones Moonlight es compatible con los sistemas operativos SUSE Linux Enterprise Desktop 11, openSUSE 11.x, Ubuntu 9.10 o posterior y Fedora 12 o posterior. Por ser un *software* de código fuente abierto, Moonlight puede ser ejecutado en cualquier otra versión de sistema operativo Linux, pero se necesitará de la compilación del código fuente.

Moonlight 2.0 contiene código fuente que está bajo licencia GNU LGPL y MIT X11, esto incluye el motor gráfico en C++, el *runtime* de Mono y las librerías de clases que utiliza Mono. El *Microsoft Media Pack* es un producto distribuido por Microsoft que incluye licencia para los distintos *codecs* de audio y vídeo; este se encuentra disponible en el sitio de Internet de Microsoft, para el consumo de usuarios de Moonlight.

2.2.4.1. Entorno de desarrollo

Mono permite el desarrollo de aplicaciones sobre sistemas operativos Microsoft Windows, Linux y Apple MacOS, utilizando el ambiente integrado de desarrollo llamado MonoDevelop.

MonoDevelop es un entorno de desarrollo integrado, diseñado originalmente para trabajar C# y otros lenguajes .NET en ambiente GNOME; aunque está diseñado para ser una plataforma que permita el desarrollo de cualquier herramienta.

Originalmente MonoDevelop fue una extensión del entorno de desarrollo SharpDevelop para Gtk#, pero ha evolucionado mucho desde entonces. MonoDevelop permite a los desarrolladores la creación de aplicaciones para escritorio y para Internet de forma rápida, que puedan ser ejecutadas en sistemas operativos GNU Linux, Microsoft Windows y Apple Mac OSX. Permite también a los desarrolladores el trasladar de una forma plana y sencilla, aplicaciones .NET creadas con Visual Studio en un ambiente Microsoft Windows a un ambiente Linux, manteniendo el mismo código base para todas las plataformas.

Los creadores de este entorno de desarrollo buscan cumplir con muchos objetivos para apoyar el desarrollo de aplicaciones basadas en el .NET *Framework* sobre sistemas basados en Unix, entre los cuales se encuentran:

- Crear un ambiente de desarrollo para Mono que pueda ser ejecutado sobre ambientes basados en Unix.

- Dado que es un ambiente escrito en Gtk#, permite agregar funcionalidad para mejorar la experiencia de desarrollo en Gtk#.

- Busca integrar las herramientas construidas previamente dentro del proyecto Mono, por ejemplo MonoDoc, NUnit-Gtk y el *debugger* de MonoDevelop.

El núcleo de MonoDevelop es distribuido bajo licencias LGPLv2, aunque mucho de su código y *addins* es distribuido bajo licencia MIT/X11. Todo el código fuente se encuentra disponible a través de un repositorio de archivos *Subversion*.

MonoDevelop permite, a partir de la versión 0.10, la creación de interfaces gráficas a partir de un diseñador basado en Stetic, pero aún no es soportada la misma característica para proyectos *web* basados en ASP.NET, aunque actualmente se encuentran trabajando en ofrecer dicha funcionalidad.

El entorno de desarrollo MonoDevelop tiene, entre sus características, la autocompletación de código con soporte hasta C# 3, *templates* de código y *folding* de código; permite la configuración de ventanas, la creación de nuevas teclas rápidas y el uso de herramientas externas diseñadas para funcionar con MonoDevelop; soporta múltiples lenguajes de programación como C#, VB.NET, C/C++ y Vala; tiene la capacidad de *debugger* para Mono y aplicaciones nativas, permite la creación de aplicaciones Gtk# y proyectos *web* con ASP.NET. MonoDevelop; actualmente se encuentra en la versión 2.4.2 estable y 2.6 beta.

Existe además un entorno de desarrollo creado para soportar diseños de interfaces gráficas utilizando XAML. Este entorno de desarrollo es Lunar

Eclipse. Es un entorno de desarrollo que funciona como una aplicación *stand-alone* que se instala en el escritorio cliente y soporta animaciones básicas, así como la creación de movimiento, dimensionamiento y cambio de propiedades básicas de las formas soportadas por XAML. Actualmente tiene un alcance limitado, pero está cumpliendo con la meta más importante que es poderse integrar con MonoDevelop.

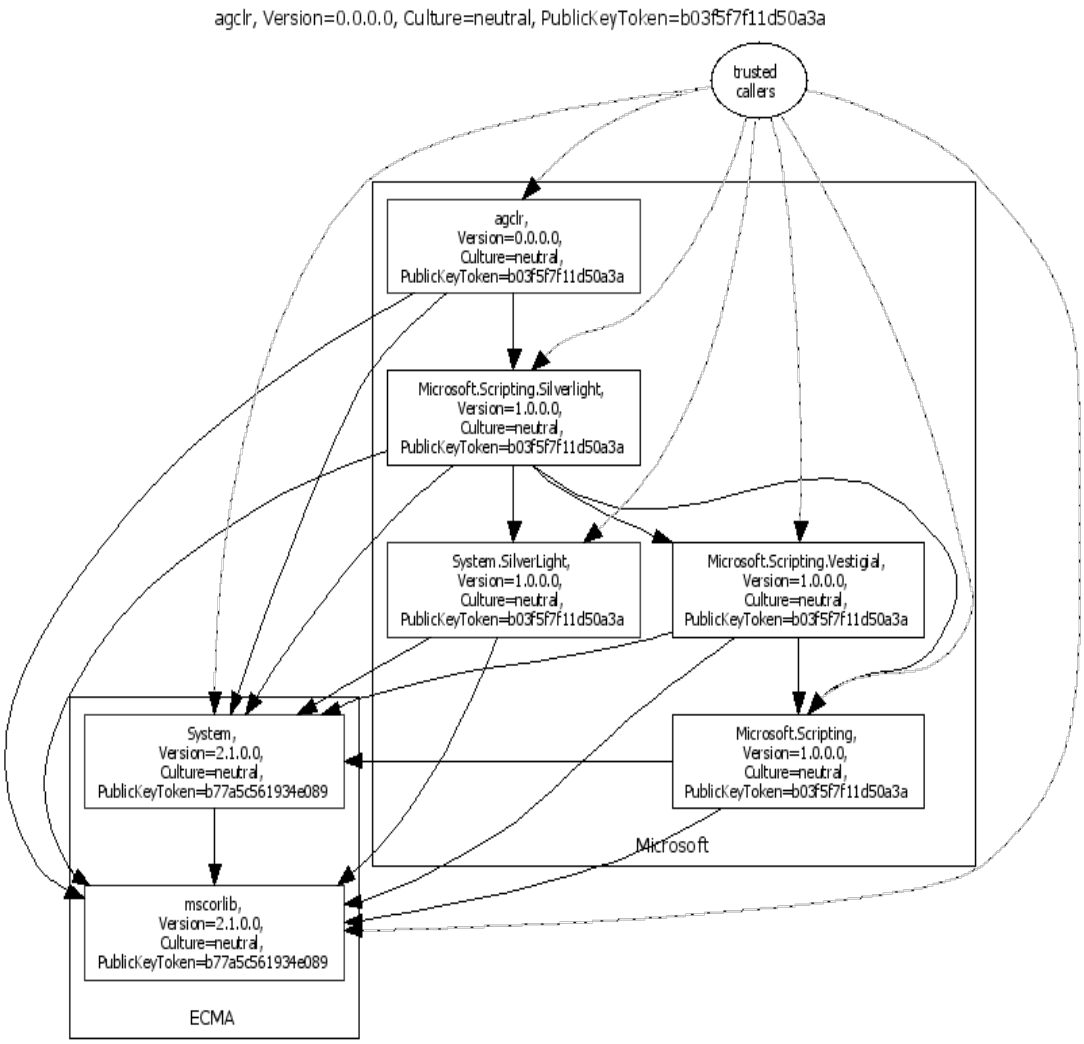
2.2.4.2. Dependencias

Mono Moonlight posee una serie de dependencias, entre las más importantes se mencionan los siguientes paquetes de desarrollo:

- Gtk+ 2.0
- XULRunner (por ejemplo mozilla-xulrunner190-devel para Mozilla Firefox 3)
- Alsa y/o PulseAudio

En la figura 18, se muestra una gráfica en donde se representan las dependencias que posee Mono Moonlight; es importante resaltar que Moonlight no posee ninguna dependencia circular entre sus *assemblies*, lo cual ayuda a tener un tiempo de construcción de aplicaciones al momento de compilar, mucho más reducido. En esta gráfica se les llama *Trusted Callers* a todos aquellos *assemblies* que son utilizados únicamente por *assemblies* catalogados como *FullTrust*. Desde el punto de vista de la construcción de la aplicación, al momento de compilar estos, son aquellos *assemblies* que no poseen el atributo *AllowPartiallyTrustedCallers*; y los *assemblies* dentro de un cuadro con los bordes en color gris claro, son todos aquellos que son catalogados como inseguros.

Figura 18. **Dependencias de Mono Moonlight**



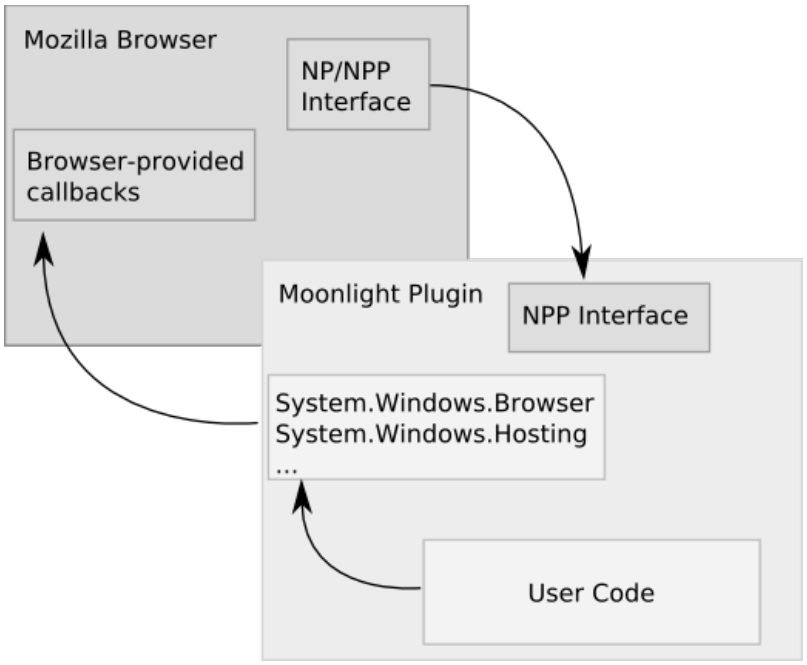
Fuente: http://www.mono-project.com/files/9/95/Agclr.png. 30 de abril de 2011.

Adicionalmente, si se desea reproducir recursos multimedia sin la necesidad de *codecs* de Microsoft, se necesita de ffmpeg.

2.2.4.3. *Plug-in* Mono Moonlight

Existen dos direcciones de comunicación entre el *plug-in* y el DOM (*Document Object Model*), las cuales se dan cuando el *plug-in* se comunica con el navegador y cuando el navegador se comunica con el *plug-in*; esto significa que es una comunicación en dos vías, como se muestra en la figura 19.

Figura 19. **Comunicación del *plug-in* Moonlight y el navegador de Internet**



Fuente: http://www.mono-project.com/Image:Plugin.png. 30 de abril de 2011.

Para la comunicación del navegador de Internet al *plug-in*, el *plug-in* expone la estructura del árbol interno del DOM hacia el host; esto es logrado utilizando el API NPAPI de Mozilla, el cual es un API para *plug-ins* con compatibilidad *cross-browser*.

Para la comunicación del *plug-in* al navegador de Internet son utilizados *namespaces*, tales como *System.Windows.Browser* y *System.Windows.Hosting,* los cuales pertenecen al DLL *System.Silverlight.dll*.

La carga del *plug-in* por el momento soporta únicamente una máquina virtual Mono en un único proceso. Actualmente, los miembros del proyecto Mono buscan poder ejecutar varias máquinas virtuales en un único proceso, resolviendo problemas como el aislar los *plug-ins* unos de otros, para la cual se está probando el uso de *AppDomains*.

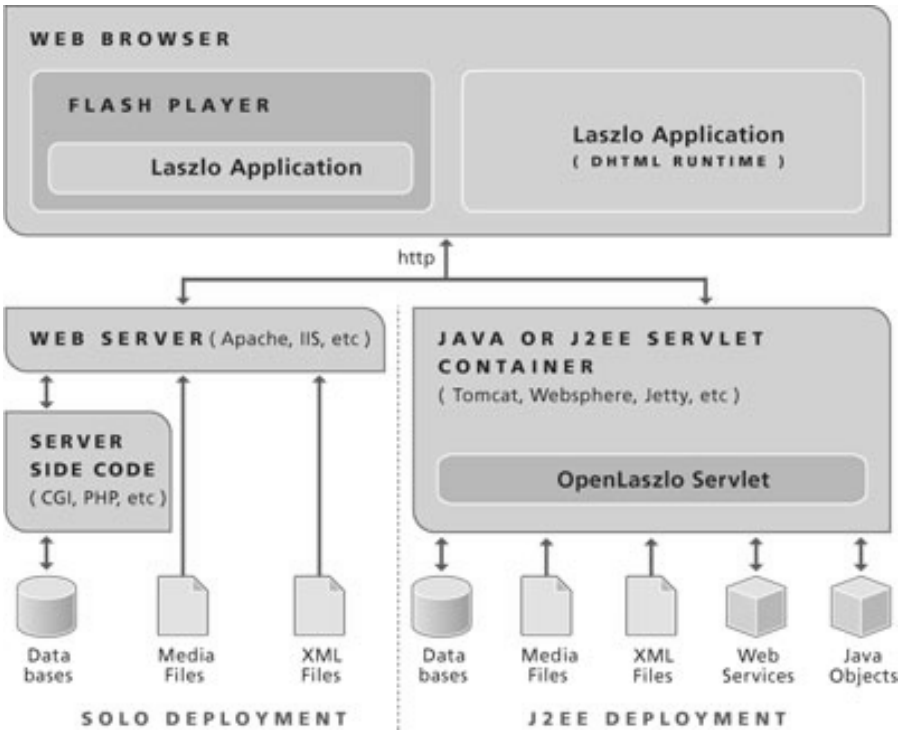
2.2.5. OpenLaszlo

OpenLaszlo es una plataforma para la construcción y despliegue de Aplicaciones Enriquecidas para Internet de manera fácil. La arquitectura de OpenLaszlo combina el poder de la usabilidad del diseño cliente-servidor con las ventajas administrativas y la efectividad de las aplicaciones *web*.

Las aplicaciones OpenLaszlo son escritas en un lenguaje basado en XML llamado LZX, el cual puede ser compilado hacia distintos tipos de plataforma, incluyendo OpenLaszlo 4.0, SWF 7, SWF 8 y JavaScript, tanto en DHTML como en AJAX.

El SDK de OpenLaszlo consiste de un compilador escrito en Java, una librería de JavaScript y un *servlet* de Java opcional, que provee de servicios adicionales a la aplicación que se está ejecutando. En la figura 20, se muestra la estructura de la arquitectura de OpenLaszlo.

Figura 20. **Arquitectura de OpenLaszlo**



Fuente: http://www.openlaszlo.org/architecture. 02 de mayo de 2011.

El compilador de OpenLaszlo compila los archivos fuente LZX en archivos binarios ejecutables para un ambiente de ejecución en específico. Actualmente, OpenLaszlo es compatible con Flash Player versiones 8 y 9, utilizando DHTML como ambientes de ejecución. Entre las características que ofrece el compilador de OpenLaszlo se encuentra la compilación de interfaces de usuario escritas en XML mediante el lenguaje LZX, el cual al ser compilado transforma el XML en *bytecodes* de SWF o en DHTML que posteriormente representarán la interfaz de usuario. El proceso de compilación es compatible con los estándares impuestos por *ECMAScript*.

OpenLaszlo puede hacer uso de un *servlet* de Java para el despliegue de aplicaciones en un servidor *web*. Este *servlet* tiene como objetivo ser el *proxy* de las solicitudes hacia la aplicación para tipos de *media* o para servicios *web* SOAP y XML-RPC. Entre las características que posee el *servlet* está que es por medio de este que se transforman los objetos LSZ, para ser objetos compatibles con Flash 8 y 9; funciona como memoria *caché* para almacenar las respuestas a los *request* en los servidores de datos y proporciona soporte para los *request* originados de servicios basados en XML como SOAP, XML-RPC y JavaRPC.

Las librerías de ejecución de OpenLaszlo, las cuales son llamadas también *Laszlo Foundation Class* (LFC), incluyen los componentes para crear las interfaces de usuario, la funcionalidad de *data binding* y los servicios de red. Dentro de esta librería se encuentran las responsabilidades de animación, creación de *request* para servicios XML y de *debugging*.

OpenLaszlo es distribuido bajo licencia *Common Public License* (CPL), la cual permite el acceso al código fuente, a los *nightly builds*, a la base de datos de *bugs* y ningún costo por adquirirlo.

Actualmente, se encuentra en versión estable OpenLaszlo 4.9, la cual soporta DHTML y Flash versiones 8, 9 y 10.

2.2.5.1. Entorno de desarrollo

Existen distintos entornos integrados de desarrollo para OpenLaszlo, a continuación se mencionan algunos de ellos.

IDE4Laszlo es un entorno de desarrollo basado en Eclipse para la creación, edición, *debugging*, y pruebas de aplicaciones basadas en el lenguaje LZX. Desde abril de 2008 existe una colaboración de la empresa Syte GMBH hacia el proyecto OpenLaszlo, la cual consiste en una actualización significativa hacia el *plug-in* IDE4Laszlo, para que permita ser utilizado con las últimas versiones de Eclipse, entre estas mejoras se menciona una fácil instalación de Eclipse RPC para Microsoft Windows y GNU Linux.

Otro de los entornos para desarrollar aplicaciones OpenLaszlo es Spket IDE, el cual es un entorno de desarrollo que posee un conjunto de herramientas para el desarrollo de aplicaciones OpenLaszlo utilizando JavaScript y XML, que es construido sobre Eclipse RPC (*Rich Client Platform*). Es un entorno de desarrollo pequeño y fácil de instalar, que ofrece capacidad de resaltado de código fuente, autocompletación de código para XML y JavaScript; este entorno de desarrollo es gratuito para uso no comercial.

Una tercera opción es el *plug-in* de OpenLaszlo para NetBeans, el cual está disponible para ser ejecutado con NetBeans a partir de la versión 6.1 con OpenLaszlo 4.0.12. Este posee un excelente editor de XML y JavaScript lo cual provee una gran facilidad para la edición de archivos .lzx. Gracias al entorno NetBeans se simplifica el proceso de despliegue de aplicaciones OpenLaszlo a cualquier servidor de aplicaciones, como por ejemplo Tomcat y GlassFish.

Uno más de los entornos para el desarrollo de las aplicaciones OpenLaszlo es LaszloParty. Este es gratuito para los desarrolladores OpenLaszlo y está escrito en Python. El proyecto fue iniciado en el 2007 y se encuentra aún en fase de desarrollo, utilizando WxPython como *framework* para la interfaz gráfica del usuario. El código es bastante liviano y capaz de realizar

un análisis sintáctico en los archivos .lzx e incluye un editor de XML para la escritura de archivos .lzx.

2.2.5.2. LZX

OpenLaszlo LZX es un lenguaje declarativo de etiquetas XML que posee código fuente JavaScript embebido, diseñado para aplicaciones *web* avanzadas. Su sintaxis es familiar para los desarrolladores que hayan trabajado con JavaScript, HTML y XML, y provee características de un lenguaje avanzado tales como animación, diseño de interfaces, capacidad de crear gráficas, *binding* de datos y comunicación con el servidor (comunicaciones persistentes, SOAP, XML-RPC y XMLHttpRequest) para crear experiencias enriquecidas para los usuarios. OpenLaszlo provee un robusto conjunto de componentes y es soportado por una extensiva documentación.

En LZX, las etiquetas de XML son utilizadas para crear objetos JavaScript, y JavaScript es utilizado dentro de los programas LZX para manipular los objetos creados por las etiquetas. En la mayoría de los casos, todo lo que puede ser hecho con una etiqueta puede ser realizado en JavaScript, de igual forma, y viceversa.

OpenLaszlo utiliza un esquema XML que define el conjunto de etiquetas utilizadas por LZX. Este esquema es utilizado por el compilador de OpenLaszlo para asegurar que los programas LZX son correctos. LZX permite también la definición de nuevas etiquetas. Las etiquetas creadas por los usuarios no son incorporadas dentro del esquema que está disponible por defecto, para la creación de interfaces de usuario; en vez de esto las nuevas etiquetas utilizan un esquema interno del cual el compilador se sirve para probar la validez de los programas.

LZX incorpora además conceptos estándar de la programación orientada a objetos, tales como la herencia, encapsulación y el polimorfismo; es por esto que cada etiqueta de LZX corresponde a la instancia de un objeto.

2.2.5.3. OpenLaszlo Development Kit

El OpenLaszlo Development Kit está constituido por instaladores compuestos de archivos binarios mediante los cuales se distribuye OpenLaszlo para las distintas plataformas. El archivo que se distribuye incluye el contenedor de *servlets* Tomcat 5.0.24, documentación, código de ejemplo, demostraciones y un tutorial sobre cómo trabajar con OpenLaszlo.

Los DevKits se encuentran disponibles para ser instalados en sistemas operativos Microsoft Windows, Unix/GNU Linux y Mac OSX.

Para poder instalar OpenLaszlo, es necesario tener instalada una versión del Java SDK mayor a 1.5; a excepción de sistemas operativos Mac en donde el Java SDK se encuentra instalado por defecto.

El paquete de *servlet* de OpenLaszlo contiene únicamente un *servlet*, el cual está compuesto por un conjunto básico de archivos necesarios para desplegar una aplicación OpenLaszlo, en cualquier sistema operativo que se encuentre ejecutando Java. Es recomendable desplegar un *servlet* únicamente cuando se tiene el conocimiento necesario para trabajar con contenedores y el conocimiento necesario respecto de las normas de seguridad que implica este tipo de despliegues.

2.2.6. HTML5

HTML (*HyperText Marckup Language*) fue desde un inicio un lenguaje diseñado para la *World Wide Web*. Originalmente diseñado como un lenguaje semántico para describir documentos científicos, aunque el diseño y su adaptación a través de los años han permitido que sea utilizado para otros tipos de documentos.

El diseño original de HTML no le permitía ser una buena herramienta para la construcción de aplicaciones *web*, por lo que la especificación de HTML5 busca rectificar este comportamiento y hacer de HTML una herramienta que pueda satisfacer las necesidades que las aplicaciones *web* demandan.

La idea de HTML5 surge por primera vez en talleres de la *World Wide Web Consortium* (W3C) en el 2004 con el apoyo de Mozilla y Opera, junto con quienes se presentó la idea de dar un giro a la funcionalidad de HTML, pero dicha propuesta fue rechazada y se acordó continuar con el desarrollo de elementos basados en XML para los documentos HTML.

Con una respuesta negativa por parte del W3C, poco tiempo después Apple, Mozilla y Opera anunciaron que trabajarían en la propuesta mostrada a la W3C por medio de un nuevo grupo de usuarios interesados en la evolución de HTML el cual llamaron WHATWG (*Web Hypertext Application Technology Working Group*). El proyecto WHATWG estaba interesado en muchos puntos específicos de la evolución del HTML, particularmente se hablaba que la tecnología debía ser compatible retrospectivamente. En el 2007, W3C mostró interés en el proyecto WHATWG y solicitó ser parte del desarrollo de lo que sería HTML5, dando así inicio el proceso de construcción de HTML5.

La especificación del HTML define un lenguaje abstracto para describir documentos y aplicaciones, y algunos APIs para interactuar con representaciones, en memoria de recursos que utiliza este lenguaje. La representación en memoria es conocida como DOM HTML o simplemente DOM y HTML5, define el DOM5 HTML.

Entre los beneficios que brinda la implementación de HTML5, se encuentra la compatibilidad con los navegadores de internet de versiones anteriores, la poca variación en la sintaxis de HTML y permitir que se omitan ciertas etiquetas definiendo la etiqueta de inicio del documento como HTML y no como XHTML. Entre las novedades que incluye para extender la capacidad de crear interfaces de usuario enriquecidas se encuentran las etiquetas <*video*> y <*audio*>, las cuales como su nombre lo indica permiten la presentación de recursos de video y audio sin hacer uso de *plug-ins* externos tales como Flash Player o Windows Media Player.

Cuando HTML fue creado, las propiedades de los estilos que se les aplicarían a los objetos eran definidas explícitamente dentro del código HTML, sin embargo en vez de continuar agregando más y más etiquetas al HTML, la W3C introdujo las hojas de estilo o CSS (*Cascading Style Sheet*), para poder separar el diseño de la codificación HTML, permitiendo a la Internet ser semántica en cuanto a la estructura.

HTML y CSS han evolucionado a través de los años, sin afectarse una a la otra, cada una ha seguido el objetivo por el que fueron diseñadas originalmente, sin embargo, tanto CSS3 como HTML5, pasarán a formar parte de la creación de nuevos *frameworks* y principios para la creación de sitios *web* modernos.

La forma como se conectan las tres tecnologías principales de la *web* son representadas en la figura 21, en donde se comparan con la forma semántica como se encuentran organizadas las oraciones del idioma español. HTML provee de los sustantivos, CSS de los adjetivos y adverbios, y JavaScript, los verbos.

Figura 21. **Interacción de HTML, CSS y JavaScript**

Fuente: CRANFORD TEAGUE, Jason. Visual Quickstart Guide CSS3, p. 8

2.2.6.1. Entorno de desarrollo

HTML5, al ser un lenguaje que los desarrolladores conocen desde hace mucho tiempo puede ser fácilmente escrito utilizando un bloc de notas, permitiendo generar archivos .html que provean interfaces de usuario consistentes y enriquecidas; sin embargo en la búsqueda de obtener un entorno de desarrollo que provea facilidades y rapidez para la construcción de las interfaces de usuarios, es necesario hacer uso de un entorno de desarrollo.

Aptana Studio 3, es una opción para el desarrollo de interfaces utilizando HTML5, además ofrece soporte para CSS3 y JavaScript. Permite el desarrollo y la ejecución de pruebas sobre las aplicaciones *web* creadas en un solo entorno de desarrollo. Además del soporte para HTML5, incluye información respecto del nivel de soporte para cada elemento dentro de los navegadores para Internet más utilizados, como por ejemplo Google Chrome, Mozilla Firefox, Microsoft Internet Explorer y Opera.

El *parser* que utiliza Aptana le permite identificar las etiquetas de CSS, JavaScript, HTML y PHP en un solo archivo de comandos. Aptana provee también de una ventana con una línea de comandos que permite editar la interfaz del entorno de desarrollo. Aptana es distribuido bajo licencia *Aptana Public License*, que permite el uso del mismo libremente.

Existe también un proyecto trabajado por Google, el *Google Web Toolkit Incubator*, el cual provee de soporte para algunas de las características de HTML5, a través de clases como GWTCanvas. Google provee de un *plug-in* GWT (*Google Web Toolkit*) para Eclipse, haciendo el desarrollo de aplicaciones *web* tan simple, como crear cualquier aplicación de escritorio con todas las facilidades de *debugging* disponibles.

Los *plug-ins* de modo de desarrollo están disponibles para Google Chrome y Mozilla Firefox, para realizar pruebas en aplicaciones antes de compilarlas con el código GWT. Una de las ventajas que provee GWTCanvas es el soporte para HTML5 en Internet Explorer, a través de imágenes. El código fuente desarrollado utilizando GWTCanvas funciona inclusive en Microsoft Internet Explorer y actualmente se encuentra en su versión 1.7.

Adobe Dreamweaver provee también un editor para HTML5, compatible con la versión CS5 de Dreamweaver llamado *HTML5 Pack*, el cual fue liberado al público el 31 de agosto de 2010 como una parte de Adobe Dreamweaver CS5 11.0.3. El soporte que provee Dreamweaver CS5 es nativo, además provee de soporte para algunas propiedades de CSS3.

2.2.6.2. CSS3

El *Cascading Style Sheets* o CSS es un lenguaje utilizado para especificar la apariencia visual de una página de Internet, a diferencia de HTML, que es utilizado para definir la estructura de documentos para ser distribuidos en la Internet. HTML indica al navegador de Internet cómo debe de organizar el contenido de una página y CSS indica al navegador de Internet cómo debe de lucir la página.

CSS3 es la abreviación para *CSS Level 3*, es la siguiente generación de este lenguaje de estilos que agrega nuevas capacidades para dar formato a los objetos que contienen las páginas de Internet. Al igual que HTML5, se encuentra aún en desarrollo por la W3C; sin embargo se encuentra en un punto estable para poder ser utilizado.

A través de los años CSS, ha evolucionado bajo la guía del W3C hasta como se conoce actualmente, pero el proceso ha sido lento en comparación con otros lenguajes. A pesar que CSS es un lenguaje estándar, es responsabilidad de cada uno de los desarrolladores de los navegadores para Internet implementar este estándar, lo cual ha llevado a encontrar algunas diferencias entre la implementación de CSS como un lenguaje *cross-browser*.

A diferencia de CSS1 y CSS2, CSS3 no se encuentra constituido por una única pieza, el *CSS Working Group* ha dividido las especificaciones en una serie de módulos, cada uno de ellos con sus propios tiempos de desarrollo. En la tabla II se muestra el listado de especificaciones que se están trabajando actualmente del CSS3.

Tabla II. **Tabla de especificaciones de CSS3**

Trabajo terminado	**Estado**	
CSS nivel 1	REC	
CSS nivel 2	REC	
Selectores	PR	
Color CSS	PR	
Prioridad alta	**Estado**	**Siguiente etapa**
Revisión 1 CSS nivel 2	PR	REC
Namespaces de CSS	CR	PR
Bordes y fondos de CSS	CR	PR
Estructura multicolumna CSS	CR	PR
Consultas de *media*	CR	PR
Prioridad media	**Estado**	**Siguiente etapa**
Instantánea de CSS 2007	LC	CR
Instantánea de CSS 2010	WD	WD
Definición de CSS móvil 2.0	CR	PR
Marquesina CSS	CR	PR
Media de página CSS	LC	CR
Perfil de impresión CSS	LC	CR

Continuación tabla II.

Valores y unidades CSS	WD	WD
Cascadas y herencia CSS	WD	WD
Texto CSS	WD	WD
Modos de escritura CSS	WD	WD
Líneas de cuadrícula CSS		WD
Ruby CSS	CR	WD
Contenido generado para *media* de página CSS	WD	WD
Fuentes CSS	WD	LC
Modelo de caja básico de CSS	WD	WD
Formato de estructura CSS	WD	WD
Discurso de CSS	WD	WD
Interface de usuario básica de CSS	CR	PR
Alcance de CSS		WD
Posicionamiento de cuadrícula en CSS	WD	WD
Formato de cuadrícula de CSS	WD	WD
Regiones de CSS		WD
Formato de caja flexible CSS	WD	WD
Valor de imágenes CSS	WD	WD
Transformadas en 2D de CSS	WD	WD
Transformadas en 3D de CSS	WD	WD
Transiciones de CSS	WD	WD
Animaciones de CSS	WD	WD
Sintaxis del atributo style de CSS	CR	PR
Prioridad baja	**Estado**	**Siguiente etapa**
Vista CSSOM	WD	WD

Continuación tabla II.

Modelo de caja extendido de CSS		WD
Modelo de objetos CSS		WD
Sintaxis CSS	WD	WD
Listas en CSS	WD	WD
Tablas en CSS		WD
Lector de tipo de *media* CSS	WD	
Posicionamiento CSS		WD
Contenido generado y reemplazado CSS	WD	WD
Estructura de líneas CSS	WD	WD
Presentación de hipervínculos CSS	WD	WD
Matemática CSS		WD
Niveles de presentación CSS	WD	WD
Hojas de estilo aural CSS		WD
Perfil de TV 1.0 de CSS	CR	PR
Comportamiento y extensiones a CSS	WD	WD
Introducción a CSS	WD	WD

Abreviatura	Nombre completo
WD	Borrador en proceso
LC	Última llamada
CR	Recomendación Candidata
PR	Recomendación Propuesta
REC	Recomendación

Fuente: http://www.w3.org/Style/CSS/current-work. 04 de mayo de 2011.

2.2.6.3. JavaScript

Es un lenguaje liviano y dinámico orientado a objetos, es conocido por ser utilizado como un lenguaje de *scripting* para las páginas de Internet. Este posee operadores y tipos, un núcleo de objetos y métodos. Su sintaxis es similar a la que se utiliza en los lenguajes Java y C, por lo que muchas estructuras de esos lenguajes aplican también para JavaScript. Una de las principales diferencias de JavaScript con estos lenguajes radica en que este no tiene clases; en vez de esto, la funcionalidad de las clases es lograda a través de prototipos de objetos.

La otra gran diferencia es que las funciones son objetos, es decir son ciudadanos de primer nivel, lo cual proporciona la capacidad a las funciones de mantener código ejecutable y ser enviado a otras funciones como si fuera un objeto.

Originalmente JavaScript y el DOM se encontraban fuertemente relacionados, pero con el paso del tiempo han ido evolucionando como entidades independientes. El contenido de la página está representado por el DOM y puede ser accedido y manipulado por JavaScript.

La ejecución de JavaScript dentro del modelo que utiliza HTML5 está constituida sobre un ambiente de ejecución de *scripts*, el cual consiste de un intérprete, el *stack* de contextos de ejecución, el código global, las funciones y los objetos funciones.

JavaScript soporta HTML ya que permite ofrecer y ejecutar eventos del lado del cliente que no sería posible ejecutar únicamente con HTML5 y CSS3.

Además de esta facilidad de mantener datos almacenados en la memoria de los navegadores de Internet, es también a través de este que es posible entablar la comunicación asíncrona con los servidores a través de AJAX, siendo este uno de los principales beneficios que provee JavaScript, para las aplicaciones enriquecidas que pueden ser escritas utilizando HTML5.

2.2.6.4. DOM

El DOM (*Document Object Model*) es una representación o modelo, de un documento y su contenido. No es únicamente un API; los criterios de conformidad de la implementación de HTML se encuentran definidos en esta especificación, en términos de operaciones que se ejecutan sobre el DOM.

El DOM fue diseñado para ser independiente de cualquier lenguaje de programación, haciendo la representación estructural del documento disponible desde un solo API de forma consistente.

La implementación debe de soportar alguna versión del *DOM Core* y *DOM Events*, puesto que esta especificación está definida en términos del DOM y algunas de las características están definidas como extensiones a las interfaces del *DOM Core.*

El *DOM Core* define los eventos y el modelo de los documentos que la plataforma *web* utiliza, mientras que el *DOM Events* o *DOM Level 3*, es una plataforma genérica y un sistema que permite el registro y el manejo de eventos, describe el flujo de los mismos a través de estructuras de árboles, provee información básica del contexto para cada evento y se encuentra construido sobre el *DOM Events Level 2*.

El DOM es representado dentro de HTML como un árbol de elementos. El elemento raíz de un objeto *document* es el primer elemento hijo del objeto, si es que existe alguno, si no tiene uno, entonces el objeto *document* no tiene ningún elemento raíz.

El término elemento raíz, cuando no se refiere al elemento raíz de un objeto *document*, se debe de interpretar como el ancestro más lejano del nodo que se está trabajando o del nodo al que se esté haciendo referencia.

El DOM junto con JavaScript permiten un manejo dinámico del documento de HTML sin necesidad de mantener una comunicación con el servidor, sino haciendo uso únicamente de las funciones almacenadas en el navegador *web,* las cuales pueden cambiar el aspecto de una página *web* o bien realizar cierto tipo de validaciones y mostrarlas al usuario sin enviar ningún dato a través de la red, proveyendo de aplicaciones que responden inmediatamente después de enviado el evento, simulando en gran parte las aplicaciones de escritorio, lo cual implica un comportamiento de una aplicación enriquecida para internet.

2.2.6.5. XHTML

Con la implementación de HTML5, se ha mostrado el XHTML5, el cual es una asociación de XML con HTML5

El XHTML (*Extensible HiperText Markup Laguage*) es una versión más estricta y limpia de HTML, por lo que la sintaxis de las etiquetas es la misma a excepción de la semántica con que son escritos ambos lenguajes. Es recomendado por la W3C por sobre el uso de HTML.

Entre los beneficios que brinda XHTML está que promueve la sintaxis XML, de forma de crear archivos con código fuente más legible y la habilidad de extender HTML hacia tecnologías basadas en XML, tales como SVG y MathML.

2.3. Elección de una herramienta para crear aplicaciones RIA

Existe una amplia variedad de herramientas para la construcción de Aplicaciones Enriquecidas para Internet, y la elección de una herramienta para ser utilizada en una aplicación deberá de realizarse basadas en distintos factores que pueden ser propios de la aplicación o de los requerimientos del usuario, evaluando las características de cada una de estas herramientas para satisfacer de la mejor forma las necesidades establecidas.

Al momento de evaluar cada una de las herramientas en función de las facilidades de desarrollo que brinde, se puede evaluar el SDK de cada una de estas aplicaciones inclusive llegando al nivel de comparar ciertas funcionalidades del *framework* y encontrando qué estructura satisface las necesidades establecidas. Por ejemplo, si se tuviera que desarrollar una aplicación en donde el tiempo de respuesta al *binding* de datos fuera primordial, se podría evaluar las clases de cada SDK y la forma como estas interactúan para obtener la que responda en mejor tiempo; también se puede evaluar desde el punto de vista de la expansión que tiene dicha tecnología, como podría ser el caso de la existencia de distintos *frameworks* que se adapten al SDK para el uso de IoC o MVC.

Se puede evaluar también en cuanto a las capacidades de comunicación que pueda tener con otros lenguajes de programación y la compatibilidad con otras tecnologías, ya que muchas veces puede ser necesario que la aplicación tenga interfaces estándar que puedan ser expuestas y utilizadas por otros

programas o servicios, por lo que si no se trabaja con un solo tipo de tecnología, será necesario tomar en cuenta esta evaluación.

Al igual que muchas otras herramientas de *software*, la decisión puede venir atada a que es la primera vez que se desarrollará en una tecnología RIA y que por lo tanto el conocimiento de la herramienta en específico no sea muy alto y deban de recurrirse a fuentes externas o documentación provistas por grupos de usuarios y la información del dueño de la herramienta, por lo que en estos casos la cantidad de documentación existente para apoyar el desarrollo de estas aplicaciones es muy importante.

La velocidad de desarrollo es crucial en muchos proyectos, por ello todas las herramientas que provean de generación de código automático utilizando un entorno de desarrollo que permita el *drag & drop* pueden ser muy importantes, también aquellas herramientas que provean del uso de *snipets* para la completacion de sentencias automáticamente, o bien en el mejor de los casos, entornos de desarrollo que provean la expansión de su funcionalidad con el uso de *plug-ins* que puedan proveer un entorno de pruebas unitarias, o una herramienta para refactorización de código pueden ser cruciales.

Como se ha visto anteriormente, existen muchos entornos de desarrollo para las distintas herramientas, unos con mejores funcionalidades que otros, siendo normalmente los entornos de desarrollo más completos y potentes aquellos que son distribuidos por los dueños de la herramienta RIA.

Finalmente, otro factor que se debe tomar en cuenta es el costo de la licencia para utilizar una herramienta RIA. Como se explicó a lo largo del capítulo, todas proveen de un SDK gratuito que puede compilar una aplicación desde línea de comandos y programado desde un bloc de notas, o bien ser

programado directamente en un documento de texto plano sin necesidad de compilarlo, como es el caso de HTML y JavaScript, por lo que si no se tiene inconveniente con el desarrollo de *software*, con este tipo de herramientas no hay ninguna restricción respecto de cuál utilizar y cuál no.

Pueden existir otros factores para la elección de una herramienta aparte de los ya mencionados, pero será necesario que sean detectados y que se encuentre el punto preciso en cada una de las herramientas que represente un medio para suplir la necesidad y una ventaja respecto de otra herramienta.

A continuación se muestra la tabla III que contiene un listado de características técnicas que se analizarán sobre las herramientas incluidas en este capítulo.

Tabla III. **Características técnicas a evaluar en las herramientas RIA**

Característica técnica	Descripción
Acceso fuera de línea	Acceso fuera de línea significa que la aplicación puede ser ejecutada desde fuera del navegador de Internet, así como cuando no hay conexión a Internet.
Aceleración gráfica	El *framework* es capaz de utilizar la Unidad de Procesamiento de Gráficos (GPU) para la administración de gráficas y el renderizado 3D. Esto da como resultado un uso más eficiente del CPU e incrementa el desempeño de la aplicación.

Continuación tabla III.

Animaciones, transformaciones y efectos	Habilidad para rotar, acercar, desvanecer, etc. Además de permitir realizar estas acciones a través del tiempo (animaciones).
Carga de archivos	Capacidad de subir archivos desde el cliente de la aplicación al servidor.
Data binding	Característica del lenguaje que permite a las variables ser enlazadas a (depender de) otras variables, y actualizarse automáticamente cuando los valores de las variables dependientes cambien.
Data grid	¿Existen componentes de *data grid* disponibles? ¿Existe soporte para paginación de datos? Control importante para distintos tipos de aplicaciones, especialmente empresariales.
Deep linking	Es un término para describir la habilidad de re-direccionar una página específica con un hipervínculo. Por ejemplo para mejorar la optimización de motores de búsqueda (SEO)
Efectos 3D	Habilidad para mostrar gráficas que visualmente parezcan tener tres dimensiones.
Interacción JavaScript	Capacidad para ejecutar código JavaScript desde dentro de la aplicación, o viceversa.
Lenguajes de programación	Lenguajes de programación con los que se pueden programar las aplicaciones.
Localización	Funcionalidad para traducir la aplicación a diferentes idiomas.
Plataformas de *hardware*	Plataformas de *hardware* sobre las cuales se puede ejecutar la aplicación.

Continuación tabla III.

Protocolos de red	Tipo de protocolos como SOAP, RTSP, etc., y protocolos de servicios *web* como REST, XML-RPC, etc., que soporta el *framework*.
Pruebas unitarias	Existe *framework* disponible para la ejecución de pruebas.
Requerimientos de *software*	Qué *software* es necesario para la ejecución de la aplicación. Este aspecto se ve categorizado dentro de sistemas operativos, navegadores para Internet y ambientes de ejecución.
Soporte multihilos	Soporte para ejecutar múltiples hilos del programa.
Soporte para estilos CSS	Habilidad para dar estilo a componentes, utilizando archivos o código CSS.
Soporte para gráficas	Existen componentes para la creación de gráficas disponibles. Característica importante para las aplicaciones empresariales.
Soporte para impresión	Capacidad de la aplicación para imprimir documentos.
Soporte para micrófono y webcam	La aplicación tiene capacidad para capturar video y audio desde micrófono y cámara *web.*
Soporte para motores de búsqueda	Soporta o no el acceso al contenido de la aplicación por medio de los motores de búsqueda.
Soporte para video HD	Habilidad para mostrar vídeo en alta definición.

Fuente: elaboración propia.

En la tabla IV se muestran las características técnicas descritas anteriormente para cada una de las herramientas que se han comentado en la sección anterior.

Tabla IV. **Características técnicas de las herramientas RIA**

Característica Técnica	Adobe Flash Builder	Microsoft Silverlight	Oracle Javafx	Mono Moonlight	Open-laszlo	HTML5
Acceso fuera de línea	Sí, con Adobe Air	Sí	Sí	Sí	Sí	Sí, por medio de la *caché* del navegador de internet
Aceleración gráfica	Sí, con Flash 10	Sí	No	Sí	--	Sí
Animaciones, transformacio-nes y efectos	Sí	Sí	Sí	Sí	Sí	Sí, mediante CSS3 y JavaScript
Carga de archivos	Sí, integrado con Flash 10	Sí, a través de controles de terceros	No de forma nativa, pero se puede implementar con Java	No	Sí	Sí
Data binding	Sí	Sí	Sí	Sí	Sí	Sí, mediante JavaScript

Continuación tabla IV.

Data grid	Sí, pero no existe pagina-ción fuera del control	Sí, con paginación	No, pero puede ser implementado con el control JTable de Java Swing (sin paginación)	Sí, con paginación.	Sí	Sí, mediante JavaScript.
Deep linking	Sí	Sí	No	No	Sí	Sí
Efectos 3D	Sí, en Flash Player 10.	Sí, con *Perspective 3D.*	Sí, con *Perspective Transform.*	Sí, a partir de Moonlight 4	--	Sí, mediante CSS con MozKit y WebKit
Interacción JavaScript	Sí, en ambas vías	Sí, en ambas vías	Sí, en ambas vías	Sí, en ambas vías	Sí, en ambas vías	Sí, en ambas vías
Lenguajes de programación	MXML Action-Script	C#, VB.NET, etc. XAML	JavaFX Script Java	C#, VB.NET, etc. XAML	LZX Java-Script	HTML JavaScript CSS3
Localización	Sí	Sí	Sí	Sí	Sí	No

Continuación tabla IV.

Plataformas de *hardware*	PCs Lap-tops. Teléfo-nos Móviles Conso-las de Juego.	PCs Laptops	PCs Laptops. Teléfonos Móviles. TV Blu-ray Consolas de Juego Vehículos.	PCs Laptops	PCs Laptops Teléfo-nos Móviles	PCs Laptops Teléfonos Móviles
Protocolos de red	*Strea-ming*: RTMP *Web Service-s*: XML-RPC SOAP REST Binario: AMF	*Streaming*: RTMP *Web Services*: WCF ASMX XML_RPC REST Binario: XML Binario	*Streaming*: RTMP *Web Services*: SOAP REST UDP XML-RPC, etc., para integración con Java.	*Web Services*: WCF ASMX XML_RPC REST	*Web Services*: XML-RPC SOAP REST	--
Pruebas unitarias	Sí, con Flex-Unit.	Sí	Sí, pero no nativo. Puede utilizar JUnit.	Sí	Sí, con LzUnit.	Sí, con herramien-tas de terceros.

Continuación tabla IV.

Requerimien-tos de *software*	Microsoft Windows Mac OS Linux Solaris	Microsoft Windows Mac OS	Microsoft Windows Mac OS Linux Open Solaris Android BlackBerry OS Windows Mobile	Linux FreeBSD SO basados en Unix	Micro-soft Win-dows Solaris Linux Mac OS	Cualquier SO con navegador para Internet que soporte HTML5.
Soporte multihilos	No	Sí, como parte de .NET.	Sí, desde que JavaFX puede utilizar las capacidades de hilos de Java.	No	No	Sí, mediante JavaScript.
Soporte para estilos CSS	Sí	No, solo XAML.	Sí	No	Sí	Sí
Soporte para gráficas	Sí	Sí	Sí	Sí, a partir de Moonlight 4.	Sí	Sí, utilizando JavaScript
Soporte para impresión	Sí	No, necesita de otra herramienta.	Sí	No	Sí	Sí
Soporte para micrófono y *webcam*	Sí	No	No	No	Sí	Sí

Continuación tabla IV.

Soporte para motores de búsqueda	Sí	Sí	No	Sí	Sí	Sí
Soporte para video HD	Sí	Sí	Sí	Sí	Sí	Sí

Fuente: elaboración propia.

En la tabla V se muestra un listado de características no técnicas a tomar en cuenta al seleccionar una herramienta, para la creación de aplicaciones RIA.

Tabla V. **Características no técnicas a evaluar a herramientas RIA**

Característica no técnica	Descripción
Costo de licencias	Costos asociados con el *framework* y las herramientas.
Licencia de distribución	Licencias con las que se distribuye el *framework* y las herramientas asociadas.
IDE para desarrollo	Qué herramientas e IDEs se ofrecen para los desarrolladores de *software*.
IDE para diseño	Qué herramientas e IDEs se ofrecen para los diseñadores de interfaces de *software*.

Fuente: elaboración propia.

En la tabla VI se encuentran los valores de las características no técnicas descritas en la tabla anterior para cada una de las herramientas que se mostraron en este capítulo.

Tabla VI. **Características no técnicas de las herramientas RIA**

Característica no técnica	Adobe Flash Builder	Microsoft Silverlight	Oracle Javafx	Mono Moonlight	Open-laszlo	HTML5
Costo de licencias	Adobe Flash Builder Premium: US$ 699 Adobe Flash Builder Standard: US$ 249	Visual Studio Ultimate: US$3799 Visual Studio 2010 Professio-nal: US$ 799	Ninguno	Ninguno	Ninguno	Ninguno
Licencia de distribución	SDK: Mozilla Public License. Flash Builder: Propietary EULA.	Propietario MS-EULA.	Compila-dor: código fuente abierto. (GPL 2.0) Herra-mientas: Código fuente abierto (GPL 2.0)	GNU LGPL y MIT X11	Common Public License (CPL 1.0)	--

Continuación tabla VI.

IDE para desarrollo	Adobe Flash Builder 4.	Microsoft Visual Studio. Eclipse 4SL.	NetBeans Eclipse *plug-in*.	MonoDe-velop	IDE4Lasz-lo Spket IDE NetBeans	Dreame-waver CS5 Eclipse Google *Web* Toolkit
IDE para diseño	Adobe Creative Suite. Adobe Catalyst.	Microsoft Expression Blend.	JavaFX Produc-tion Suite. JavaFX Authoring Tool	--	--	CSS3Ge-nerator CSS3 Gradient Mentor CSS3 Trans-forms

Fuente: elaboración propia.

3. APLICACIONES ENRIQUECIDAS PARA INTERNET Y OTROS PARADIGMAS

Como se ha mencionado con anterioridad, las Aplicaciones Enriquecidas para Internet ofrecen una serie de mejoras respecto de la experiencia del usuario, utilizando aplicaciones para Internet; también se ha mencionado la variedad de herramientas que existen para la creación de estas aplicaciones. Esta capacidad de brindar una mejor experiencia de usuario y las diversas herramientas para construir RIAs motivan la adopción de este paradigma para la creación de aplicaciones para Internet mucho más sofisticadas.

Entre los nuevos tipos de aplicaciones para Internet que se han desarrollado se encuentran aquellas que buscan aprovechar la mayor cantidad de beneficios que ofrece tales como su infraestructura y la ubicuidad, además de la gran cantidad de datos que son enviados y recibidos a diario, los cuales en gran parte son de libre acceso dentro de la Internet.

Las aplicaciones para Internet han evolucionado constantemente y gracias a esto se ha encontrado un amplio campo en dónde utilizar las Aplicaciones Enriquecidas para Internet, siendo ejemplo de esto los paradigmas tales como la *web* 2.0 y la *web* 3.0, el *cloud computing* y un conjunto de novedades que se han creado a partir de estos paradigmas. A través de este capítulo se encontrará cómo estos paradigmas han ido madurando, utilizando las RIA como una pieza clave.

3.1. *Web* 2.0

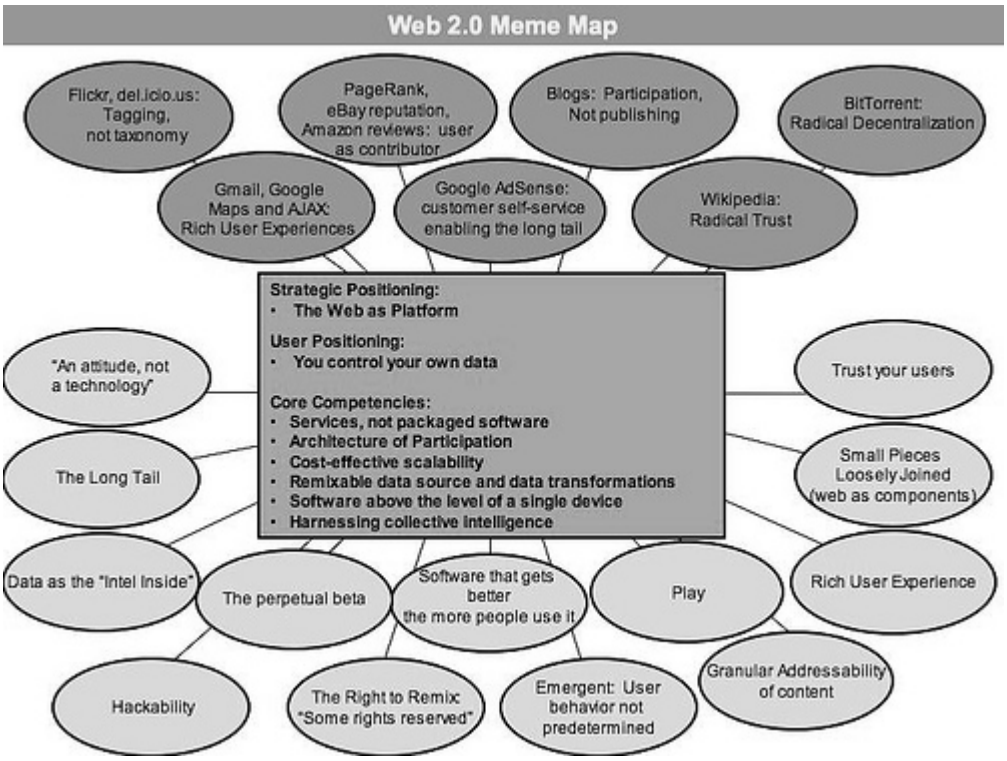
El paradigma de *web* 2.0 nace durante una reunión entre *O'Reilly Media* y *MediaLive International* en la cual se sostenía una lluvia de ideas. Dale Dougherty, un pionero de la *web*, indicaba que la *web* en vez de haber fracasado con la explosión de la burbuja en 2001, había evolucionado de una forma impresionante, ya que se había proveído de nuevas aplicaciones y nuevos sitios de Internet que eran actualizados con regularidad, además que las empresas que habían sobrevivido a la explosión de la burbuja habían demostrado poseer características similares entre sí.

Bajo la observación de las características en común surgió la pregunta ¿el colapso del "punto com" había marcado algún hito en que las cosas habían cambiado en la *web*, de tal forma que se pudiera llamar a lo que se estaba viendo en ese momento en la *web*, *web* 2.0? La respuesta fue afirmativa, y fue de este modo como nació el término *web* 2.0.

Como muchos otros conceptos, *web* 2.0 no establece un límite definido, en su lugar, provee de un núcleo principal de donde se expande; esto debido a que se puede visualizar la *web* 2.0 como un conjunto de principios y prácticas enlazadas entre sí que crean un sistema de sitios, que demuestran una o varias características a una distancia variable del núcleo.

En la figura 22 se muestra el mapa mental que fue desarrollado por O'Reilly Media, en el cual muestra las ideas principales que rodean el núcleo de la *web* 2.0.

Figura 22. **Mapa mental de la *web* 2.0**



Fuente: http://oreilly.com/web2/archive/what-is-web-20.html. 14 de mayo de 2011.

Como se menciona anteriormente, las empresas que sobrevivieron a la explosión de la burbuja del Internet tenían algo en común; su éxito se había centrado en la comprensión del poder de la *web* para explotar la inteligencia colectiva, por lo que todas aquellas plataformas o aplicaciones que estaban basadas en la gestión del conocimiento sobrevivieron, ya que se estaba dando una razón al Internet para existir y un uso que involucraba la colaboración de los usuarios, lo cual creaba cierto compromiso entre el usuario y la información que este compartía en la Internet.

Entre estas plataformas que representan a la *web* 2.0 se puede mencionar los *blogs*, las *wikis*, los sistemas de administración de contenidos y los medios de comunicación social (*social media)*.

3.1.1. *Blogs*

Un *blog* desde un punto de vista básico, es una página personal que posee el formato de un diario, aunque posee ciertas diferencias tales como el orden cronológico del *blog* y una forma de entregar la información diferente, además de la cadena de valor que ofrece, que es mayor, debido a la colaboración de las personas a la publicación, comentando al respecto o redistribuyendo la información.

Una de las características principales que diferencian a los *blogs* respecto de otros tipos de publicaciones en la Internet es el uso de los RSS (*Really Simple Syndication*). Los RSS son uno de los avances más significativos en la arquitectura que fundamenta la *web,* después del avance que mostraron los *hackers* al encontrar que era posible administrar bases de datos utilizando CGI. Los RSS permiten a los usuarios el crear enlaces no únicamente a una página de Internet, sino suscribirse a ella; esto significa que se enviarán notificaciones cada vez que la página a la que se esté suscrito mediante RSS tenga algún cambio.

Los *blogs* lograron ser parte de la *web* 2.0 no solamente gracias al RSS sino también a los *permalinks*. Los *permalinks* eran piezas de código con una funcionalidad básica, que lograron convertir los *weblogs* de un fenómeno constituido de publicaciones fáciles y sencillas, a conversaciones complejas entre comunidades; la simplicidad de gestionar la información de una publicación en el *blog* de otra persona y entablar conversaciones respecto de la

publicación, provocando de esta manera discusiones y pláticas entre las personas interesadas en el tema. Los *permalinks* fueron el primer intento de construir enlaces entre los *weblogs*.

Una analogía utilizada para describir y representar qué papel juegan los *blog*s en la *web* 2.0 es la siguiente: si se imagina que la *web* es un cerebro compuesto por las personas del mundo, todos los *blogs* existentes son el equivalente a las pláticas mentales que comúnmente se tienen en el cerebro al momento de pensar.

3.1.1.1. Blogspot

Blogspot es un servicio para la publicación de *blog*s creado por *Pyra Labs* y adquirido por Google en 2003[17]. Desde la adquisición de Blogspot por Google, este ha evolucionado constantemente, apoyando la integración de nuevos servicios, motivando así la colaboración en la generación de información a través de la Internet. Entre los servicios que fueron introducidos a Blogspot se encuentra la integración de Picasa, una aplicación para la compartición de fotografías en la Internet.

El uso de tecnologías RIA permitió la introducción de mejoras significativas en cuanto a la administración de los *blog*s dentro de Blogspot, entre estas mejoras se encuentra el incluir la organización de etiquetas y una interfaz para la edición de plantillas mediante el arrastrar y soltar (*drag-and-drop*). Posteriormente, se incluyó una nueva interfaz para la edición de publicaciones y se mejoró el manejo de imágenes en las publicaciones del *blog*.

[17] MCINTOSH, Neil. *Google buys Blogger web service*. <http://www.guardian.co.uk/business/2003/feb/18/digitalmedia.citynews>.

3.1.2. *Wikis*

Las *wikis* son páginas *web* que permiten a los usuarios la creación y edición de contenido sobre ellas, libremente. Las *wikis* fueron creadas en 1990 con el propósito de construir bases de conocimiento dinámicas y proveer de dicho conocimiento a los mismos usuarios que las consultan. Su principal diferenciador con otros medios de comunicación en Internet consiste en permitir la organización de las contribuciones para editar el contenido existente y creación de nuevo contenido, y que este forme parte del contenido del sitio *web*.

Las *wikis* ofrecen un conjunto de características que la ayudan a ser una excelente herramienta para proveer conocimiento, tales como el soporte de hiperenlaces, herramientas necesarias para el ingreso de texto enriquecido y enlaces cruzados entre páginas que pueden ser creados en el momento de la edición.

Esta herramienta forma parte de la *web* 2.0 debido a la capacidad de presentar información escrita y también la incorporación de sonidos, películas y fotografías; de forma que ofrecen al usuario toda una experiencia interactiva en la consulta que él está realizando.

3.1.2.1. Wikipedia

Wikipedia es una enciclopedia gratuita, basada en Internet, colaborativa y multilingüe, construida sin fines de lucro. Los escritores son voluntarios en todo el mundo, y la gran mayoría de artículos pueden ser editados por cualquier persona con acceso al sitio, previa aprobación de un equipo de control de la

empresa. Wikipedia, además de su contenido, ha sido reconocida por la rapidez con la que se actualizan o crean temas de acontecimientos recientes.

En la actualidad la Wikipedia se encuentra disponible en dos formatos, el primero de ellos es un formato *web*, accesible a través de navegadores de Internet, y el segundo es un ambiente para dispositivos móviles, el cual puede ser accedido desde estos.

Para la escritura de los artículos y definiciones que se publican en Wikipedia es necesario dar formato al texto que es ingresado, en función de estándares que utiliza Wikipedia, por lo que se necesita de la interfaz de un editor de texto que provea un entorno enriquecido, para facilitar la escritura y formato de nuevos artículos.

3.1.4. CMS

Un Sistema de Administración de Contenido (CMS por sus siglas en inglés) es una herramienta utilizada para la creación, administración y distribución de información, por medio de la creación de sitios para Internet.

La utilidad de esta herramienta se basa en la facilidad que brinda para gestionar el ciclo de vida de una página *web*, desde proveer las herramientas necesarias para crear el contenido, hasta la publicación de la misma, para finalmente almacenarla dentro del sistema CMS. Este provee también de las facilidades necesarias para la administración de la estructura de un sitio, la apariencia de las páginas publicadas y la navegación que se le provee a al usuario.

Comúnmente la funcionalidad de un CMS es dividida en cuatro etapas principales, las cuales son:

- Creación del contenido: esta etapa es llevada a cabo a través de un ambiente proveído por el CMS, diseñado para redactar el contenido, el cual busca brindar un ambiente similar al de un editor de texto. Este permite también la creación de páginas *web*, o la actualización de contenido, sin necesidad de tener ningún conocimiento de HTML;

- Administración del contenido: una vez que la página ha sido creada, es almacenada en un repositorio de contenido dentro del sistema CMS, este repositorio permite llevar un control de las versiones de una página, limita el acceso a la edición de contenidos en función del usuario, y es capaz de integrarse con otros sistemas de TI que provean información. En esta etapa pueden crearse también flujos de trabajo, utilizando para ello las mismas capacidades del CMS;

- Publicación del contenido: una vez que el contenido está en el repositorio, este puede ser publicado en un sitio *web* en la Internet o en una Intranet. Las publicaciones de CMS respaldan la consistencia de las páginas entre los sitios, además de darles la ventaja de utilizar altos estándares de apariencia dentro de todo el sitio;

- Presentación del contenido: los CMS proveen de facilidades para la presentación de los sitios *web*, como la creación de toda la navegación del sitio automáticamente, leyendo la estructura del sitio establecida en el repositorio, al momento de crear las páginas. Esto ayuda también con el soporte *cross-browser* de la aplicación.

Dada la facilidad para la creación y despliegue de información que proveen, los CMS son herramientas que apoyan la *web* 2.0, debido a que hace accesible a personas sin un nivel de conocimientos de desarrollo de *software* o de administración de sitios *web* especializados, al poder escribir y compartir información, además de gestionarla dentro del sitio construido. Esta herramienta provee también de la seguridad necesaria para que el sitio pueda ser administrado por un colectivo de personas, que aporten más información, para enriquecer el contenido del sitio *web*.

3.1.3.1. Drupal

Es un sistema de administración de contenidos escrito en PHP, gratuito y distribuido bajo licencia GNU. Utilizado principalmente para la construcción de *blogs* corporativos, políticos y sitios gubernativos en todo el mundo; también es utilizado para la gestión de conocimiento y como una herramienta de colaboración en las empresas.

Las herramientas como Drupal proveen de distintas características que ayudan a la generación de contenido multimedia, así como de texto enriquecido, lo cual es logrado a través de controles embebidos de reproducción de audio y video, además de las interfaces para la creación de texto enriquecido. Estas bondades son logradas utilizando interfaces de usuarios basadas en herramientas RIA.

3.1.4. *Social media*

Es la comunicación entre individuos, basada en la gestión del conocimiento. Es utilizado para transferir información hacia otros, como un instrumento de comunicación social.

Un medio social es un sitio *web* que no únicamente provee de información, sino que interactúa con el usuario de forma que este pueda colaborar con la información que está consultando. Esta interacción puede ser solicitar un comentario respecto de una publicación, votar calificando el contenido de un artículo, o recomendando un servicio o producto que se ha dado en un sitio *web*; esta interacción permite una comunicación en dos vías, en donde el sitio ofrece la información y el usuario opina respecto de ella para dar una referencia a personas que consulten la misma información posteriormente y proveyendo así del conocimiento colectivo, lo cual es la raíz de la *web* 2.0.

El medio social es una parte fundamental de la *web* 2.0 ya que sin ella los *blogs* o las *wikis* no poseerían ningún valor, pues no habría quién los consumiera y pudiera criticar o compartir la información, y así sumarle valor.

El medio social no únicamente se trata de *blogs* y *wikis*, también abarca medios como:

- *Microblogging*: son publicaciones de *blogs* caracterizadas por el tamaño reducido de sus publicaciones. Los *microblogs* permiten a los usuarios colocar pequeños elementos de contenido, como por ejemplo oraciones, imágenes individuales o enlaces a videos. Twitter es un ejemplo de *microblogging*.

- Redes sociales: son estructuras sociales construidas por individuos que se encuentran atadas o relacionadas a causa de uno o varios tipos específicos de interdependencia, como por ejemplo una amistad, intereses comunes, negocios, conocimiento o prestigio. Las redes sociales más utilizadas actualmente son Facebook y MySpace. Existen otras redes

sociales basadas en intereses específicos de usuarios, como por ejemplo redes sociales para compartir videos (YouTube) y fotografías (Flickr).

Estos son algunos de los ejemplos del conjunto de aplicaciones que abarca el medio social.

Las interfaces de usuario de estas aplicaciones, al ser consumidas por una gran cantidad de personas deben de cumplir con los requisitos de ser una interfaz intuitiva y descriptiva, de modo que el usuario pueda deducir cómo se utiliza la aplicación; es en este punto en donde se hacen necesarias las aplicaciones RIA, para crear todo el ambiente por medio del cual el usuario pueda brindar valor a la información que fue publicada, proveyéndole de los medios necesarios para hacerla de forma fácil y rápida.

3.2. *Web* 3.0

Al repasar el proceso evolutivo de la *web*, se puede comparar cada una de las etapas en que se ha encontrado, de la siguiente manera:

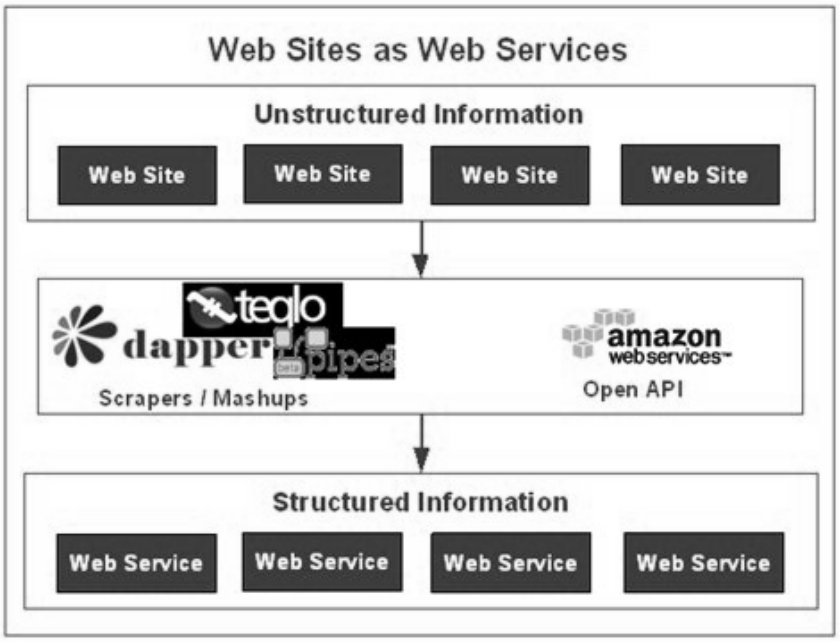
- La *web* 1.0 representa una librería, en donde esta funciona únicamente como una fuente de información pero no se puede contribuir de ninguna forma para expandir dicha información;

- La *web* 2.0 es como un grupo de amigos y conocidos. Se puede utilizar las conversaciones de amigos para recibir información, pero también se puede contribuir a la conversación y hacer de la plática una experiencia mucho más valiosa;

- La *web* 3.0 crea conexiones de personas con la información, a través de la Internet, esto por medio de *software* capaz de almacenar las preferencias de uso; también ofrece servicios de búsquedas, los cuales obtienen la información, la analizan y presentan los datos; todo esto gracias a que el principal diferenciador de la *web* 3.0 es que esta busca simular el entendimiento de la información que está publicada en la Internet. Un motor de búsqueda para la *web* 3.0 no únicamente busca palabras claves *(keywords*) sino que busca resultados en función del contexto de la solicitud de búsqueda.

La *web* 3.0 apunta a transformar la mayoría de sitios *web* en servicios *web*, como se muestra en la figura 23; los servicios pueden exponer y brindar efectivamente la información que poseen los usuarios.

Figura 23. **Efecto de red en sitios *web* como servicios *web***



Fuente: http://www.readwriteweb.com/archives/web_30_when_web_sites_become_web_services.php. 17 de mayo de 2011.

Esta transformación puede llegar a ocurrir de dos formas: la primera de ellas es a través de un API REST como lo ha hecho Amazon, del.icio.us y Flickr; y la segunda se refiere a aquellos que quieran mantener su información como propia, y la única forma en que son capaces de compartirla es a través de *mashups* mediante los cuales ofrecerán los servicios, como es el caso de Dapper, Teqlo y Yahoo! Pipes. El efecto de red para este caso, provocará que la información no estructurada, sea apartada de la información estructurada.

3.2.1. *Web* semántica

Actualmente, la estructura de la *web* está construida para que la información que contiene pueda ser únicamente interpretada por humanos. Es fácil para el ser humano visitar páginas *web* y entender cuál es el contenido y contexto que estas tienen. Una computadora es incapaz de desarrollar este pensamiento. Un motor de búsqueda está capacitado para buscar por medio de palabras claves (*keywords*), pero no puede entender cómo estas palabras claves pueden ser relacionadas para dar un contexto a la información de la página.

Con la *web* semántica las computadoras son capaces de escanear e interpretar la información en las páginas *web*, utilizando agentes. Estos agentes son programas que pueden rastrear datos a través de la *web*, buscando información relevante relacionada a la petición recibida.

La *web* semántica está organizada en colecciones de información llamadas ontologías. En términos de Internet, una ontología es un archivo que define la relación entre un grupo de términos.

La *web* semántica propone ayudar a las computadoras a “leer” y utilizar la *web*. La idea principal es la siguiente: la *metadata* agregada a las páginas *web* puede hacer de la existencia de la *World Wide Web*, algo que las computadoras puedan leer. Esto no le dará inteligencia artificial o hará que las páginas *web* tomen decisiones, pero si le proveerá a las computadoras las herramientas necesarias para encontrar, intercambiar e interpretar la información contenida en la página *web*.

Las soluciones para la *web* semántica evolucionan, en cuanto a lenguajes diseñados para el análisis de datos, entre los que se puede mencionar: *Resource Description Framework* (RDF), *Web Ontology Language* (OWL) y *Extensible Markup Language* (XML). HTML es utilizado para describir los documentos y los enlaces entre ellos. Por su parte RDF, OWL y XML pueden describir objetos arbitrariamente, como por ejemplo, personas, reuniones o repuestos.

Estas tecnologías son utilizadas en conjunto para proveer de una descripción que satisfaga o remplace el contenido de los documentos *web*. El contenido puede ser escrito como datos descriptivos almacenados en bases de datos accesibles en la *web*, o en otros archivos que utilizan HTML o XHTML que contienen XML o únicamente código XML.

3.2.2. *Web* 3D

La *web* 3D es la evolución de la *web* a un ambiente tridimensional. Esta combina elementos de realidad virtual con elementos creados a través de juegos en línea del tipo MMORPGs (*Massively Multiplayer Online Roleplaying Games*), en la cual la *web* se convierte en un mundo digital, en donde se pueda crear la ilusión de la profundidad para crear el ambiente de tres dimensiones.

La *Web* 3D se ha empleado en distintos servicios, además de los juegos en línea, tales como modelos de realidad virtual y soluciones multimedia.

3.2.2.1. *Second life*

Es un mundo virtual en línea creado por Linden Lab y lanzado en el 2003. En su funcionamiento se identifican dos partes principales: los clientes llamados *viewers* que son gratuitos, y los usuarios de la aplicación llamados residentes, los cuales interactúan unos con otros a través de avatares. Se entiende por *avatar* a las representaciones digitales de los usuarios.

Una de las características sobresalientes de este proyecto, es que además de crear un "mundo virtual" este maneja su propia economía. Dentro de *Second Life* existe el Linden dólar, y es utilizado para comprar, alquilar o intercambiar propiedades, bienes y servicios con otros usuarios. Este dinero puede ser cambiado por dólares norteamericanos.

En enero de 2007 se creó el OpenSimulator, con el objetivo de ser un *software* de código fuente abierto para crear clientes de *Second Life*. Este simulador está escrito en C# y es ejecutado bajo el ambiente de ejecución de Mono, el cual provee de las herramientas necesarias para la construcción de la interfaz enriquecida de usuario, que es necesaria para interactuar con la aplicación.

3.2.3. *Web* penetrante

La *web* penetrante es un término utilizado para referirse a la aplicación de la *web* mas allá de las computadoras personales y teléfonos móviles, ya que

puede usarse en otros dispositivos electrónicos u objetos de uso diario, tales como refrigeradoras, ventanas, calzado deportivo o automóviles por ejemplo. El que estos objetos tengan acceso a la Internet brinda la posibilidad de recabar información personalizada de cada usuario de estos dispositivos, y proveer de dichos datos a los sistemas interesados en conocer dicha información; esta alimentación de datos a otros sistemas es realizada por medio de servicios *web*.

Esto conlleva una responsabilidad para el usuario ya que él es el único responsable de la información que compartirá con los demás.

3.3. *Cloud computing*

La computación en la nube (*cloud computing*) facilita la implementación de *software* empresarial, minimizando costos y ayudando a aumentar la velocidad de crecimiento de los negocios.

La estructura del *cloud computing* se encuentra conformada por dos secciones: el *front-end* y el *back-end*. Estas dos secciones se conectan entre sí por medio de una red de computadoras, comúnmente la Internet.

El *front-end* es la sección en donde se encuentra la computadora que utiliza el usuario o cliente. El *back-end* es la "nube" en la que se encuentra el sistema.

En la sección del *back-end* se encuentran varias computadoras que funcionan como servidores de aplicaciones y de datos que crean los servicios que se ofrecen y que son consumidos por el cliente. En teoría, cualquier programa para computadora puede ser ejecutado desde la sección del *back-end*.

Un servidor central administra el sistema, monitorea el tráfico y la demanda del cliente para asegurarse de la disponibilidad del servicio sin interrupciones.

Entre las razones más contundentes respecto de por qué utilizar *cloud computing* se encuentran las siguientes:

- El cliente es capaz de acceder a las aplicaciones y a los datos contenidos en ellas, desde cualquier lugar, en cualquier momento.

- Ayuda a reducir el costo de *hardware*, ya que no se tiene necesidad de comprar el equipo físico, ni de la utilización del espacio para colocarlo, ni de brindar el mantenimiento e instalaciones que este necesita.

- Se reducen los costos en cuanto a soporte de TI, ya que lo servidores son mantenidos por el proveedor del servicio.

3.2.1. *Software as a service*

El *software* como servicio (SaaS, por sus siglas en inglés) ha ganado mucha aceptación como un nuevo modelo de acceso al *software*, en el cual los vendedores proveen una versión de su *software* en un ambiente basado en Internet, permitiendo a los clientes utilizar el servicio pagando únicamente la carga de uso que haya recibido el *software*, ya sea por proyecto o por medio de una suscripción al servicio. El beneficio que esto representa es que las actualizaciones del *software* son realizadas de forma transparente al usuario y no dependen del cliente que contrata el servicio, no existe ningún proceso de licenciamiento, ni el robo de licencias.

La tecnología RIA ha sido catalogada como un habilitador del *software as a service* (SaaS)[18] ya que provee de una funcionalidad enriquecida la cual anteriormente era únicamente posible en aplicaciones de escritorio.

3.3.1.1. *Salesforce*

Es una compañía que provee de servicios de *cloud computing*. Distribuye *software* empresarial por medio de suscripciones que venden a sus clientes; esto les permite hacer uso de las aplicaciones en la nube. La herramienta más conocida de *Salesforce* es su CRM, el cual se ha dividido en distintas áreas de la administración de clientes debido al gran crecimiento que ha tenido, por lo que se pueden encontrar servicios tales como *Sales Cloud*, *Data Cloud*, *Service Cloud*, *Colaboration Cloud* y *Custom Cloud*.

Los servicios SaaS que provee *Salesforce* necesitan brindar un ambiente similar a las aplicaciones de escritorio, ya que debe de permitir al usuario realizar una serie de procedimientos extensos, y dado que este sistema tiene como objetivo sustituir una aplicación de escritorio, debe brindar un ambiente amigable para el usuario; este objetivo es logrado por medio del uso de tecnologías RIA, para proveer de la interfaz de usuario enriquecida que se necesita.

3.3.2. *Platform as a service*

Las plataformas como servicio son una variación del SaaS; esta forma de computación en la nube brinda ambientes de desarrollo de *software* como un

[18] DEB, Brijesh, BANNUR, Sunil G., BHARTI, Shaurab. *Rich Internet Applications (RIA) Opportunities and Challenges for Enterprises.*

servicio. Se pueden construir aplicaciones que son ejecutadas sobre la infraestructura del proveedor y entregadas a los usuarios por medio de la Internet desde los servidores del proveedor. Como ejemplo de una plataforma como servicio se encuentra Salesforce.com y Google *App Engine*.

Son plataformas para las cuales la herramienta de desarrollo como tal se encuentra alojada en la nube y es accedida a través de un navegador de Internet. Con PaaS, los desarrolladores pueden construir aplicaciones *web* sin la necesidad de instalar ninguna herramienta en sus computadoras y pueden realizar el despliegue casi sin ningún conocimiento en administración de sistemas de información.

Entre los beneficios que provee el uso de PaaS se encuentra la capacidad de poder soportar una gran cantidad de usuarios desarrollando, dando mantenimiento y desplegando aplicaciones al mismo tiempo, así como el proveer de los medios necesarios para crear aplicaciones *web*, sin la necesidad de tener un conocimiento experto en la creación de *software*.

3.3.2.1. Google *app engine*

Google *App Engine* permite ejecutar aplicaciones *web* en la infraestructura de Google, virtualiza las aplicaciones a través de múltiples servidores. Soporta el desarrollo sobre lenguaje Python, Java y Go. Soporta otros lenguajes como Groovy, JRuby o Scala utilizando extensiones. Existe un plan gratuito por medio del cual los desarrolladores pueden utilizar el servicio de PaaS; si se desea aumentar las capacidades que ofrece el servicio gratuito deberá de pagar por la mejora en recursos.

3.4. Tendencias del desarrollo *web*

La tendencia del desarrollo de la *web* está enfocada en obtener y dar información personalizada al usuario, de manera que tengan una forma simple de recolectar información que sea de su interés para poder analizarla e interpretar los resultados de la forma como lo necesiten.

La *web* puede llegar a ofrecer alternativas del interés de los usuarios en función de las preferencias que estos hayan demostrado en su comportamiento al utilizar la Internet, optimizando de esta forma su interacción con la *web.*

En función de las capacidades que posee la *web* y el rumbo que está tomando, se puede clasificar como puntos clave para el crecimiento, evolución y expansión de la *web* y la creación de mejores interfaces de usuario, ya que este es el medio por el cual el usuario interactúa con la aplicación.

La gestión de la información es otro punto importante para el desarrollo de la *web,* debido a que existe una gran cantidad de datos disponibles en la que hace necesaria la existencia de métodos de búsqueda que trabajen de mejor forma, como es el caso de las búsquedas semánticas en la *web*.

Por último la expansión a otros dispositivos que pueda llegar a alcanzar, debido a que esto permitirá recopilar más información en tiempo real respecto de aspectos muy específicos de cada usuario, permitiendo a la *web* posteriormente utilizar estos datos para satisfacer de mejor manera las necesidades que posea un usuario en las distintas actividades que son realizadas día a día, en las cuales utiliza la Internet.

3.4.1. Interfaces de usuario

La creación de interfaces de usuario es uno de los campos a los que se les ha puesto mucha atención en la actualidad, dando como resultado el uso de las tecnologías RIA para ofrecer al usuario un entorno mucho más amigable.

Las empresas han puesto mucha intención en este aspecto y se han llegado a desarrollar herramientas y servicios como *Windows Presentation Foundation* o la integración de herramientas de Adobe como *Photoshop* e *Illustrator* con herramientas de desarrollo como Adobe Flash Builder o Java FX, para permitir crear mejores interfaces de usuario de forma más sencilla, e inclusive creando nuevos especialistas de UX (*User Experience*) encargados exclusivamente del diseño de pantallas que utilizarán los programas como interfaces de usuarios.

Esta avance en la interfaz de usuario ha llevado a que muchas herramientas busquen facilitar el trabajo en paralelo, en equipos de creación de *software,* en donde existen los encargados del diseño y creación de la interfaz de usuario, y del área de desarrollo como tal, ofreciendo una forma de integrar la interfaz de usuario con la lógica programada por los desarrolladores de una forma rápida y sencilla.

Las interfaces de usuario han venido también a facilitar y a permitir que se construyan aplicaciones mucho más potentes, basadas en la recolección y presentación de gran cantidad de información a los usuarios, la creación de interfaces que aprovechen los beneficios de los dispositivos tales como pantallas *touchscreen* o la detección de movimiento, para poder manipular el

software que se está utilizando y de esta forma brindar toda una experiencia al utilizar alguno de estos dispositivos.

3.4.2. Gestión de la información

La Internet se encuentra constituida por una gran cantidad de información que ha sido creada a lo largo del tiempo; esta información ha sido lograda a través de la colaboración de millones de usuarios de la Internet, gracias a las facilidades que brindó el paradigma de la *web* 2.0.

Al navegar en Internet es impresionante la cantidad de información que se muestra cuando se busca un tema en específico. El problema es que esa información no puede ser interpretada dentro de un contexto de búsqueda, para optimizar el listado de resultados, por lo que crear sistemas informáticos que gestionen la información es algo necesario de hacer.

La implementación de este tipo de búsqueda es realizada por medio de la *web* semántica, buscando crear bases de datos que permitan encontrar la información que se necesita basadas en el sentido semántico y no en etiquetas.

El problema más grande que se ha encontrado hasta ahora son las herramientas y la construcción de motores de búsqueda que sean útiles y funcionales para los usuarios, por lo que la creación de sistemas informáticos que gestionen la información, es de gran ayuda y representa un gran avance en cuanto al uso de la información en la Internet.

3.4.3. Expansión hacia otros dispositivos

La expansión hacia otros dispositivos permite el poder utilizar Internet para recopilar, procesar, almacenar y distribuir información en relación con un individuo en específico. Existe una gran variedad de dispositivos que tienen la capacidad de conectarse a una red e informar respecto de situaciones que están ocurriendo en un momento dado. La información es transferida y mostrada en tiempo real. Por ejemplo el uso de Internet en el calzado deportivo el cual puede ofrecer información al usuario tal como: la ruta que se ha recorrido, la distancia y la velocidad a la que se está moviendo.

3.4.4. *Web* 4.0

Desde ya se ha empezado a hablar respecto a la *web* 4.0. Esta evolución de la *web* parece estar orientada al uso de sistemas operativos en la Internet, haciendo uso de infraestructuras provistas por el *cloud computing*, en donde el sistema operativo se encuentre disponible a través de Internet y los clientes necesiten únicamente de una conexión a esta red para acceder a su sistema operativo y a toda la información que esté dentro de su sesión.

Esta idea ya ha sido implementada por Google, con el sistema operativo Google Chrome OS el cual fue liberado al público el 15 de junio de 2011, por medio de la puesta en el mercado de las computadoras personales ChromeBooks. Estas computadoras son *notebooks* que accederán a Internet para cargar el sistema operativo creado por Google.

El anuncio del lanzamiento de este sistema operativo no ha sido del todo bien recibido por la comunidad de críticos de tecnología. La mayoría de ellos opina que el utilizar un sistema operativo que se encuentra en la Internet

significaría que toda la información que posee el usuario estaría a disposición de Google.

El que toda la información de los usuarios esté a disposición de Google representa un riesgo debido a distintas causas tales como: ¿Qué pasaría con los datos de un usuario que ha sido retirado de la red de Google por alguna razón en específico?, ¿Qué pasaría si las leyes de un país solicitaran a Google toda la información contenida en la sesión de un usuario?, ¿Qué tan segura puede ser la red de Google para no tener ningún ataque informático que permita la pérdida de información? o ¿Qué pasaría si la red de Google llegara a colapsar por alguna razón, como ha pasado con muchas otras redes de alta disponibilidad?

Estas son algunas razones por las cuales los usuarios deberían de pensar en utilizar los sistemas operativos en la nube; así como los empresarios deben de pensar en el costo que implica para sus empresas utilizar esta tecnología después de haber evaluado sus riesgos.

4. ESTUDIO DE MERCADO

4.1. Investigación preliminar conceptual

Para la elaboración de este capítulo fue necesaria la realización de una investigación a través de libros de texto e información electrónica, que permitieron llevar a cabo la formulación y ejecución del estudio de mercado de manera correcta.

La importancia de esto radica en que el estudio de mercado genera una gran cantidad de información útil, siempre y cuando este se diseñe de manera correcta, es decir, de manera que el estudio permita obtener las respuestas a las preguntas que son determinantes para alcanzar los objetivos planeados y poder así formular conclusiones respecto a las Aplicaciones Enriquecidas para Internet.

4.1.1. ¿Qué es un estudio de mercado?

Los estudios de mercado son conocidos también como investigaciones o análisis de mercado y asimismo existen diversos criterios acerca de su concepto.

La definición global de Philip Kotler para un estudio de mercado, en la que abarca todas las etapas que lo conforman, dice: “Es el proceso sistemático

de diseño, obtención, análisis y presentación de datos y descubrimientos pertinentes a una situación de marketing específica que enfrenta la empresa".[19]

Otra definición por parte del Lic. Alfredo López Altamirano en su obra "¿Qué son, para qué sirven y cómo se hacen las investigaciones de mercado?", dice que un estudio de mercado es: "el esfuerzo para obtener y analizar la información sobre las necesidades, deseos, gustos, recursos, actitudes y comportamiento del público"[20]. Así se puede citar a diversos autores y definiciones acerca del tema, pero el fin de realizar un estudio de mercado es el mismo.

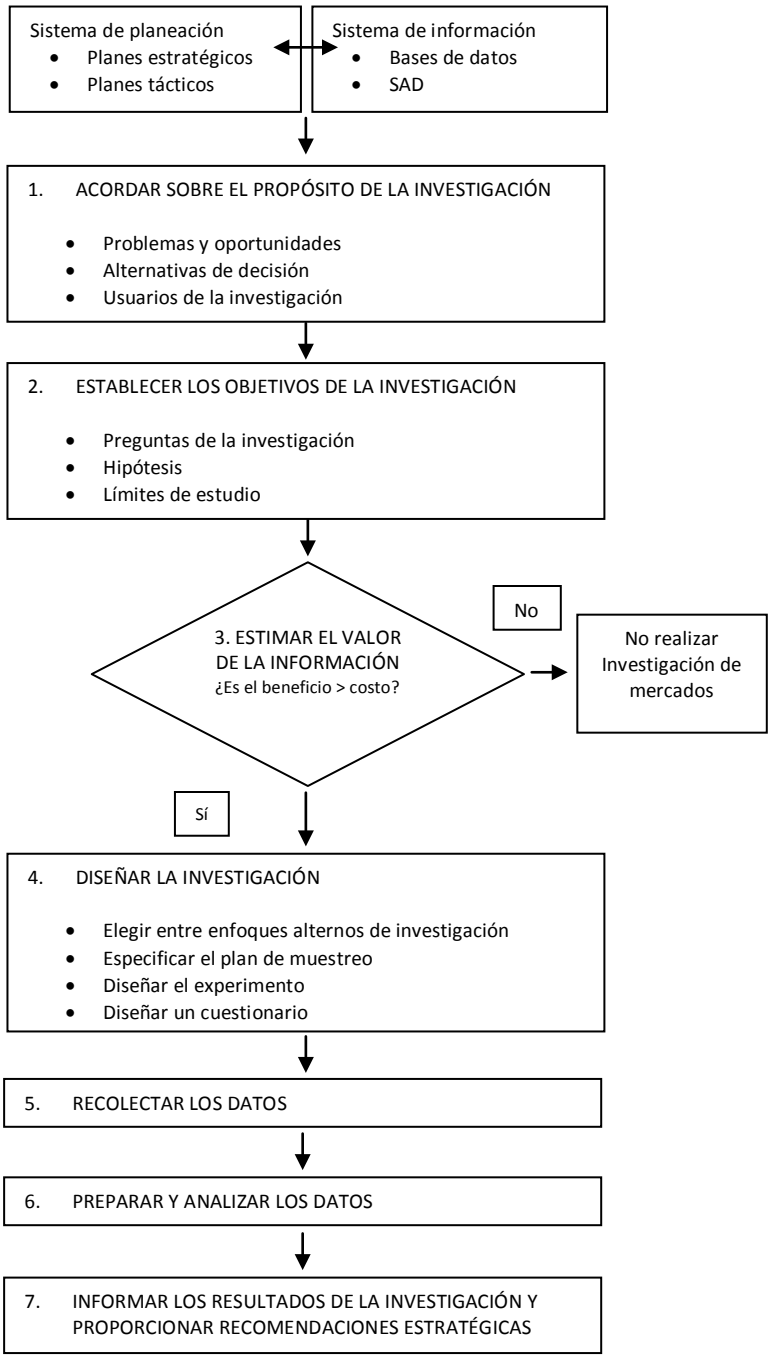
En general, un estudio de mercado es una herramienta utilizada con el objetivo de proporcionar ayuda en la toma de decisiones, a través de información selecta, exacta y oportuna; ya que busca dar solución a un problema o una interrogante definida al dar inicio con el estudio.

Las etapas que conforman un estudio de mercado se ilustran en la figura 24. Estas etapas guían el estudio de mercado desde la concepción del mismo hasta las conclusiones, a través de un enfoque sistemático que busca aplicarse correctamente para obtener resultados acertados al problema de forma eficiente.

Es importante mencionar que no todas las etapas del proceso de investigación son necesarias de realizar, según el objetivo, el tiempo y los recursos de los cuales se disponga, así se decidirá la elaboración del mismo.

[19] KOTLER, Philip. *Fundamentos de Marketing.*
[20] LOPEZ ALTAMIRANO, Alfredo. *Que son y para qué sirven las investigaciones de mercado.*

Figura 24. **El proceso del estudio de mercados**



Fuente: AAKER-KUMAR-DAY, Investigación de mercados. p.40.

4.1.2. Propósito de la investigación

Establecer un propósito claro para realizar la investigación o estudio de mercado es de vital importancia para el mismo debido a que si no es formulado correctamente y se delimitan los intereses sobre los cuales se realizará el estudio, solamente se obtendrá información y hallazgos sin importancia e inútiles, que no aportarán valor a la toma de decisiones sino solamente generará un gasto de la inversión.

4.1.3. Objetivos de la investigación

La redacción de los objetivos de un estudio de mercado debe realizarse de manera cuidadosa y precisa para que a través de ellos pueda asegurarse que la información que se obtenga al finalizar el estudio cumpla el propósito del mismo.

Es necesaria la redacción de un objetivo general y de objetivos específicos. El primer objetivo da un rumbo al estudio de mercado, comunicando al investigador la información que será necesaria para el desarrollo del estudio.

Los objetivos específicos proporcionan límites y alcances al estudio de mercado; la correcta redacción de los mismos ayudan a lo largo de todo el diseño del estudio, ya que como su lo nombre indica, especifican los intereses que posee el investigador.

4.1.4. Estimar el valor de la información

Para este estudio de mercado se debe de estimar el valor de la información, ya que los beneficios obtenidos a través del estudio de mercado para este trabajo de graduación sobrepasan por mucho el tiempo y esfuerzo a dedicar, para la elaboración del mismo.

4.1.5. Diseñar la investigación

El libro “Investigación de Mercados” de Aaker-Kumar-Day provee de una corta y concisa definición de lo que implica diseñar: “asegurarse de que sus piezas encajen bien”[21]. Esta definición es algo que nunca debe de pasarse por alto ni tomarse a la ligera, ya que es en esta etapa en donde todas las decisiones se encuentran interrelacionadas, por lo que cualquier argumento que se disponga, afectará a toda la elaboración del estudio.

4.1.6. Recolectar los datos

Para la recolección de la información existen métodos cualitativos, de observación y cuantitativos. Los métodos cualitativos son necesarios cuando se desea conocer la perspectiva mental del cliente, todo lo que no ha de poderse determinar con base en una observación o medición.

Los métodos de observación son los menos utilizados debido a la limitación de información que puede ocasionar, aún así, estos presentan varios beneficios y pueden ser utilizados también en combinación con los otros dos métodos para completar la recolección de la información.

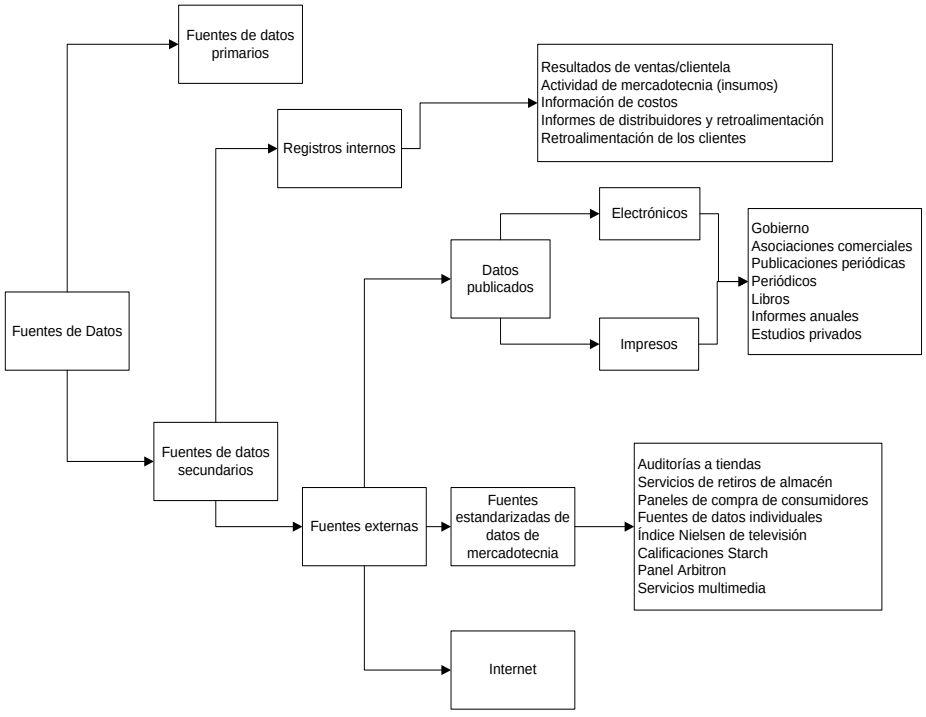
[21] AAKER, David, KUMAR V., DAY, George S. *Investigación de Mercados*.

Los métodos cuantitativos conllevan distintos cálculos matemáticos y estadísticos para dar inicio con el estudio. Entre los métodos cuantitativos se encuentran los métodos de muestreo, proyecciones de tendencias, evaluaciones de la estadística descriptiva e inferencial.

Cualquiera de estos tres métodos necesita herramientas que les otorguen la facultad de recolectar información de manera apropiada, siendo algunas de ellas las encuestas, entrevistas, censos y los formularios, entre otras.

A continuación se muestran en la figura 25, las principales fuentes de datos.

Figura 25. **Fuentes de datos**



Fuente: AAKER-KUMAR-DAY, Investigación de mercados. p.128.

Existe una amplia variedad de fuentes de información de las cuales dispone el investigador para la realización del estudio, pero el problema radica en la variabilidad de ellas respecto de factores como el tiempo, la calidad, el costo, la disponibilidad, etc.

Las fuentes de datos primarios se refieren a las que el propio investigador va a realizar, es decir, el propio estudio de mercado se convierte en una fuente primaria de datos, ya que son originales. Mientras que los datos secundarios son los que han sido realizados y recabados por otras personas, es decir es información que ya existe y puede ser utilizada por el investigador según su conveniencia.

4.1.7. Preparar y analizar los datos

Luego de haber concluido con la recolección de la información utilizando el método que mejor se adecue al estudio y las herramientas necesarias para obtener la información deseada, el investigador debe prepararse para un trabajo minucioso y de vital importancia, para el resultado exitoso del estudio de mercado realizado.

Contando con toda la información proporcionada por los profesionales del desarrollo de *software* se procede a revisar y preparar todos los datos de la investigación con exactitud, para evitar cualquier trastorno a la información original y que su preparación sea la adecuada para su análisis, mediante las técnicas estadísticas. Al tener toda la información organizada se da inicio con el análisis, el cual le permitirá al investigador formular las conclusiones y las recomendaciones pertinentes.

El análisis de los datos es sumamente importante, ya que un mal análisis o interpretación de datos puede hacer inservible todo el trabajo realizado hasta este punto por el investigador, ya que esto conduce a conclusiones erróneas y recomendaciones desacertadas. Asimismo, el análisis de los datos tampoco puede rescatar un estudio de mercado mal formulado y desarrollado desde su origen, esa es la razón de la importancia de un análisis a profundidad sobre lo descubierto y también que el investigador tenga la capacidad de ver más allá de los datos tabulados.

4.1.8. Informar los resultados y proporcionar recomendaciones

En la fase final para el investigador radica una gran responsabilidad ya que debe de saber transmitir lo descubierto gracias al estudio de mercado, explicarlo claramente y lograr que el esfuerzo del estudio sea aprovechado proporcionando información útil.

El investigador deberá realizar una impecable redacción en el informe final de los resultados, una formulación sólida de las conclusiones a los objetivos trazados al diseñar el estudio.

Es aconsejable incluir en la presentación gráficos e imágenes que despierten el interés a las personas; la redacción debe ser la adecuada para evitar confusiones al estar leyendo los resultados.

El investigador debe también buscar la manera de que toda la información que exponga se encuentre relacionada, es decir evitar referirse a los resultados de manera aislada, sino recordar que al igual que en el diseño, todas las piezas deben de encajar fácilmente.

4.2. Marco práctico

Luego de tener los fundamentos brindados por la investigación preliminar conceptual, se procede a la planificación y diseño del estudio de mercado apropiado a lo que el presente trabajo de graduación necesita para puntualizar lo investigado, además del valor agregado que le proporciona.

4.2.1. Propósito

En la actualidad la Internet ha pasado a formar parte del diario vivir de un gran porcentaje de la población del mundo, incluso gracias a las actividades económicas que se desarrollan sobre ella se ha hablado de una nueva economía sobre la Internet, lo cual demuestra el gran impacto que esta red tiene sobre sus usuarios. Una gran cantidad de información es enviada, recibida y mostrada diariamente alrededor del mundo, utilizando para ello aplicaciones para la Internet.

Debido a este aumento de información en la Internet, los usuarios han encontrado una gran necesidad de nuevas formas para organizar y mostrar dicha información en sus computadoras, las cuales no limiten las aplicaciones que utilizan por ejecutarse sobre la Internet, sino que por el contrario, les ofrezca mayor interacción, usabilidad y mejor acceso multimedia.

A través de este trabajo de graduación se busca brindar a la comunidad de desarrolladores de aplicaciones para la Internet y los interesados en este campo, un nuevo paradigma en el desarrollo de aplicaciones para la Internet, el cual presenta al usuario una forma más amigable de interactuar con la información, mejor acceso multimedia y facilidad en el desarrollo de dichas aplicaciones, sin eliminar ninguna de las capacidades ya existentes de las

aplicaciones tales como el acceso a distintas bases de datos, el uso de servicios *web* y de plantillas de presentación.

Con base en este objetivo se fundamenta el propósito del presente estudio de mercado de la siguiente manera: brindar información respecto del uso que se le está dando a este paradigma en las empresas de desarrollo de *software* y en las áreas de informática de las empresas en Guatemala, para conocer cuál es la aceptación de este tipo de aplicaciones en el mercado nacional, y así conocer el potencial que tiene Guatemala para el desarrollo de *software* de vanguardia.

4.2.2. Objetivos

Teniendo el propósito del estudio definido, el cual consiste fundamentalmente en conocer el panorama actual de las Aplicaciones Enriquecidas para Internet en Guatemala, se han determinado los siguientes objetivos para el estudio de mercado:

- Objetivo general

Establecer la situación tecnológica que poseen las empresas en la ciudad de Guatemala a nivel de desarrollo *web*.

- Objetivos específicos

  - Identificar la razón de selección de las Aplicaciones Enriquecidas de Internet en la realización de proyectos.

- Determinar la participación de las Aplicaciones Enriquecidas de Internet en el ámbito del desarrollo para Internet en Guatemala.
- Identificar las fortalezas y debilidades de este paradigma.

4.2.3. Diseño

Habiendo definido tanto el propósito como los objetivos de este estudio de mercado, lo que resta es diseñar las maneras en que logrará alcanzar los objetivos fijados y cumplir con el propósito establecido.

El diseño se encuentra conformado por la determinación de los sujetos de estudio, la recolección de la información necesaria y el diseño de la herramienta a utilizar; estos se definen a continuación.

4.2.3.1. Sujetos de estudio

Los sujetos de estudio son el conjunto de personas evaluadas para la realización del estudio de mercado, es a quienes se dirige el estudio que se ejecuta, conociendo de esta manera sus opiniones, puntos de vista, críticas, etc.

En este caso, los sujetos de estudio son los desarrolladores de *software*. Ellos poseen el conocimiento y la experiencia apropiada para responder a la encuesta de las Aplicaciones Enriquecidas para Internet, siendo la información que ellos proporcionen, altamente valiosa y recolectada de manera apropiada, dará la información necesaria para alcanzar los objetivos establecidos.

Los sujetos de estudio han sido seleccionados basados en las siguientes características:

- Desarrolladores de *software* que actualmente estudien carreras afines a informática, entre ellas: Ingeniería en Ciencias y Sistemas y Licenciatura en Informática, involucrados en proyectos de desarrollo de *software.*

- Desarrolladores de *software* que cuenten con estudios universitarios completos en el área de informática, que se encuentren laborando en el campo del desarrollo de *software* e involucrados en proyectos de desarrollo de *software*.

Los sujetos de estudio, son líderes potenciales en la toma de las decisiones sobre los paradigmas que van a utilizarse en los proyectos de *software* en los que se vean involucrados. Además de esto se necesitan conocimiento y experiencia entrelazados, no basta solamente con la experiencia construida únicamente con base en conocimiento empírico, sino que también posean un conocimiento y formación universitaria que los respalde, por lo que con esta segmentación se busca abarcar al sujeto de estudio idóneo que provee la información necesaria, para dar el valor agregado buscado a través de los objetivos del estudio de mercado.

4.2.3.2. Recolección de datos

Teniendo como premisa el propósito de la investigación, se ha decidido utilizar el método cualitativo, ya que este permite conocer la percepción de los desarrolladores de *software* que establecerán no únicamente la aceptación/rechazo de las Aplicaciones Enriquecidas para Internet en diversos sectores económicos del ámbito guatemalteco, sino sus fortalezas y

debilidades. Se optó por el uso de una encuesta diseñada de manera estratégica que revele las respuestas a los objetivos trazados de manera general y específica.

Se ha diseñado una encuesta *web,* debido a que se determinó que es la mejor manera de acercamiento a los sujetos de estudio y obtener la información en un tiempo óptimo, con la utilización de recursos reducida y amigable con el sujeto de estudio. La conexión entre la encuesta y el sujeto de estudio es a través del envío de la dirección de acceso a la encuesta mediante un correo electrónico, proveyendo de esta manera un fácil acceso a los sujetos de estudio que se desean encuestar. Esta modalidad de encuesta *web* disminuye radicalmente el tiempo que normalmente toma realizarlo de manera personal y además previene la negativa del sujeto de estudio a responder la encuesta debido al tiempo disponible que este posea.

4.2.4. Diseño de la encuesta

La encuesta se encuentra conformada por cuatro preguntas generales, posteriormente contiene dos secciones de dos preguntas cada una, la primera está dirigida para quienes han utilizado las Aplicaciones Enriquecidas para Internet, y la segunda sección para aquellos que no las han utilizado.

Las cuatro preguntas de estas dos secciones se diseñaron abiertas, para que el encuestado indique sus respuestas sin limitantes, permitiendo así tener una perspectiva clara de lo que se está preguntando y observando al final el total de los encuestados, si existe una coincidencia en sus opiniones o existen discrepancias entre las mismas.

4.3. Análisis y resultados

Posterior a la ejecución del trabajo de campo del estudio de mercado diseñado anteriormente, se procede a la tabulación de los resultados obtenidos de las encuestas para el correspondiente análisis, que conlleve a las conclusiones y recomendaciones pertinentes.

4.3.1. Tabulación

Se analizaron las respuestas de 100 personas, a quienes se envió la encuesta. Los resultados obtenidos son los siguientes:

1. ¿A qué sector pertenece la empresa para la que labora?

Se contabilizaron 23 personas laborando para el sector público y 77 laborando para el sector privado, como se muestra en la tabla VII.

Tabla VII. **Sector en que laboran los sujetos de estudio**

Opciones	Respuestas	Porcentaje respuestas
Público	23	23%
Privado	77	77%
Total	**100**	**100%**

Fuente: elaboración propia. Investigación de campo.

2. ¿Conoce usted las Aplicaciones Enriquecidas de Internet?

Se contabilizaron 76 personas que sí conocen el paradigma de la Aplicaciones Enriquecidas para Internet y 26 que no lo conocen, como se muestra en la tabla VIII.

Tabla VIII. **Conocimiento del paradigma RIA por parte de los sujetos de estudio**

Opciones	Respuestas	Porcentaje respuestas
Sí	74	74%
No	26	26%
Total	**100**	**100%**

Fuente: elaboración propia. Investigación de campo.

3. ¿Utiliza o ha utilizado las Aplicaciones Enriquecidas de Internet en el desarrollo de los proyectos en los que ha trabajado la empresa?

Mediante el conteo de respuestas se encontró que 56 personas utilizan o han utilizado el paradigma RIA en el ámbito laboral y 44 nunca, como se muestra en la tabla IX.

Tabla IX. **Uso del paradigma RIA por parte de los sujetos de estudio**

Opciones	Respuestas	Porcentaje respuestas
Sí	56	56%
No	44	44%
Total	**100**	**100%**

Fuente: elaboración propia. Investigación de campo.

4. Según la proporción de proyectos totales de la empresa, aproximadamente ¿en cuántos de los proyectos ha utilizado las Aplicaciones Enriquecidas de Internet?

En la tabla X se muestra la tabulación de los porcentajes de participación que posee el uso del paradigma RIA, en los proyectos en los que se ven involucrados los sujetos de estudio.

Tabla X. **Uso del paradigma RIA en proyectos en los que participan los sujetos de estudio**

Opciones	Respuestas	Porcentaje respuestas
75% - 100%	15	15%
50% - 75%	11	11%
25% - 50%	7	7%
0% - 25%	67	67%
Total	**100**	**100%**

Fuente: elaboración propia. Investigación de campo.

Las preguntas 5 y 6 fueron elaboradas para ser respondidas únicamente para los sujetos de estudio que utilicen Aplicaciones Enriquecidas para Internet.

Es importante indicar que el total de las respuestas no coincidirá, no solo con el total de los encuestados, sino tampoco con el total de los que sí han utilizado o utilizan las RIAs; esto debido a que un encuestado normalmente indicaba más de una razón del porqué las escoge.

5. ¿Por qué razón o razones escoge usted las Aplicaciones Enriquecidas de Internet para trabajar en los proyectos?

En la tabla XI se muestran los resultados de las razones que brindaron los sujetos de estudio respecto de por qué se utiliza la tecnología RIA en los lugares donde laboran. Estas fueron agrupadas en función de lo descrito, interpretando las respuestas, a modo de poder ordenarlas por grupos para no repetir razones que están contenidas dentro de un mismo objetivo.

Tabla XI. **Razones por las que se utiliza el paradigma RIA**

Opciones	Respuestas	Porcentaje respuestas
Facilidad de desarrollo	15	28%
Mantenimiento sencillo	5	9%
Portabilidad	3	6%
Rendimiento	13	24%
Experiencia de uso	18	33%
Total	**51**	**100.00%**

Fuente: elaboración propia. Investigación de campo.

6. ¿Qué tecnologías utilizan en la empresa para desarrollar las Aplicaciones Enriquecidas de Internet?

En la tabla XII se muestra la tabulación de las tecnologías utilizadas para la construcción de aplicaciones RIA.

Tabla XII. **Tecnologías utilizadas para el desarrollo de RIAs**

Opciones	Respuestas	Porcentaje respuestas
AJAX	24	44%
Microsoft Silverlight	17	31%
Oracle JavaFX	4	7%
Adobe Flash Builder	10	18%
Total	**55**	**100.00%**

Fuente: elaboración propia. Investigación de campo.

Las preguntas 7 y 8 fueron diseñadas para ser respondidas únicamente por aquellos sujetos de estudio que no hayan utilizado las aplicaciones enriquecidas para Internet.

Es importante indicar que el total de las respuestas no coincidirá no solo con el total de los encuestados global, sino tampoco con el total de los que sí han utilizado o utilizan las RIAs, esto debido a que un encuestado normalmente indicaba más de una razón del porqué no las escoge.

7. ¿Por qué razón o razones no escoge usted las Aplicaciones Enriquecidas de Internet para trabajar en los proyectos?

En la tabla XIII se muestra la tabulación de las razones por las cuales no se utiliza el paradigma RIA en los desarrollos en los que participan los objetos de estudio.

Tabla XIII. **Razones por las que no se utiliza el paradigma RIA**

Opciones	Respuestas	Porcentaje respuestas
Desconocimiento	19	33%
Poco tiempo para desarrollo	7	12%
No aplica al negocio	25	44%
Curva de aprendizaje	6	11%
Total	**57**	**100%**

Fuente: elaboración propia. Investigación de campo.

8. ¿Qué otro paradigma utilizan en su empresa en vez de las aplicaciones enriquecidas de internet?

En la tabla XIV se muestran las respuestas de qué otros paradigmas son utilizados en lugar del paradigma RIA.

Estos paradigmas fueron agrupados en función de la estructura principal de la aplicación, es decir por ejemplo, que las aplicaciones construidas con AJAX ASP.NET, o con RichFaces se encuentran dentro del grupo de las Aplicaciones *Web* Asíncronas no Enriquecidas.

Tabla XIV. **Otros paradigmas utilizados en el desarrollo de aplicaciones *web***

Opciones	Respuestas	Porcentaje respuestas
Aplicaciones *web* asíncronas no enriquecidas	25	43%
Aplicaciones *web* síncronas	22	38%
CMS	4	7%
Oracle Forms	7	12%
Total	**58**	**100.00%**

Fuente: elaboración propia. Investigación de campo.

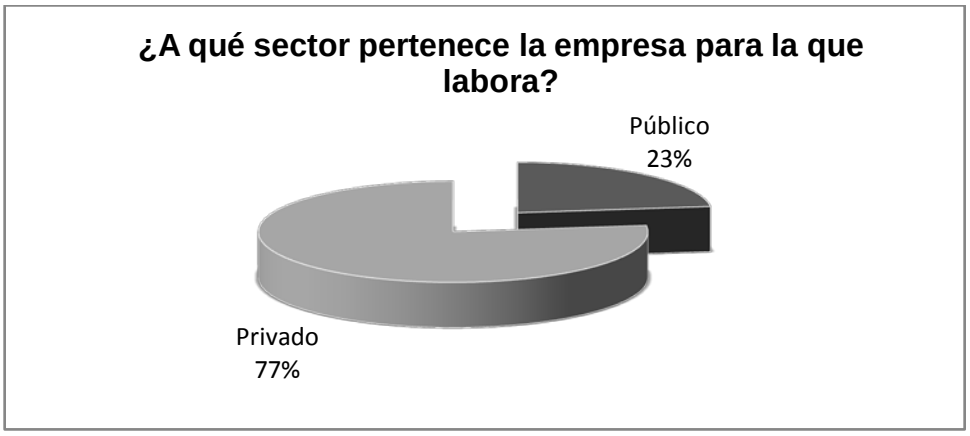
4.3.2. Análisis e Interpretación

En función de los resultados tabulados anteriormente, se ha realizado un análisis sobre los mismos, el cual se presenta a continuación:

Al analizar los resultados de las encuestas se encuentra que el 77% de los encuestados labora para empresas del sector privado, mientras un 23% corresponde a trabajadores pertenecientes al sector público, como se muestra en la figura 26.

Esta segmentación permitirá mostrar con detalle, la opinión de cada sector en cuanto a las Aplicaciones Enriquecidas para Internet.

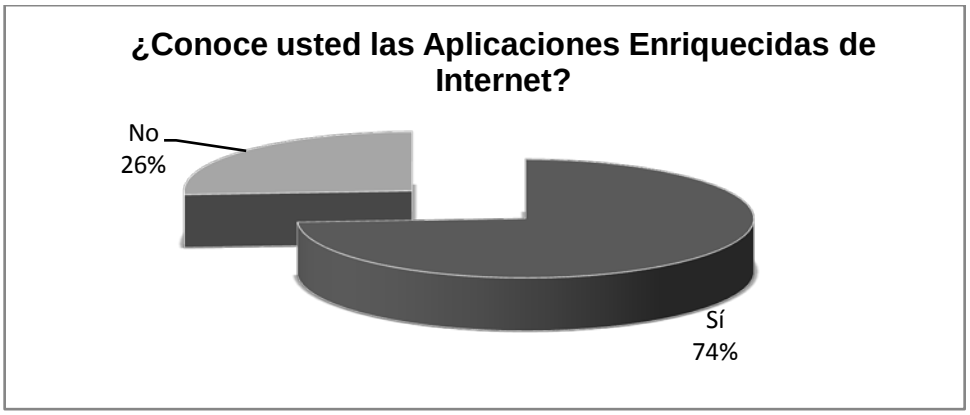
Figura 26. **Sector en que laboran los sujetos de estudio**



Fuente: elaboración propia. Investigación de campo.

Como parte del estudio se desea evaluar el conocimiento del paradigma en la muestra seleccionada. El 74% de los individuos encuestados conoce qué son y para qué sirven las Aplicaciones Enriquecidas para Internet, dejando así un 26% que no conoce el paradigma RIA, como se muestra en la figura 27.

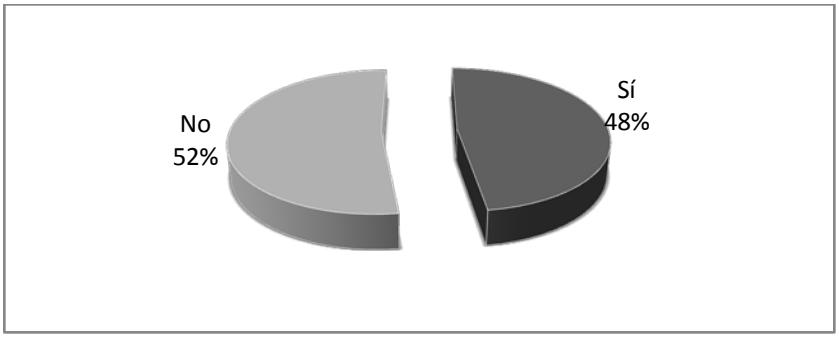
Figura 27. **Conocimiento del paradigma RIA por parte de los sujetos de estudio**



Fuente: elaboración propia. Investigación de campo.

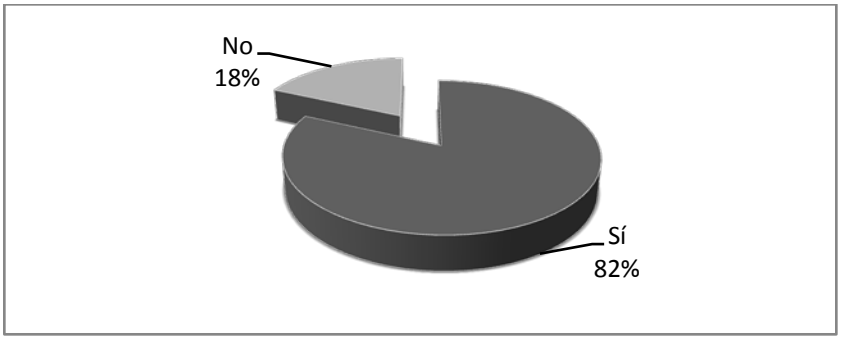
Analizando los resultados, se encuentra que el 74% de la muestra conoce el paradigma de las Aplicaciones Enriquecidas para Internet, esto representa más de tres cuartos de la muestra seleccionada, lo cual indica que una cantidad bastante aceptable de sujetos de estudio conoce el tema. Se puede profundizar en el análisis, para encontrar qué sector posee la mayor cantidad de la muestra que conoce respecto del paradigma RIA, por lo que se procede a analizar las figuras 28 y 29.

Figura 28. **Conocimiento y desconocimiento del paradigma RIA en el sector público**



Fuente: elaboración propia. Investigación de campo.

Figura 29. **Conocimiento y desconocimiento del paradigma RIA en el sector privado**



Fuente: elaboración propia. Investigación de campo.

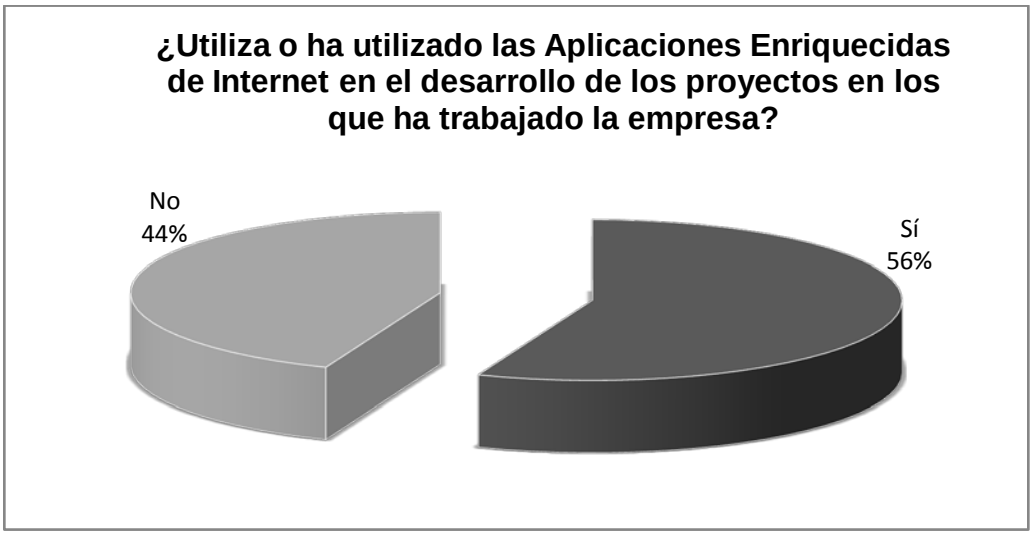
Al analizar los porcentajes de conocimiento y desconocimiento entre ambos sectores, se encuentra una diferencia bastante marcada, en donde el sector privado posee un conocimiento del paradigma RIA del 82% del total de la muestra que pertenece a dicho sector, contra un 48% del total de la muestra, que pertenecen al sector público.

Los porcentajes de sujetos de estudio en cada sector que conocen del paradigma RIA, indican que el desconocimiento de la muestra en el sector privado es del 18% y del sector público es del 52%; esto significa que las personas involucradas en los procesos de desarrollo de *software* en el sector privado se encuentran con una percepción distinta respecto de la vanguardia de paradigmas de aplicaciones *web*, colocando al sector privado como el grupo principal para el desarrollo de este tipo de aplicaciones en Guatemala.

A pesar de que el 74% de los encuestados conoce las Aplicaciones Enriquecidas para Internet, solamente el 56% de ellos ha utilizado o utiliza actualmente este paradigma. Ver figura 30. Esto quiere decir que un 18% conoce las Aplicaciones Enriquecidas para Internet, pero no las ha implementado nunca.

Al analizar estos porcentajes se puede interpretar que existe una diferencia del 12% entre el total de la muestra que sí utiliza el paradigma y los que no. Una diferencia tan pequeña quiere decir que el paradigma RIA empieza a formar parte de la predilección para la construcción de aplicaciones *web*, lo cual indica que el paradigma está siendo acogido y utilizado de buena manera, ya que ha logrado obtener un buen porcentaje del total de proyectos que se desarrollan dentro de la muestra seleccionada.

Figura 30. **Uso del paradigma RIA en proyectos en los que participan los sujetos de estudio**

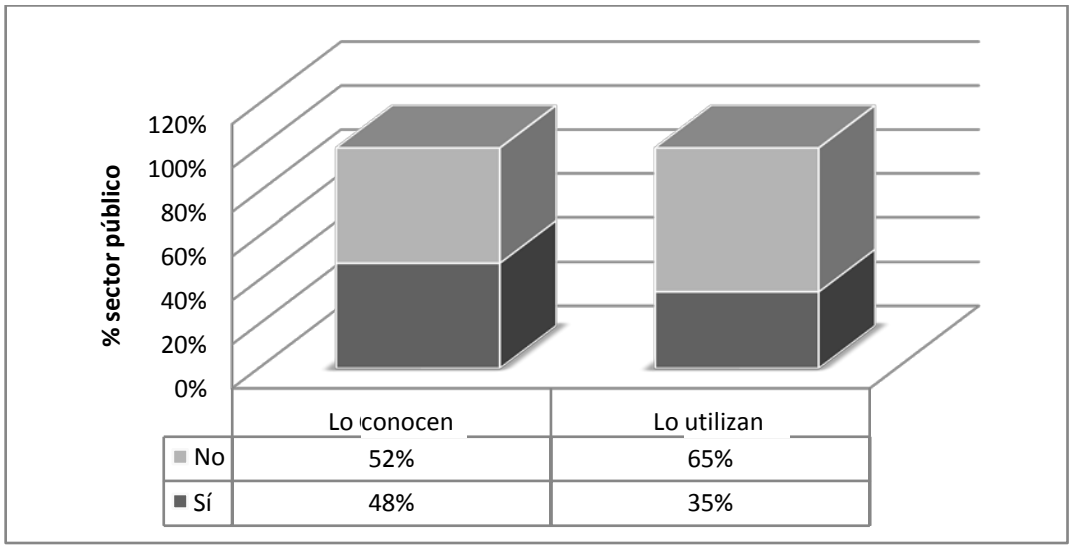


Fuente: elaboración propia. Investigación de campo.

Se procede ahora a analizar el impacto que están teniendo las RIAs en los sectores público y privado. Ver figuras 31 y 32.

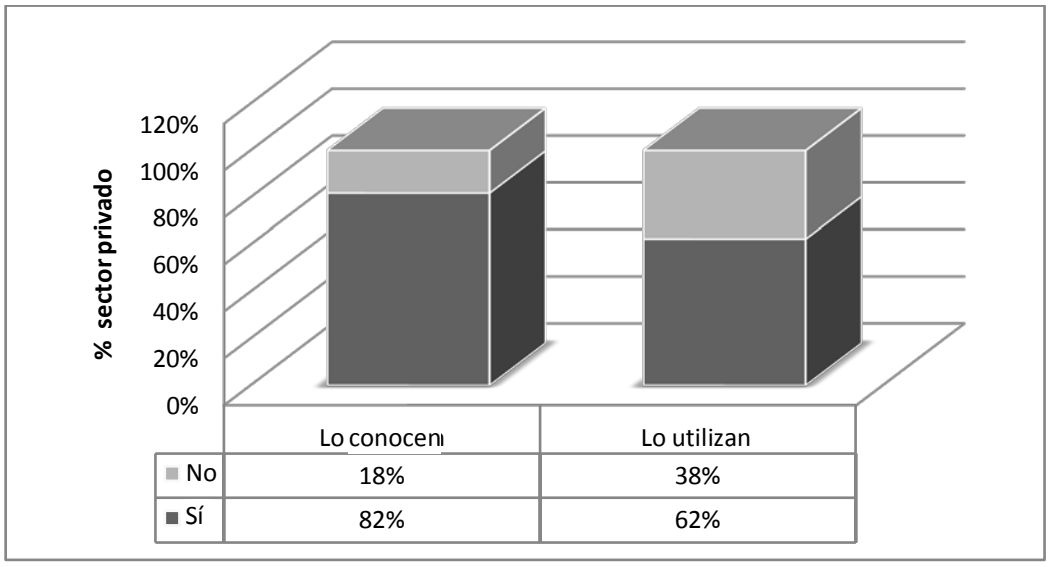
Al visualizar estas figuras se puede encontrar que existe un estado crítico en el uso del paradigma RIA en el sector público, debido a que es conocido por un 48% de su muestra pero únicamente el 35% lo ha aplicado; así como de igual forma se encuentra que, ni el conocimiento del paradigma ni el uso del mismo sobrepasan la mitad de su población. En cuanto al sector privado se tiene una vista más optimista respecto del paradigma RIA, puesto que de su muestra, el 82% lo conoce y el 62% lo está utilizando, lo cual indica que se está adoptando de mejor manera en este sector.

Figura 31. **Conocimiento y uso del paradigma RIA en el sector público**



Fuente: elaboración propia. Investigación de campo.

Figura 32. **Conocimiento y uso del paradigma RIA en el sector privado**



Fuente: elaboración propia. Investigación de campo.

Ya se ha establecido el estado de la adopción del paradigma RIA en ambos sectores y se ha encontrado que el sector público ha adoptado en menor medida el paradigma, mientras que el sector privado lo ha hecho bastante bien. A pesar de las diferencias entre el uso del paradigma entre ambos sectores se puede detectar cierto equilibrio a nivel general en la adopción del paradigma, puesto que la falta de uso que tiene en el sector público se ve compensado por su uso en el sector privado, superando de esta forma la mitad del total de individuos que utilizan o han utilizado el paradigma RIA del total de encuestados.

Conociendo el porcentaje de individuos que utilizan y que no utilizan el paradigma, se procede a encontrar en qué medida es utilizado el paradigma RIA en cada sector.

En cuanto al uso del paradigma RIA en los proyectos realizados, el mayor porcentaje (67%) se encuentra en el primer rango comprendido del 0% al 25%. Este porcentaje, a pesar de ser el más alto en la encuesta, representa únicamente a 23 sujetos de estudio, quienes han utilizado el paradigma RIA en un intervalo antes indicado, del total de proyectos que trabajan.

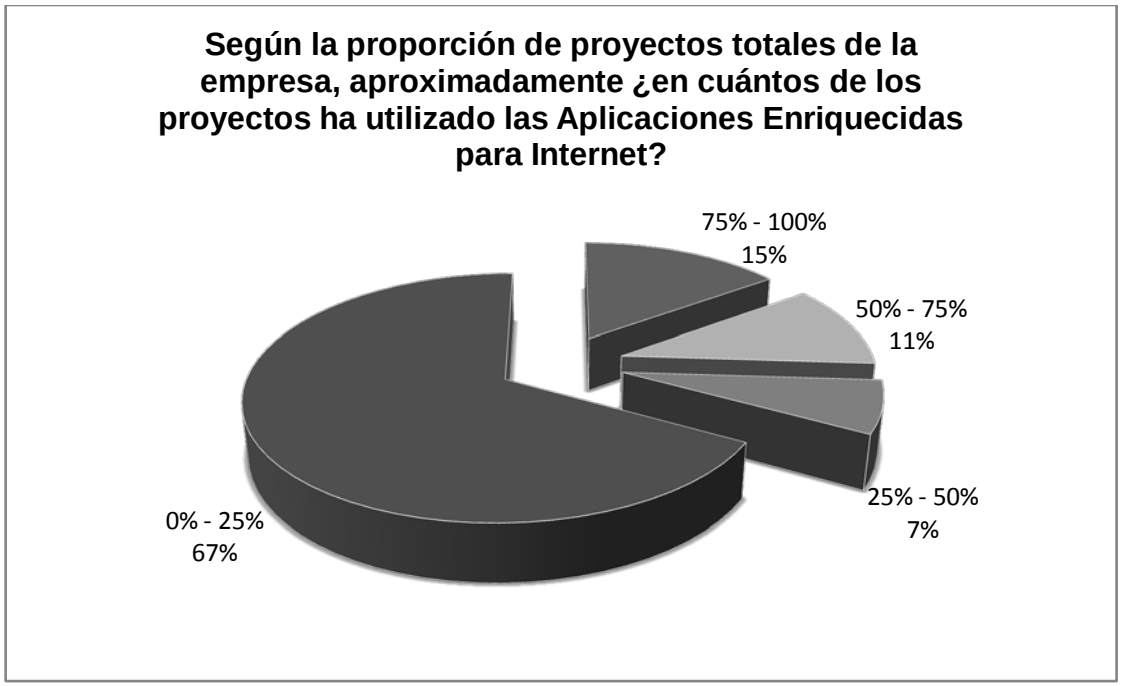
El siguiente rango es el comprendido entre el 75% y el 100% de los proyectos totales de las empresas en las que laboran, con un 15%. Este es un porcentaje pequeño comparado con el primer lugar, con una diferencia del 52%, lo cual indica que más de la mitad de la muestra utiliza este paradigma. Los individuos que constituyen este porcentaje, son los que pueden representar y apoyar de mejor manera el paradigma dentro de la industria del *software* en Guatemala, ya que ellos cuentan con la capacidad para la construcción de *software* de vanguardia y por el nivel de uso que dan a este paradigma, han generado también nuevo conocimiento al respecto.

Un 11% de los encuestados utilizan las RIA entre un 50% y 75% de los proyectos totales en los que trabaja, mientras que un 7% las utiliza entre un 25% y 50%.

En estos porcentajes se ve reflejado que la cantidad de sujetos de estudio que posee mayor experiencia en el uso y aplicación del paradigma RIA, conforman un porcentaje reducido; esto representa la brecha que existe en cuanto a la participación de este paradigma en los proyectos de *software*, en donde el mayor porcentaje que lo utiliza está en el rango de quienes lo aplican en menor cantidad de proyectos.

Estos rangos se muestran graficados en la figura 33.

Figura 33. **Porcentajes de uso del paradigma RIA**

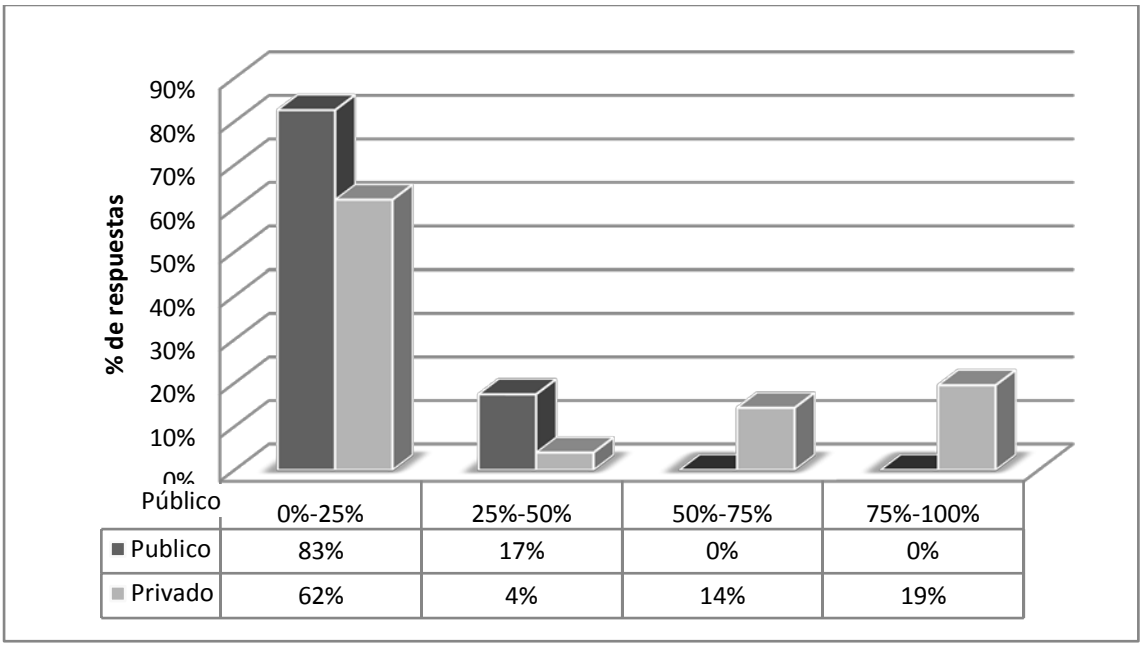


Fuente: elaboración propia. Investigación de campo.

Al comparar los porcentajes de uso del paradigma RIA entre el sector público y privado, se visualiza que la implementación del paradigma RIA es mayor en el rango del 0% al 25% en donde se tiene el 83% para el sector público y el 62% para el sector privado, esto quiere decir que la diferencia del 22% existente entre ambos sectores, debe de estar distribuida en otros rangos.

En el rango del 25% al 50% se encuentra que el sector público tiene una participación del 17% y el sector privado del 4%. En el rango del 50% al 75% se encuentra un 0% para el sector público y un 14% para el sector privado y finalmente en el rango del 75% al 100% se encuentra un 0% para el sector público y un 19% para el sector privado, como se muestra en la figura 34.

Figura 34. **Comparación del uso del paradigma RIA entre el sector público y el sector privado**

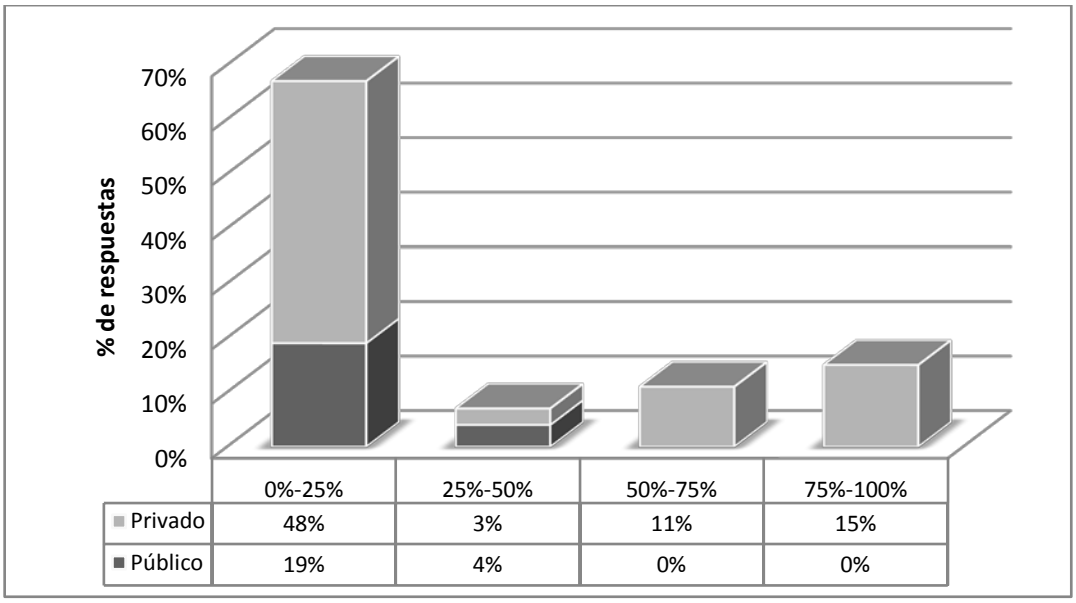


Fuente: elaboración propia. Investigación de campo.

Al analizar la figura 34 se puede ver que el uso del paradigma RIA por el sector público es muy bajo, lo cual viene a reforzar el comportamiento encontrado anteriormente; en donde se mostraba que el sector público poseía un bajo conocimiento respecto de este paradigma. Se puede ver también que el sector privado posee su segundo porcentaje más alto entre el uso del paradigma en la mayoría de sus proyectos, lo cual respalda también lo descrito anteriormente, en donde se indica que el sector privado es quien provee de mayor cantidad de *software* utilizando el paradigma RIA.

En la figura 35 se puede apreciar el porcentaje de participación que posee el sector público y el sector privado en cada uno de los rangos, en los cuales indican qué porcentaje de proyectos que elaboran, utilizan el paradigma RIA.

Figura 35. **Participación de uso del paradigma RIA entre el sector público y sector privado**



Fuente: elaboración propia. Investigación de campo.

Al analizar la gráfica se encuentra que la participación del sector público entre los rangos de uso es poco representativa, puesto que si no se tomara en cuenta a dicho sector, el comportamiento sería el mismo. Ahora bien, dado que el sector público es pequeño en comparación al sector privado, la representatividad del sector público debe ser tomada en cuenta, aunque no comparada directamente con el sector privado.

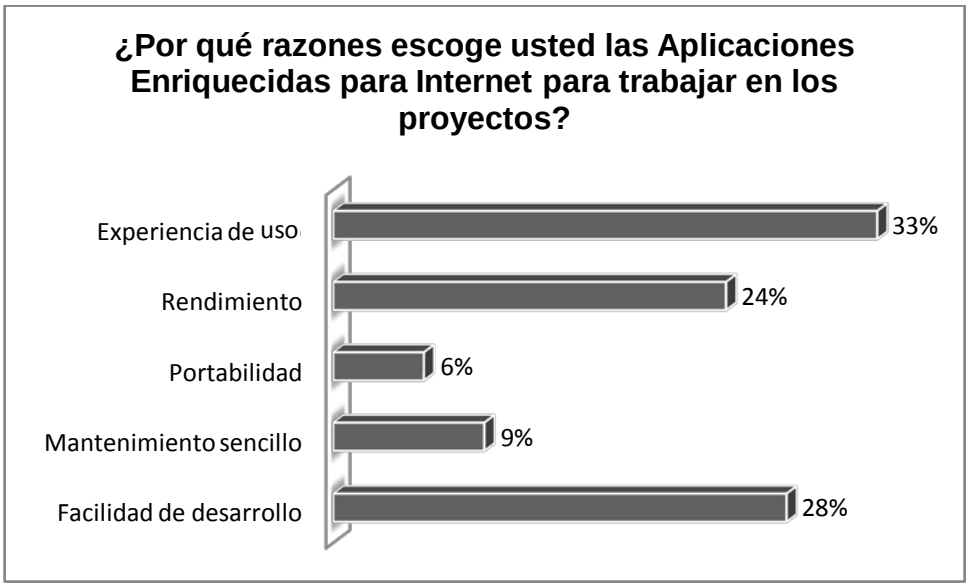
Dentro de la encuesta se incluyó una sección dirigida a las personas que sí han utilizado o utilizan el paradigma RIA; ahora se procede a analizar los resultados obtenidos.

Es importante indicar que el total de las respuestas no coincidirá no solo con el total de los encuestados ni tampoco con el total de los que sí han utilizado o utilizan las RIA, esto debido a que un encuestado normalmente daba más de una razón de por qué utilizaba este paradigma.

Se deseaba conocer por qué razones el paradigma RIA es seleccionado para el desarrollo de aplicaciones *web*, la lista se conformó por cinco elementos que fueron obtenidos de las respuestas a la encuesta, los cuales fueron: la experiencia de uso que dan al usuario, el rendimiento, la portabilidad, la facilidad de mantenimiento y la facilidad para desarrollar aplicaciones. Estas razones se encuentran graficadas con sus respectivos porcentajes en la figura 36.

En primer lugar de las razones por las cuales se utiliza el paradigma RIA se encuentra la experiencia de uso que estas brindan a los usuarios, colocándola entre la preferencia de los encuestados con un 33%, lo cual indica que esta es la característica principal con la cual se identifica el paradigma RIA entre las personas involucradas en el proceso de creación del *software*.

Figura 36. **Razones por las que se utiliza el paradigma RIA**



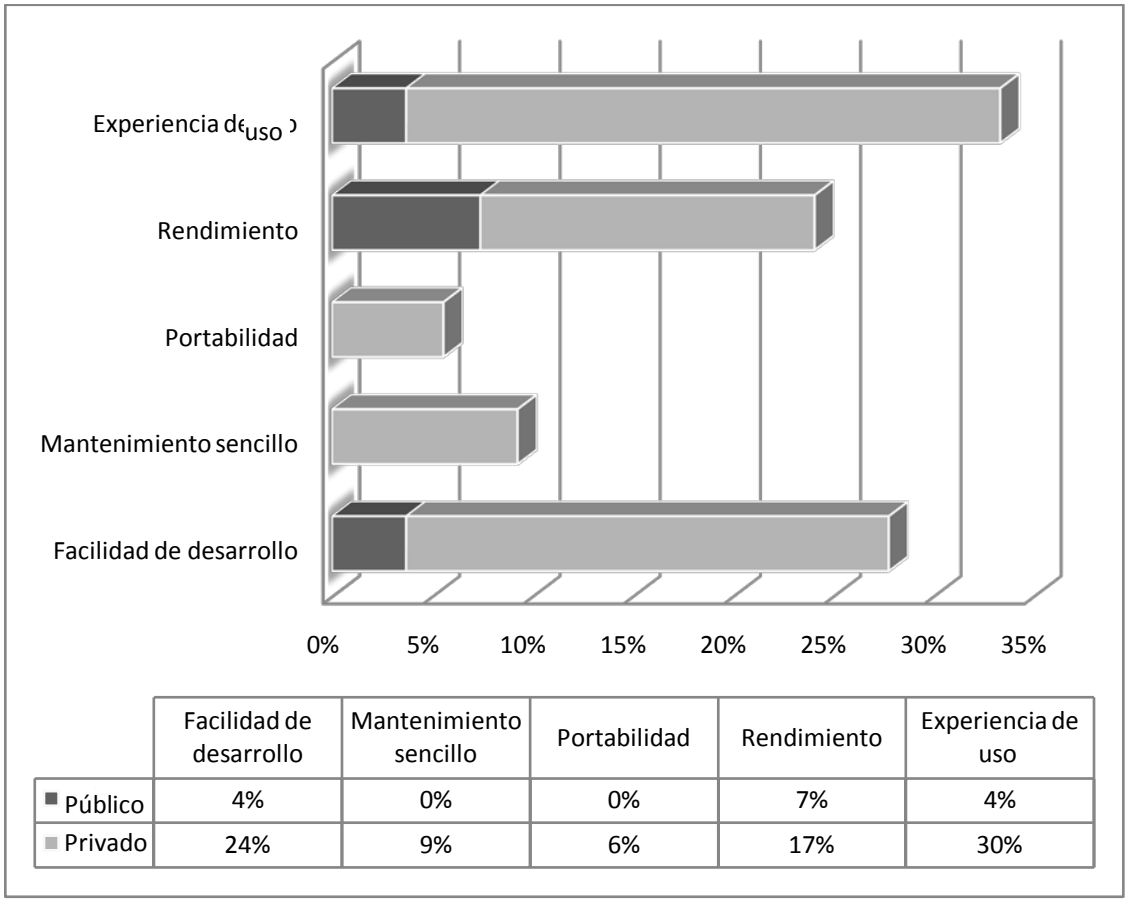
Fuente: elaboración propia. Investigación de campo.

Cabe destacar la característica “Rendimiento”, la cual se encuentra ubicada en el tercer lugar con un 24%, esta es una de las más representativas del paradigma RIA, además de la experiencia de uso.

Siendo el rendimiento una de las tres primeras razones por la cuales se utiliza este paradigma, esto indica que se está comprendiendo cuáles son los objetivos que tienen las Aplicaciones Enriquecidas para Internet y que se están utilizando por las razones correctas, y no únicamente porque “esté de moda”. En segundo lugar se encuentra la facilidad del desarrollo con un 28%, lo cual indica que los individuos perciben este paradigma como una forma fácil de crear aplicaciones *web*, lo cual también representa un gran avance en cuanto a la adopción del paradigma, puesto que significa que se sienten cómodos utilizando el paradigma RIA.

En la figura 37 se muestra el listado de razones por las que se utiliza el paradigma RIA, indicando en cada razón el porcentaje de participación que tiene el sector público y privado. En esta gráfica se puede apreciar las razones principales por las que el sector público utiliza el paradigma.

Figura 37. **Composición de razones por las que se utiliza el paradigma RIA en el sector público y el sector privado**



	Facilidad de desarrollo	Mantenimiento sencillo	Portabilidad	Rendimiento	Experiencia de uso
Público	4%	0%	0%	7%	4%
Privado	24%	9%	6%	17%	30%

Fuente: elaboración propia. Investigación de campo.

Al ser el sector privado quien posee la mayor cantidad de implementaciones de proyectos utilizando este paradigma, es lógico pensar que a esto se debe la existencia de razones en las cuales todo el porcentaje pertenece a este sector.

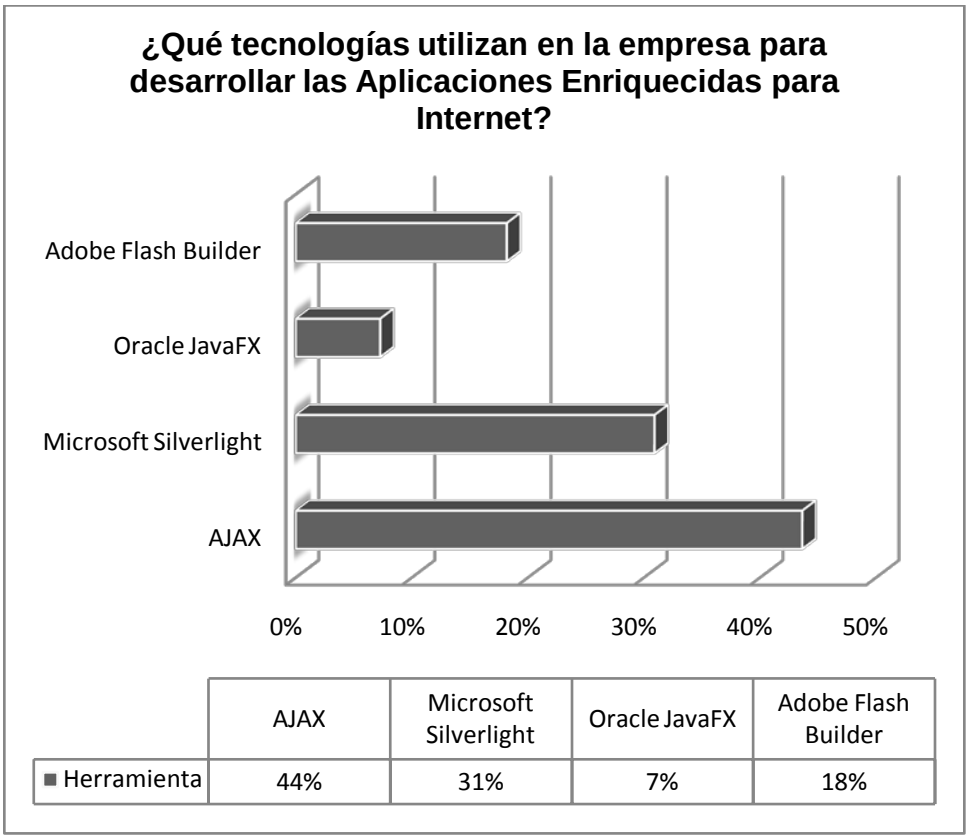
En cuanto a la evaluación de qué herramienta es más utilizada para la creación de proyectos utilizando tecnologías RIA, se encontraron cuatro opciones al analizar las respuestas de las encuestas, en donde con un 44%, AJAX es el primer lugar del listado. Ver figura 38.

En segundo lugar se encuentra Microsoft Silverlight con un 31%, en tercer lugar se encuentra Adobe Flex/Flash Builder (18%) y por último Oracle JavaFX (7%), dejando claramente marcado el uso de AJAX como la herramienta preferida para la construcción de Aplicaciones Enriquecidas para Internet, ver figura 38.

La primera posición que ocupa AJAX puede deberse a que esta herramienta fue la que dio inicio y permitió la comunicación asíncrona en las aplicaciones web, sin necesidad de utilizar ningún *plug-in*, sino únicamente las herramientas que se encuentran dentro de cualquier navegador de Internet.

Esto representó una fortaleza que fue llevada a otras herramientas como por ejemplo ASP.NET y RichFaces, las cuales pudieron ofrecer interfaces de usuarios más atractivas y aprovechar mejor la transmisión de datos en la red. A pesar de que AJAX fue quien habilitó este comportamiento en la comunicación cliente-servidor, este no puede ofrecer por sí solo una interfaz enriquecida, aunque recientemente con la liberación de HTML5 se podrá construir verdaderas aplicaciones enriquecidas utilizando AJAX, HMTL5 y CSS3.

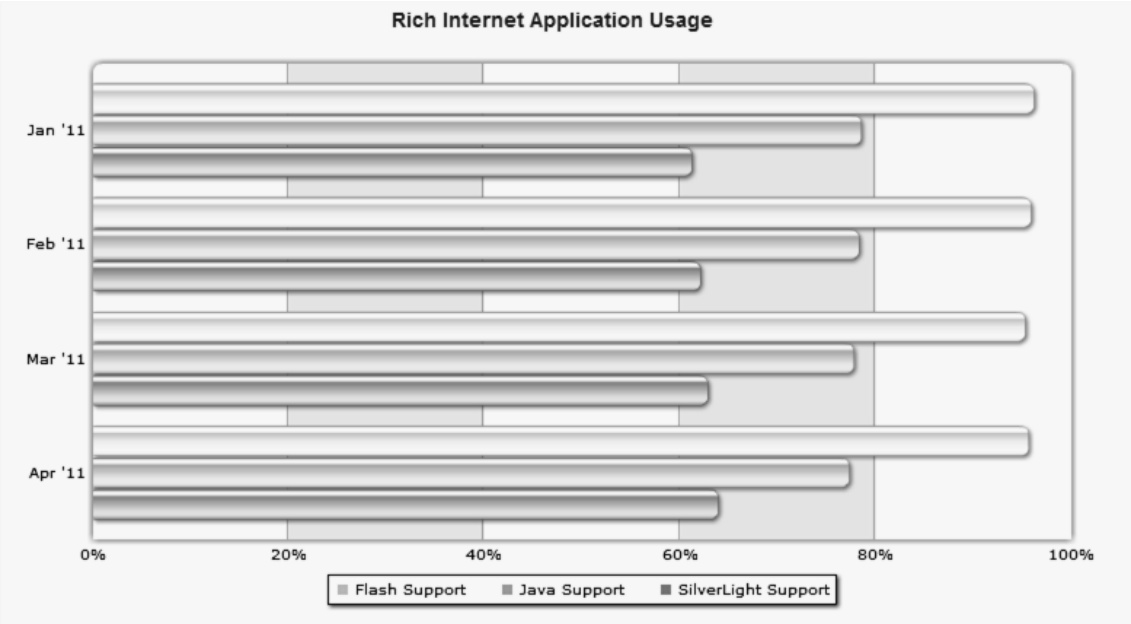
Figura 38. **Tecnologías para el desarrollo de RIAs**



Fuente: elaboración propia. Investigación de campo.

Al analizar el comportamiento del mercado de las Aplicaciones Enriquecidas para Internet, se encuentra que la tendencia de penetración en dicho mercado la domina Adobe Flash, con un promedio en el primer cuatrimestre del 2011 de 95.74%, como lo muestra la figura 39, obtenida del sitio de estadísticas statowl.com; dejando a Microsoft Silverlight en el tercer puesto con una penetración de 62.49%, en el mismo período y Java FX en el segundo lugar con un 77.98%.

Figura 39. **Uso de Aplicaciones Enriquecidas para Internet**



APPLICATION	TOTAL AVG	JAN '11	FEB '11	MAR '11	APR '11
Flash Support	95.74%	96.18%	95.86%	95.27%	95.65%
Java Support	77.98%	78.54%	78.26%	77.80%	77.31%
SilverLight Support	62.49%	61.19%	62.07%	62.82%	63.92%

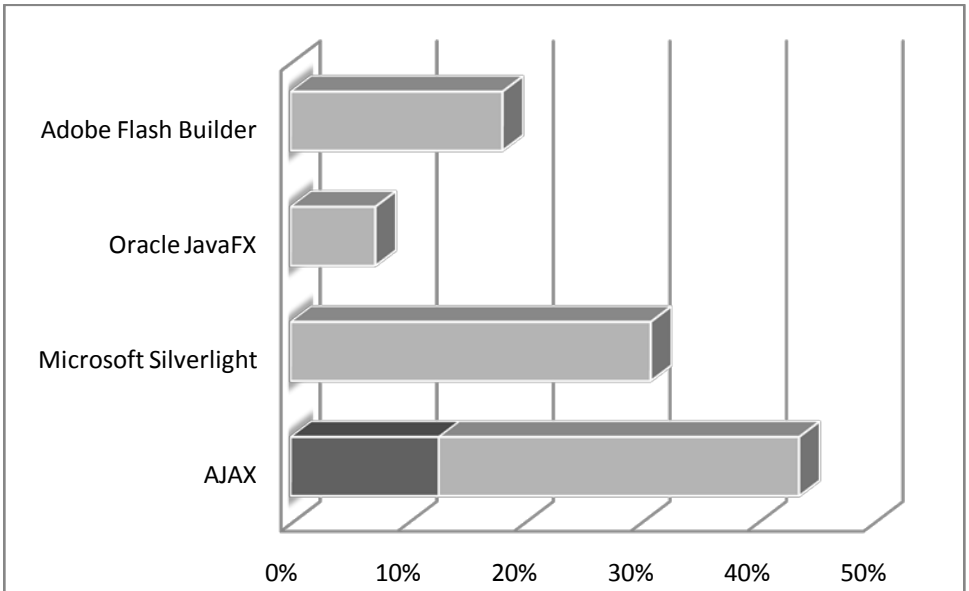
Fuente: www.statowl.com. 23 de junio de 2011.

Este comportamiento puede ser debido a que Adobe es conocido en el ámbito informático más por su *suite* para generación de contenido gráfico y multimedia Adobe CS5, que por proveer herramientas para el desarrollo y creación de *software*.

Otro factor que influye es que en Guatemala aún no se cuenta con un ente que provea de cursos y que informe respecto de certificaciones para los desarrolladores de *software* en alguna tecnología de desarrollo de *software* Adobe; mientras que por el otro lado, se tiene un caso totalmente contrario para Microsoft, en donde existen varias opciones para el estudio de las herramientas

que ofrece y además se encuentran con una mayor difusión de certificaciones para tecnologías Microsoft, por lo que las personas tienen más facilidades para involucrarse en el desarrollo y uso de herramientas Microsoft por medio de constantes capacitaciones o por la facilidad que en algún momento pueda ser conseguir asesoramiento o capacitación al respecto.

Figura 40. **Distribución del uso de herramientas para la construcción de RIA entre el sector público y el sector privado**



	AJAX	Microsoft Silverlight	Oracle JavaFX	Adobe Flash Builder
Público	13%	0%	0%	0%
Privado	31%	31%	7%	18%

Fuente: elaboración propia. Investigación de campo.

En la figura 40 se muestra la participación que tiene cada sector en cuanto al uso de herramientas para la construcción de Aplicaciones Enriquecidas para Internet. En esta figura se puede ver que la única herramienta utilizada por el

sector público es AJAX, con un 13% del total de las herramientas utilizadas, mientras que el sector privado posee una cantidad de usuarios igual entre AJAX y Microsoft Silverlight, con un 31%; luego Oracle JavaFX y Adobe Flash Builder, son utilizados únicamente por el sector privado en su totalidad.

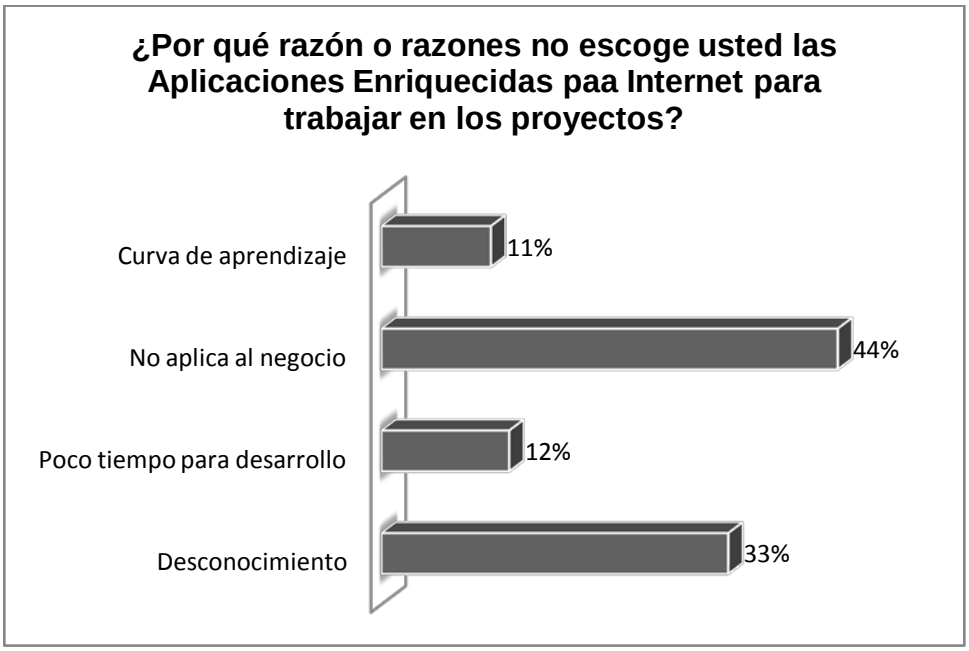
Al analizar la figura anterior se puede decir que debido a que el sector privado provee mayor cantidad de aplicaciones que utilizan el paradigma RIA, la mayoría de estas aplicaciones estarán construidas sobre tecnología AJAX o Microsoft, por lo que Guatemala tiene como fortaleza el poder ofrecer aplicaciones basadas en AJAX y tecnologías Microsoft en el sector privado, mientras que en el sector público, la mayoría de aplicaciones que utilizan el paradigma RIA estarán basadas en AJAX.

Dentro de la encuesta se incluyó una sección para conocer las opiniones de aquellos individuos que no han utilizado el paradigma RIA. A continuación se procede a hacer el análisis respectivo.

Es importante indicar que el total de las respuestas no coincidirá no solo con el total de los encuestados sino tampoco con el total de los que no han utilizado las RIA; esto debido a que un encuestado normalmente daba más de una razón de por qué nunca las utilizó.

Las razones por las cuales los desarrolladores encuestados no han escogido las Aplicaciones Enriquecidas para Internet para trabajar proyectos, son menos que las razones por las cuales son escogidas. Las razones y sus respectivas apariciones en las encuestas se muestran en la figura 41.

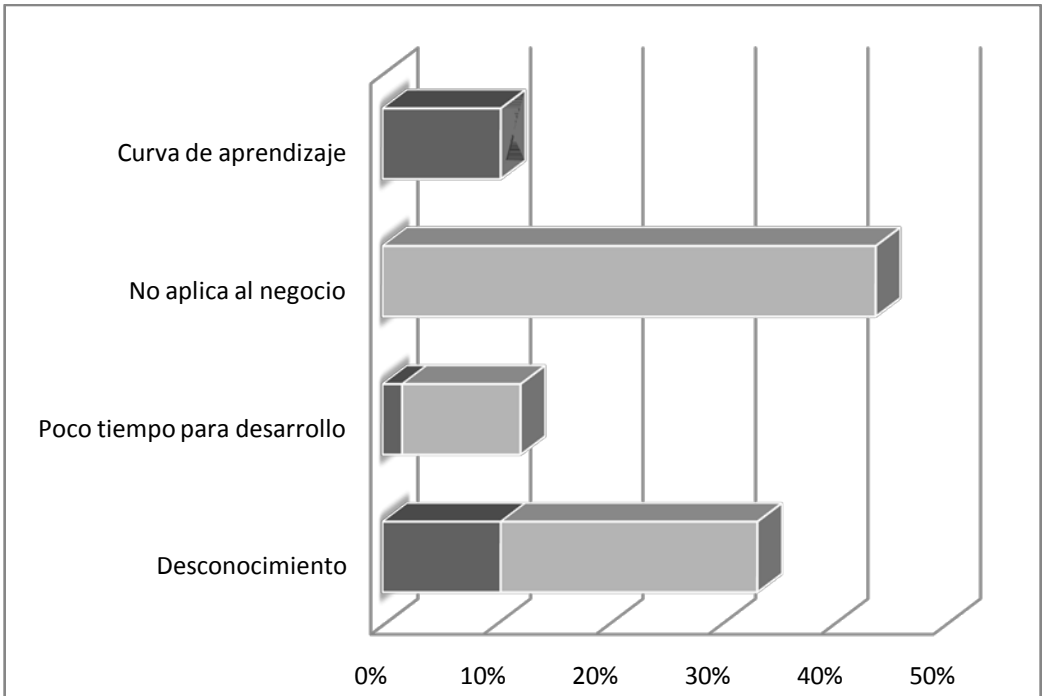
Figura 41. **Razones por las que no se utiliza el paradigma RIA**



Fuente: elaboración propia. Investigación de campo.

La no selección del paradigma RIA para la construcción de aplicaciones *web* se debe principalmente a que este paradigma no aplica a las necesidades del negocio, esta razón ocupa el 44% de todas las respuestas. Con un 33% la segunda razón es el desconocimiento que se tiene en relación con este paradigma; a estas dos razones las siguen el poco tiempo con que se cuenta para hacer un desarrollo con un 12%, y la curva de aprendizaje de un nuevo paradigma, con un 11%.

Figura 42. **Distribución de las razones por las que no se utiliza el paradigma RIA entre el sector público y el sector privado**



	Desconocimiento	Poco tiempo para desarrollo	No aplica al negocio	Curva de aprendizaje
Público	11%	2%	0%	11%
Privado	23%	11%	44%	0%

Fuente: elaboración propia. Investigación de campo.

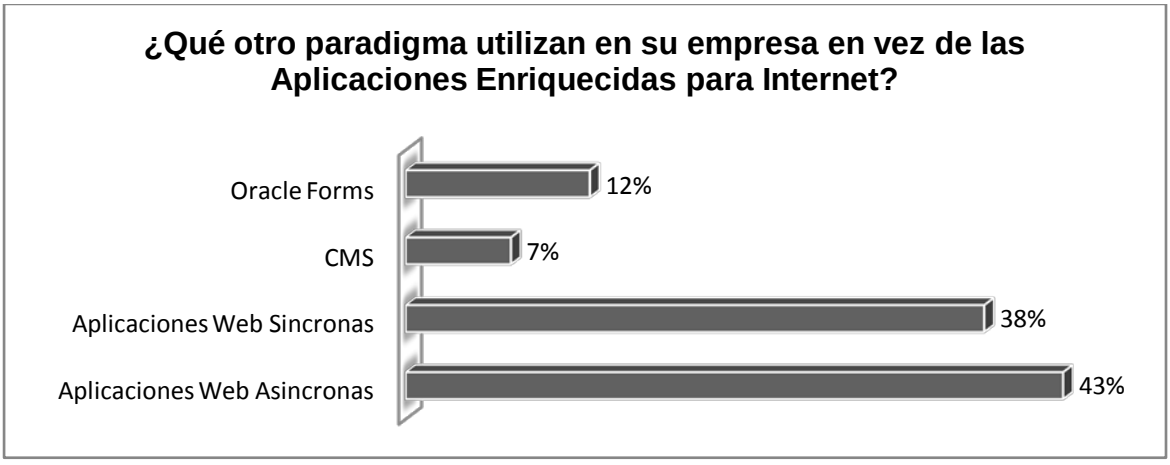
En la figura 42 se muestra la participación de cada sector exponiendo las razones por las que no se utiliza el paradigma RIA en la construcción de aplicaciones *web*. En este se observa que el sector privado comprende todo el porcentaje de la razón de no utilizar las RIA porque no aplica al negocio; esto puede deberse a la diversidad de campos que abarca el desarrollo de *software* en el sector privado. Al ver esta razón hay que recordar que el porcentaje que

comprende la cantidad de aplicaciones que no utilizan este paradigma en el sector privado es del 38% y sobre este porcentaje es que se aplica el 44% que comprende la razón de que no utilizan el paradigma RIA porque no aplica al negocio.

En el sector público se encuentran dos razones con el mismo porcentaje de incidencia. Estas son: la curva de aprendizaje para la adopción de un nuevo paradigma y su desconocimiento. Esto afecta en una escala mayor que en el sector privado, ya que en el sector público es un 65% de la muestra total que pertenece a este sector que no utiliza este paradigma, lo cual hace que estas razones sean más significativas que en el sector privado.

Como parte del estudio, se deseaba conocer qué paradigmas sustituyen al paradigma RIA, obteniendo como resultado la gráfica que se muestra en la siguiente figura.

Figura 43. **Otros paradigmas utilizados para la creación de aplicaciones *web***



Fuente: elaboración propia. Investigación de campo.

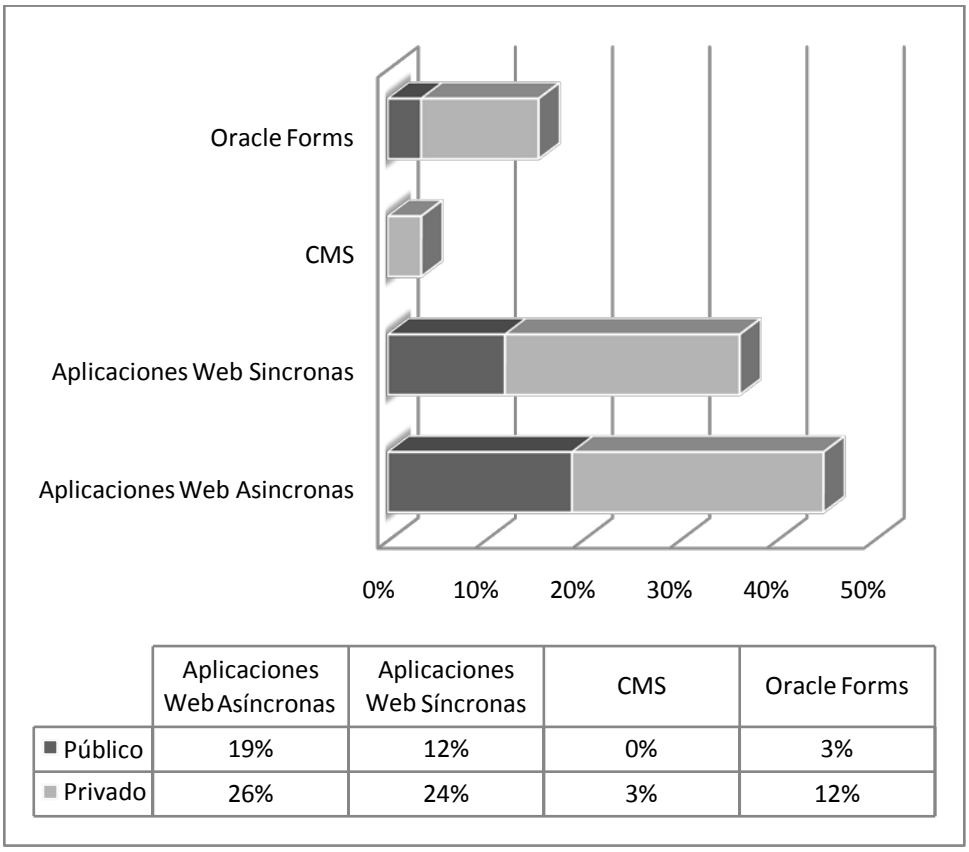
Las Aplicaciones *Web* Asíncronas No Enriquecidas se encuentran en primer lugar con un 43%; en segundo lugar, las aplicaciones *web* síncronas con un 38%; seguido por los Oracle Forms con el 12% y por último los CMS con un 7%, como se muestra en la figura 43.

El uso de Aplicaciones *Web* Asíncronas no Enriquecidas indica que los involucrados en el desarrollo de los proyectos de *software* han encontrado y utilizado los beneficios que la comunicación asíncrona brinda, por lo que estos pueden ser los usuarios potenciales del paradigma RIA, ya que al obtener el conocimiento empírico del funcionamiento de estas herramientas se podrá realizar una transición al paradigma RIA sin mayores dificultades.

En la figura 44 se muestra la participación que tienen los sectores público y privado, en los paradigmas que se utilizan en vez de las RIA. En este caso se encuentra un equilibrio en el uso del las Aplicaciones *Web* Asíncronas no Enriquecidas, en donde el sector público cuenta con 19% y el sector privado con un 26%, conformando así el total del paradigma que más uso tiene. A este lo siguen las Aplicaciones *Web* Síncronas con una participación del sector privado del 24% y del público con un 12%; seguido se encuentran los Oracle Forms con una participación del sector público del 3% y del sector privado con el 12%; finalmente los CMS, con un 3% para el sector privado.

En función del estado de las aplicaciones *web* asíncronas se encontró que ambos sectores utilizan este tipo de aplicaciones y que por lo tanto tienen oportunidades similares para convertirse en usuarios potenciales del paradigma RIA; además, que con el conocimiento adquirido mediante la construcción de estas aplicaciones, se cuenta ya con la base respecto de cómo interactúan las RIA al comunicarse con servicios y la administración de información dentro de la aplicación.

Figura 44. **Distribución del uso de otros paradigmas entre sector público y sector privado**



	Aplicaciones Web Asíncronas	Aplicaciones Web Síncronas	CMS	Oracle Forms
Público	19%	12%	0%	3%
Privado	26%	24%	3%	12%

Fuente: elaboración propia. Investigación de campo.

4.3.3. Conclusiones de la investigación de campo

El conocimiento de las Aplicaciones Enriquecidas para Internet en Guatemala se encuentra comprendido por un significativo 74%, lo cual indica que existe oportunidad de la implementación del paradigma por parte de las personas involucradas en el proceso de la creación de *software* dado el alto porcentaje de conocimiento. Existen diferencias en cuanto al conocimiento del paradigma entre el sector público y el sector privado, de modo que para el

sector público existe una diferencia marcada entre el 48% que conoce el paradigma y el 52% que no lo conoce, caso contrario en el sector privado en donde el 82% conoce el paradigma y el 18% no.

La comprensión que se tiene respecto al paradigma RIA tanto en sus beneficios como en las mejoras que ofrece a las aplicaciones *web*, está orientado a los beneficios principales que provee este paradigma, los cuales son la experiencia de uso que ofrece al usuario con un 33% y el rendimiento con un 24%.

Se encontró también que se percibe otro beneficio a nivel más técnico por parte de la muestra seleccionada, el cual es la facilidad de desarrollo que provee el paradigma para la construcción de aplicaciones *web*; este beneficio obtuvo un 28% del total de razones por las cuales se utiliza. Este porcentaje indica que la adopción del paradigma ha traído consigo beneficios a los involucrados en el desarrollo del *software*, lo cual puede representar un móvil para la ampliación de su uso en Guatemala.

Para la construcción de aplicaciones RIA, el uso de una herramienta no debe de ser una restricción o una regla que todos deban de seguir, pero sí se debe de tener en cuenta por qué en un contexto distinto al nacional, las herramientas sean utilizadas en distintas proporciones, esto con el fin de poder evaluar ciertos aspectos tales como la aceptación y compatibilidad que tengan dentro del mercado de navegadores de Internet o la documentación existente, ya que el no analizar estos aspectos, se puede llegar a limitar el alcance de la aplicación que se esté construyendo.

Esto ocurre en cuanto al uso de herramientas en Guatemala, en donde la que posee un mayor porcentaje de preferencia entre la muestra es Microsoft Silverlight con un 31%, mientras que el mercado mundial de uso de aplicaciones enriquecidas para internet indica que son las aplicaciones basadas en Flash (Adoble Flash Builder) las que tienen mayor penetración, con un promedio del 95.74% en el primer cuatrimestre del 2011.

Del 46% de proyectos en los cuales no se utiliza el paradigma RIA, el motivo principal por el cual no se utiliza, que obtuvo mayor porcentaje según las respuestas obtenidas, es debido a que no aplica al negocio, con un 44%; pero también se encontró que entre los paradigmas que se utilizan en lugar del RIA es el de las Aplicaciones *Web* Asíncronas no Enriquecidas, con un 43%. Esto quiere decir que existe una alta probabilidad de que los individuos que respondieron que las RIA no aplican al negocio, estén utilizando el paradigma de las aplicaciones *web* asíncronas no enriquecidas.

Ahora, al analizar cuál es la diferencia entre el paradigma RIA y las aplicaciones *web* asíncronas no enriquecidas, se encuentra que la diferencia es el enriquecimiento que se ofrece en la interfaz de usuario de la aplicación, la cual no está diseñada para utilizar o reproducir multimedia de forma embebida, pero su comportamiento es bastante similar al paradigma RIA.

Conociendo esta información se puede establecer que los usuarios que indican que no utilizan el paradigma por no aplicarse al negocio, pero que sí usan el paradigma de aplicaciones asíncronas no enriquecidas, son usuarios potenciales del paradigma RIA, puesto que únicamente se necesita que las aplicaciones en las que trabajan requieran de una interfaz de usuario que ofrezca facilidades para el uso multimedia, para que estos migren a utilizar este paradigma.

CONCLUSIONES

1. Las Aplicaciones Enriquecidas para Internet son una mejora a las aplicaciones *web* convencionales, que les permite ofrecer una interfaz de usuario enriquecida y un mejor rendimiento. Las Aplicaciones Enriquecidas para Internet conservan las mismas características que las aplicaciones *web* tradicionales, además de permitir heredar responsabilidades al cliente, comunicación asíncrona con el servidor, reproducción multimedia no dependiente del cliente y mejores interfaces de usuario, ya que tienen como objetivo el brindar un ambiente de uso similar a las aplicaciones de escritorio.

2. Las Aplicaciones Enriquecidas para Internet, hacen uso de otras herramientas o utilidades para su ejecución. Entre estas se encuentran el uso de comunicación asíncrona con el servidor, el uso de *rich clients*, el aislamiento de procesos como medida de seguridad, y el uso de *plug-ins,* los cuales permiten la ejecución de herramientas como Adobe Flash o Microsoft Silverlight.

3. Las ventajas que ofrecen las Aplicaciones Enriquecidas para Internet son principalmente mejora en el desempeño de la aplicación minimizando la comunicación cliente-servidor mediante la comunicación asíncrona y la capacidad de brindar interfaces de usuario con capacidades similares a las interfaces de las aplicaciones de escritorio.

4. Las desventajas de las Aplicaciones Enriquecidas para Internet son el tiempo de carga inicial de la aplicación y la falta de estandarización en cuanto al uso de estas aplicaciones.

5. Al comparar las ventajas que provee el paradigma contra sus desventajas, tienen mayor importancia sus ventajas puesto que estas pueden ayudar a resolver problemas que se presentan en determinadas circunstancias y mejoran la presentación de las aplicaciones *web*; mientras que sus desventajas vienen heredadas desde generaciones anteriores de paradigmas de aplicaciones *web* que son causadas por factores externos a la aplicación, dependientes del funcionamiento de las redes de computadoras y del crecimiento no estandarizado de tecnologías para la construcción de la *web*.

6. Gracias al uso que están teniendo las Aplicaciones Enriquecidas para Internet y al valor agregado que dan a las aplicaciones *web*, se encuentran actualmente disponibles en Guatemala diversas herramientas para el desarrollo de éstas; la herramienta predominante es AJAX, la cual permitió por primera vez la comunicación asíncrona entre cliente-servidor; la tecnología Flash, que podría ser utilizada para la construcción de aplicación con interfaces enriquecidas pero que se debía de modificar la herramienta para que fuera mejor aceptada entre los desarrolladores de *software*, creando así Adobe Flex Builder, la herramienta con mayor penetración en el mercado internacional de Aplicaciones Enriquecidas para Internet.

7. Microsoft y Sun lanzaron sus respectivas herramientas para la creación de RIA, con Silverlight y JavaFX, respectivamente, además de la aparición de otras herramientas gratuitas de código fuente abierto como Mono Moonlight y Open Laszlo.

8. Actualmente se está iniciando a utilizar la herramienta HTML5, una mejora al lenguaje HTML para soportar controles que proveen de interfaces enriquecidas, aunque aún se está trabajando en la estandarización de todas sus etiquetas y funcionalidades por parte de los distintos navegadores de Internet.

9. El paradigma más utilizado para los proyectos que no hacen uso del paradigma RIA es bastante similar a él, por lo que la mezcla del conocimiento y de la experiencia en crear aplicaciones similares, crean el contexto perfecto para que al momento que sea requerido por las exigencias de los proyectos, se pueda hacer la migración a este paradigma sin encontrar dificultades.

RECOMENDACIONES

1. El sector público debe de procurar crear aplicaciones *web* basadas en este paradigma, ya que son aplicaciones que son accedidas por una amplia cantidad de usuarios a diario y en muchas ocasiones han sido criticadas por el malestar que ocasiona a los usuarios al utilizarlas debido a los tiempo de respuesta y la experiencia de uso, factores que pueden ser superados en un porcentaje distinto por el paradigma RIA.

2. Se debe de validar las características y documentación disponible de cada herramienta para poder seleccionar una que se adapte de mejor forma al proyecto que se va a desarrollar y a las fortalezas que posee el equipo de trabajo.

3. Debido a que las aplicaciones RIA basadas en Flash son las que mayor uso tienen en el mundo, es recomendable promover su uso y así crear un contexto en el cual empresas como Adobe provean y autoricen centros de capacitación sobre esta tecnología, ya que de esta forma se aumentan las posibilidades para capacitar y certificar a profesionales involucrados en el proceso de elaboración de *software*, aumentando su valor para la empresa y para la calidad del *software* elaborado en Guatemala.

4. Desarrollar aplicaciones RIA utilizando los beneficios que provee HTML5 y CSS3 aprovechando sus nuevas características para proveer mejores soluciones de *software*, estando consientes que algunas nuevas versiones de navegadores para Internet con soporte para HTML5, dejarán de ser compatibles con sistemas operativos anteriores.

5. Elaborar un documento que respalde y motive el cambio de paradigma de Aplicaciones *Web* Asíncronas No Enriquecidas hacia las Aplicaciones RIA. Plasmando en dicho documento los resultados de un análisis completo que refleje las diferencias positivas que posee el paradigma RIA al ser utilizado y al no utilizarse, demostrando también a través de un estudio estadístico, el tiempo que puede ahorrar a los usuarios, enriqueciendo las interfaces de las aplicaciones que lo utilizan.

BIBLIOGRAFÍA

1. AAKER, David; KUMAR V.; DAY, George S. *Investigación de mercados*. México DF: Limusa, 2001. 816 p. ISBN-10: 0471170690.

2. Adobe Labs. *Adobe Dreamweaver CS5 HTML5 Pack* [en línea]. Adobe Systems Incorporated, 2011 [ref. de 04 de mayo de 2011]. Disponible en *Web*: <http://labs.adobe.com/technologies/html5pack/>.

3. ALLAIRE, Jeremy. *Macromedia Flash MX-A next-generation rich client*. California, U.S.A: 2002. Macromedia White Paper. 14 p.

4. ALLEN, Jonathan. The *death and rebirth of Mono* [en línea]. InfoQ.com. mayo 2011 [ref. de 17 de mayo de 2011]. Disponible en *Web*: <http://www.infoq.com/news/2011/05/Mono-II>.

5. Aptana. *Aptana Studio 3* [en línea]. Aptana, Inc. [ref. 04 de mayo de 2011]. Disponible en *Web*: <http://aptana.com/products/studio3>.

6. BBC News. *Adobe buys Macromedia for $3.4bn* [en línea]. Reino Unido: abril 2005 [ref. de 26 de marzo de 2011]. Disponible en *Web*: <http://news.bbc.co.uk/2/hi/business/4456895.stm>.

7. Bright Software Pty. *Rich Clients vs. Thin Clients*. Sydney, Australia: Bright Software White Paper. 2003. 3 p.

8. BUSCH, Marianne; KOCH, Nora. *Rich Internet applications State-of-the-Art*. Múnich, Alemania: Ludwig-Maximilians-Universität, 2009. 18 p.

9. CLINICK, Andrew. *Remote Scripting* [en línea]. Microsoft Corporation. Washington, U.S.A.: abril 1999 [ref. de 16 de marzo de 2011]. Disponible en *Web*: <http://msdn.microsoft.com/en-us/library/ms974566.aspx>.

10. CRANFORD TEAGUE, Jason. *Visual Quickstart Guide CSS3.* Berkeley, California, U.S.A.: Pearson Education, 2011. 412 p.

11. DEB, Brijesh; BANNUR, Sunil G.; BHARTI, Shaurab. *Rich Internet applications (RIA) opportunities and challenges for Enterprises*. Bangalore, India: Infosys Technologies, 2011. 8 p.

12. DUNCAN DAVIDSON, James; COWARD, Dany. *Java Servlet Specification, v2.2*. Palo Alto, California, U.S.A.: Sun Microsystems, 1999. 79 p.

13. Free Software Foundation. *The free Software definition*. [en línea]. Free Software Foundation, Inc. noviembre 2010 [ref. de 12 de abril de 2011]. Disponible en *Web*: <http://www.gnu.org/philosophy/free-sw.html>.

14. HAMMON, Jeffrey S.; RYMER, John R.; KNOLL Adam. *Does HTML5 Herald The End Of RIA Plug-Ins? Not Really.* Cambridge, USA: Forrester Research, 2010. 9 p.

15. HUNT, Lachlan. *A Preview of HTML5* [en línea]. A List Apart, diciembre 2007 [ref. 04 de mayo de 2011]. Disponible en *Web*: <http://www.alistapart.com/articles/previewofhtml5/>.

16. IONESCU, Daniel. *3 Reasons why iPhone won't get Adobe Flash* [en línea]. PCWorld Communications, Inc. octubre 2009 [ref. de 25 de marzo de 2011]. Disponible en *Web*: <http://www.pcworld.com/article/173092/3_reasons_why_iphone_wont_get_adobe_flash.html>.

17. ISKOLD, Alex. *Web 3.0: When Web Sites Become Web Services* [en línea]. ReadWriteWeb, marzo 2007 [ref. de 17 de mayo de 2011]. Disponible en *Web*: <http://www.readwriteweb.com/archives/web_30_when_web_sites_become_web_services.php>.

18. KNORR, Eric; GRUMAN, Galen. *What cloud computing really means* [en línea]. InforWorld, Inc. [ref. de 18 de mayo de 2011]. Disponible en *Web*: <http://www.infoworld.com/d/cloud-computing/what-cloud-computing-really-means-031?>.

19. KOTLER, Philip. *Fundamentos de marketing*. 8a ed. México DF: Prentice Hall, 2008. 744 p. ISBN-10: 9702604001.

20. Laszlo Systems. *OpenLaszlo Architecture* [en línea]. Laszlo Systems, Inc, 2010 [ref. de 02 de mayo de 2011]. Disponible en *Web*: <http://www.openlaszlo.org/lps4.9/docs/developers/architecture.html>

21. LÓPEZ ALTAMIRANO, Alfredo. *Qué son y para qué sirven las investigaciones de mercado*. México DF: CECSA, 2002. 236 p. ISBN-10: 9702401976.

22. MALY, Robin Jan. *Comparison of Centralized (Client-Server) and Decentralized (Peer-to-Peer) Networking*. ETH Zurich, Suiza: Computer Engineering and Networks Laboratory, 2003. 90 p.

23. MCINTOSH, Neil. *Google buys Blogger web service* [en línea]. The Guardian, febrero 2003 [ref. de 16 de mayo de 2011]. Disponible en *Web*: <http://www.guardian.co.uk/business/2003/feb/18/digitalmedia.citynews>.

24. Microsoft Corporation. *Windows Communication Foundation is…* [en línea]. Microsoft Corporation. [ref. de 22 de abril de 2011]. Disponible en *Web*: <http://msdn.microsoft.com/en-us/netframework/aa663324>.

25. Mono**.** *Moonlight – Mono* [en línea]. Novell [ref. 30 de abril de 2011]. Disponible en *Web*: <http://www.mono-project.com/Moonlight>.

26. MonoDevelop. *FAQ Monodevelop* [en línea]. MonoDevelop [ref. de 30 de abril de 2011]. Disponible en *Web*: <http://monodevelop.com/FAQ>.

27. MSDN. *WCF RIA Services* [en línea]. Microsoft Corporation, 2011 [ref. de 22 de abril de 2011]. Disponible en *Web*: <http://msdn.microsoft.com/es-es/library/ee707344(d=printer,v=vs.91).aspx>.

28. _____. *WPF Architecture* [en línea]. Microsoft Corporation, 2011 [ref. de 23 de abril]. Disponible en *Web*: <http://msdn.microsoft.com/en-us/library/ms750441.aspx>.

29. _____. *XAML Overview (WPF)* [en línea]. Microsoft Corporation, 2011 [ref. de 23 de abril de 2011]. Disponible en *Web*: <http://msdn.microsoft.com/en-us/library/ms752059.aspx>.

30. NASCIMBENE, Carlos. *¿Qué son las Rich Internet Applications?* [en línea]. Buenos Aires, Argentina: Canal-AR, diciembre 2005 [ref. de 23 de marzo de 2011]. Disponible en *Web*: <http://www.canal-ar.com.ar/noticias/noticiamuestra.asp?Id=2639>.

31. Oracle Corporation. *JavaFX Moblie – At a Glance* [en línea]. Oracle Corporation, 2010. [ref. de 25 de abril de 2011]. Disponible en *Web*: <http://java.sun.com/javafx/mobile/>.

32. PAUL, Ryan. *Lunar Eclipse: open source Silverlight design tool for Linux* [en línea]. Ars Technica, 2008 [ref. de 30 de abril de 2011]. Disponible en *Web*: <http://arstechnica.com/open-source/news/2007/09/lunar-eclipse-open-source-silverlight-design-tool-for-linux.ars>.

33. STRICKLAND, Jonathan. *How Web 3.0 Will Work* [en línea]. Discovery Communications, LLC. [ref. de 16 de mayo de 2011]. Disponible en *Web*: <http://computer.howstuffworks.com/web-301.htm>.

34. The Eclipse Foundation. *Eclipse Public License – v 1.0* [en línea]. The Eclipse Foundation. [ref. de 20 de abril de 2011]. Disponible en *Web*: <http://www.eclipse.org/org/documents/epl-v10.php>.

35. TILLMAN, Karen. *Oracle Buys Sun* [en línea]. Redwood Shores, California, U.S.A.: abril 2009 [ref. de 27 de marzo de 2011]. Disponible en *Web*: <http://www.oracle.com/us/corporate/press/018363>.

36. Web 3.0 Web Semántica. *Tecnologías actuales en la Web* 3.0 [en línea]. [ref. de 17 de mayo de 2011]. Disponible en *Web*: <http://web30websemantica.comuf.com/tecnologiaweb3-0.htm>.

37. W3School. *Introduction to XHTML* [en línea]. W3School. [ref. de 04 de mayo de 2011]. Disponible en *Web*: <http://www.w3schools.com/xhtml/xhtml_intro.asp>.

38. Wikipedia. *Adobe Integrated Runtime*. [en línea]. Wikipedia, abril 2011 [ref. de 22 de abril de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/Adobe_Integrated_Runtime>.

39. _____. *Extensible Application Markup Language* [en línea]. Wikipedia, abril 2011 [ref. 23 de abril de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/Extensible_Application_Markup_Language>.

40. _____. *Social Media* [en línea]. Wikipedia, mayo 2011 [ref. de 14 de mayo de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/Social_media>.

41. _____. *Fat Client* [en línea]. Wikipedia, febrero 2011 [ref. 26 de marzo de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/Fat_client>.

42. _____. *JavaFX Script* [en línea]. Wikipedia, abril 2011 [ref. de 26 de abril de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/JavaFX_Script>.

43. _____. *MXML* [en línea]. Wikipedia, febrero 2011 [ref. de 17 de abril de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/MXML>.

44. _____. *Plug-in (computing)* [en línea]. Wikipedia, marzo 2011 [ref. de 23 de marzo de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/Plug-in>.

45. _____**.** *Sandbox (computer security)* [en línea]. Wikipedia, febrero 2011 [ref. de 27 de marzo de 2011]. Disponible en *Web*: <http://en.wikipedia.org/wiki/Sandbox_(computer_security)>.

46. WILLSON, Simon. *A re-introduction to JavaScript* [en línea]. Mozilla Developer Network, marzo 2006 [ref. de 04 de mayo de 2011]. Disponible en *Web*: <https://developer.mozilla.org/en/A_re-introduction_to_JavaScript>.

47. WILSON, Tracy V. *How Semantic Web Works* [en línea]. Discovery Communications, LLC. [ref. de 18 de mayo de 2011]. Disponible en *Web*: <http://computer.howstuffworks.com/semantic-web1.htm>.

48. World Wide Web Consortium. *Document Object Model (DOM) Level 3 Events Specification* [en línea]. W3C Working Draft, SHEPERS, Doug, ROSSI, Jacob, 2010 [ref. de 04 de mayo de 2011]. Disponible en *Web*: <http://www.w3.org/TR/DOM-Level-3-Events/>.

49. World Wide Web Consortium. *HTML5* [en línea]. W3C Working Draft, [ref. de 04 de mayo de 2011]. Disponible en *Web*: <http://w3.org/TR/html5/introduction.html>.

50. YAGÜE, Helga. *Un experto en seguridad consigue burlar la seguridad sandbox de Adobe* [en línea]. eWeek Europe, enero 2011 [ref. de 27 de marzo de 2011]. Disponible en *Web*: <http://www.eweekeurope.es/noticias/un-experto-en-seguridad-consigue-burlar-la-seguridad-sandbok-de-adobe-9964>.

51. YANG, Herong. *Web Scripting Architecture Overview* [en línea]. 2008 [ref. de 05 de abri de 2011]. Disponible en *Web*: <http://www.herongyang.com/JavaScript/Web-Scripting-Architecture-Overview.html>.

52. ZAPF, Mariko. The Death of the Web is inevitable, according to Forrester *Research* [en línea]. Massachusette, U.S.A.: Forrester Research, mayo 2001 [ref. de 18 de marzo de 2011]. Disponible en *Web*: <http://www.forrester.com/ER/Press/.

APÉNDICES

APÉNDICE 1: Diseño de la encuesta

ENCUESTA SOBRE LAS APLICACIONES ENRIQUECIDAS PARA INTERNET

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

Objetivo general

Establecer la situación tecnológica que poseen las empresas en la ciudad de Guatemala a nivel de desarrollo *web*.

La encuesta va dirigida a los desarrolladores de *software*.

1) ¿A qué sector pertenece la empresa para la que labora?

___ PÚBLICO
___ PRIVADO

2) ¿Conoce usted las Aplicaciones Enriquecidas de Internet?

___ SÍ
___ NO

3) ¿Utiliza o ha utilizado las Aplicaciones Enriquecidas de Internet en el desarrollo de los proyectos en los que ha trabajado la empresa?

___ SÍ
___ NO

4) Según la proporción de proyectos totales de la empresa, aproximadamente ¿en cuántos de los proyectos ha utilizado las Aplicaciones Enriquecidas de Internet?

___ 75% - 100%
___ 50% - 75%
___ 25% - 50%
___ 0% - 25%

SI UTILIZA LAS APLICACIONES ENRIQUECIDAS DE INTERNET:

5) ¿Por qué razón o razones escoge usted las Aplicaciones Enriquecidas de Internet para trabajar en los proyectos?

_____
_____
_____
_____

6) ¿Qué tecnologías utilizan en la empresa para desarrollar las Aplicaciones Enriquecidas de Internet?

_____
_____
_____
_____

SI NO UTILIZA LAS APLICACIONES ENRIQUECIDAS DE INTERNET:

5) ¿Por qué razón o razones no escoge usted las Aplicaciones Enriquecidas de Internet para trabajar en los proyectos?

_____

_____

_____

_____

6) ¿Qué otro paradigma utilizan en su empresa en vez de las aplicaciones enriquecidas de internet?

_____

_____

_____

_____