Universidad Central De Venezuela

Facultad De Ciencias

Escuela De Computación

Aplicaciones con Tecnología en Internet

DESARROLLO DE UN SISTEMA DE GESTIÓN ESCOLAR USANDO ELEMENTOS DE APLICACIONES WEB ENRIQUECIDAS

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
Por el bachiller
Samuel O. Carvajal M.
C.I. 14.645.065
para optar al título de
Licenciado en Computación

Tutor: Prof. Andrés Castro

Caracas, Abril de 2009.

UNIVERSIDAD CENTRAL DE VENEZUELA

FACULTAD DE CIENCIAS

ESCUELA DE COMPUTACIÓN

TRABAJO ESPECIAL DE GRADO

Titulo: Desarrollo de un Sistema de Gestión Escolar usando elementos de Aplicaciones Web Enriquecidas.

Autor: Samuel Carvajal

Palabras Claves: AJAX, Sistema, Web, Gestión Escolar, CEAPUCV

Tutor: Andrés Castro
Fecha de presentación: 29 de abril de 2009.

Resumen

Para el desarrollo del Sistema de Gestión Escolar del C.E.A.P.U.C.V es indispensable el uso de herramientas que ayuden de manera versátil y eficaz en la construcción de esta aplicación. Adicionalmente, se quiere que esta aplicación tenga una interacción con el usuario de la forma mas fluida posible, para ello se utilizaron elementos de Aplicaciones Web Enriquecidas más específicamente el enfoque AJAX y todas las bondades que este ofrece.

ACTA

Quienes suscriben miembros del Jurado, designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por el bachiller **Samuel O. Carvajal M., C.I. 14.645.065** con el título **"Desarrollo de un Sistema de Gestión Escolar usando elementos de Aplicaciones Web Enriquecidas",** a los fines de optar al título de Licenciado en Computación, dejan constancia lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del Jurado, se fijó el 29 de Abril de 2009 a las 3:00 p.m., para que su autor lo defendiera en forma pública, se hizo en el aula PA III de la Escuela de Computación, mediante una presentación oral de su contenido. Finalizada la defensa pública del Trabajo Especial de Grado, el Jurado decidió aprobarlo con una nota de _____ puntos, en fe de lo cual se levanta la presente Acta, en Caracas a los 29 días del mes de Abril del año dos mil nueve.

Prof. Nora Montaño
Jurado Principal

Prof. Sergio Rivas
Jurado Principal

Prof. Andrés Castro
Tutor

DEDICO UNO

"Gracias te doy mi preferida por darme la vida
por ser mi mejor amiga a la entrada y a la salida
por la total entrega los primeros nueve meses
Cecilia Mujica, Gracias una y mil veces
con nada te comparo, porque a nada te pareces
y mientras menos me pides, mucho mas te mereces
Mi juventud se marchita mientras que tu vejez florece
perdón por el sufrimiento a causa de mis inmadureces

Siempre serás la mía hasta que la muerte nos separe
llevamos el mismo corazón, solo que en diferentes lugares
Agradezco que me ampares, y por dejarme decidir
por entender quien soy, y cual es mi forma de vivir
por existir y hacer de mi un hombre resuelto
y por hacer de padre y madre aunque mi padre no esté muerto

Por eso Dedico Uno
a la mujer que me vio nacer,
llorar, gatear, correr, ganar, perder
a este gran ser que me ha hecho ver
lo estúpida que es la vida del hombre sin la mujer"

Dedico este T.E.G. a mi madre Cecilia Mujica y a la memoria de mi "MAMATIA" Josefa Garnier mis musas y las personas mas influyentes de mi vida.

Samuel Carvajal

Agradezco

A mis mejores amigos y compañeros de todos los semestres Porfi, Hanson, Ler y Nino.

A mis amigos Edwin Maíz e Yrving Tovar por siempre estar alli en las buenas y en las malas incondicionalmente.

A mis 3 tesoros, mi hermana Marta y mis 2 bellas sobrinas Valeria y Ariana, las quiero con todo mi corazón.

A Adelis Nieves por ser la persona que más me motivo y lucho por que cumpliera esta meta y por los 6 años que estuvo a mi lado siempre motivándome a ser una mejor persona, sin ti no lo hubiese logrado.

Especial agradecimiento a los licenciados Carlos Moreno y Harold González por toda la colaboración que me brindaron a la hora de realizar este T.E.G., mis peores mejores amigos.

A mis amigos de Coche Carlin, Daniyer, Jackson, Chipi, Pipo, Gineidy y Day que siempre me han apoyado y han estado muy pendientes de mi carrera universitaria.

A mi queridísima madre Cecilia Mujica por ser mi pilar y fuente de Inspiración.

Índice de Contenidos

Índice de Figuras

Índice de Tablas

Introducción

La importancia de Internet es innegable, especialmente cuando la tendencia es cada vez más hacia las aplicaciones vía Web que agrupan personas con interés en común y donde éstas puedan ser visualizadas de una forma sencilla y que a su vez estas aplicaciones se asemejen mas en su funcionamiento e interacción a las aplicaciones de escritorio, además que Internet permite la comunicación en tiempo real a un número ilimitado de usuarios. Al momento de desarrollar estas aplicaciones, vía Web, se debe buscar la alta calidad, eficiencia y efectividad del sistema, minimizar los tiempos de respuesta y que la interacción con la aplicación sea lo mas fluida posible. Existe un tipo de aplicación Web llamadas Aplicaciones Web Enriquecidas (RIA) que mediante el uso de una serie de tecnologías utilizadas en la programación Web nos brindan todas estas funcionalidades rompiendo con el esquema tradicional de las aplicaciones Web. La selección de las tecnologías y herramientas a utilizar depende de la elección del programador y de las necesidades del usuario.

Cada vez son más las instituciones que se unen a la idea de buscar automatizar sus procesos a través de aplicaciones que puedan ser desarrolladas vía Web, que permitan ser visualizadas de una forma sencilla y que puedan ser acezadas desde casi cualquier computador. Tal es el caso del C.E.A.P.U.C.V (Centro Educativo de la Asociación de los Profesores de la Universidad Central de Venezuela), una institución que requiere de un sistema que permita la creación, aprobación, distribución de planes de evaluación y la entrega de notas de las diferentes actividades que van a ser evaluadas durante el periodo correspondiente. Este sistema debe poder estar al alcance de los diferentes miembros (Oficina de Control de Estudios, Profesores, Alumnos y Representantes) que conforman la comunidad educativa. Este proceso debe ser fácil, sencillo y casi inmediato, y la Internet brinda una excelente alternativa, es por esta razón que se desea que este proceso se realice vía Web.

Por todas las razones mencionadas anteriormente nace el interés de desarrollar una aplicación Web, que permita la gestión de los planes de evaluación y entrega de notas, basado en el esquema de programación AJAX y favorecerlos con las comodidades y beneficios que este enfoque posee. En el presente Trabajo Especial de Grado se tendrá como objetivo la aplicación del esquema de programación AJAX para brindar una interacción mas fluida con la aplicación, basándose también en una arquitectura MVC

usando el lenguaje de programación JAVA y MySQL para el manejo de la base de datos, todo esto con el objeto de desarrollar un sistema de gestión escolar para el C.E.A.P.U.C.V.

Planteamiento del Problema

En este punto se conocerán los objetivos tanto generales como objetivos específicos que se desean alcanzar y el problema que se desea atacar con este Trabajo especial de grado los cuales son los siguientes:

Problema

Actualmente en el C.E.A.P.U.C.V la notificación, creación y aprobación de los planes de evaluación para cada periodo se hacen de forma manual, así como también la entrega de notas por parte del profesor a los alumnos. Debido a esto surgen los siguientes problemas:

- La cantidad de tiempo invertido en la entrega de los planes de evaluación, corrección, aprobación y entrega de notas dentro del C.E.A.P.U.C.V. ya que esto genera retrasos en la preparación de cada año escolar.
- La cantidad de recursos (papel, tinta, etc.) que se utiliza en el intercambio de estas informaciones entre los miembros de la comunidad del C.E.A.P.U.C.V.. ya que esto genera un costo extra a la institución y requeire que la entrega sea presencial.
- Poco control para los representantes de las actividades agendadas a sus representados y el acumulado de notas durante el periodo escolar.

La automatización de estos procesos vía Web permitirá a los representantes tener una verdadera visión de las actividades que realiza el alumno así como de su evolución a lo largo del curso, además de agilizar el proceso de creación y aprobación de los planes de evaluación por parte de la oficina de control de estudios y los profesores de la institución.

Objetivo General

El objetivo de este trabajo es desarrollar un sistema de Gestión escolar basado en el esquema de desarrollo RIA (Aplicaciones de Internet enriquecidas), que permita a los profesores y supervisores del C.E.A.P.U.C.V llevar un control automatizado de la creación, modificación y aprobación de los planes de evaluación, así como un control

de las notas de cada actividad de los planes de evaluación por parte de los profesores, representantes y alumnos.

Objetivos Específicos

El presente trabajo tiene como objetivos específicos los siguientes:

- Determinar los requerimientos solicitados por el usuario.
- Diseñar una aplicación Web que cumpla con los principios de usabilidad.
- Desarrollar una aplicación para el C.E.A.P.U.C.V que permita tener su propio sistema de gestión escolar vía Web.
- Realizar pruebas para verificar el correcto funcionamiento del sistema desarrollado.

Justificación

Debido a la falta de automatización que posee el Centro Educativo de la Asociación de Profesores de la Universidad Central De Venezuela (C.E.A.P.U.C.V) para llevar a cabo la creación , entrega, aprobación y notificación de los planes de evaluación; y el problema que esto genera en cuanto a costos y tiempo por la cantidad de material que se necesita, resulta preciso desarrollar una aplicación basada en la Web que permita desarrollar un sistema de gestión escolar para los profesores, alumnos y representantes y empleados de control de estudio que conforman esta comunidad. De este modo se facilitará las tareas tanto del profesor, estudiantes y representante como el de la institución.

Un sistema basado en la Web permitirá automatizar los procesos de creación de planes de evaluación, entrega y aprobación de los mismos del centro educativo así como entrega de notas por parte de los profesores. Como consecuencia se evidenciará una reducción de costos en materiales así como de tiempo, que se consumía realizando estos procesos de forma manual, garantizando de esta forma un mejor desarrollo y automatización del centro educativo.

Capítulo 1. Marco Teórico

Durante mucho tiempo ha existido una brecha entre la experiencia que brindan al usuario las aplicaciones de escritorio y la experiencia que éste obtiene con las aplicaciones Web tradicionales, en cuanto a la mayor riqueza de interacción y rapidez de respuesta de las primeras; algo que parecía estar fuera del alcance de la Web.

Mientras que las aplicaciones de escritorio ofrecen al usuario una gran riqueza en cuanto a interfaz e interacción, y una respuesta casi instantánea ante sus acciones, las aplicaciones Web tradicionales ofrecen una interacción lenta, donde el usuario normalmente, después de cada acción, debe esperar unos segundos por una respuesta, teniendo así un tiempo ocioso. Esta diferencia, básicamente, se debe a la manera en que trabaja el protocolo que hace posible la Web: el protocolo HTTP.

El esquema de funcionamiento que siguen las aplicaciones Web tradicionales se resume de la siguiente manera: para la mayoría de las acciones del usuario se genera una petición HTTP, la cual viaja al servidor; el servidor, al recibir esta petición, y según los parámetros indicados en la misma, realiza algún procesamiento específico para generar la respuesta correcta, la cual es enviada al cliente y es desplegada e interpretada por el navegador.

En la Figura 1 se puede apreciar, gráficamente, el esquema de interacción que siguen las aplicaciones Web tradicionales.

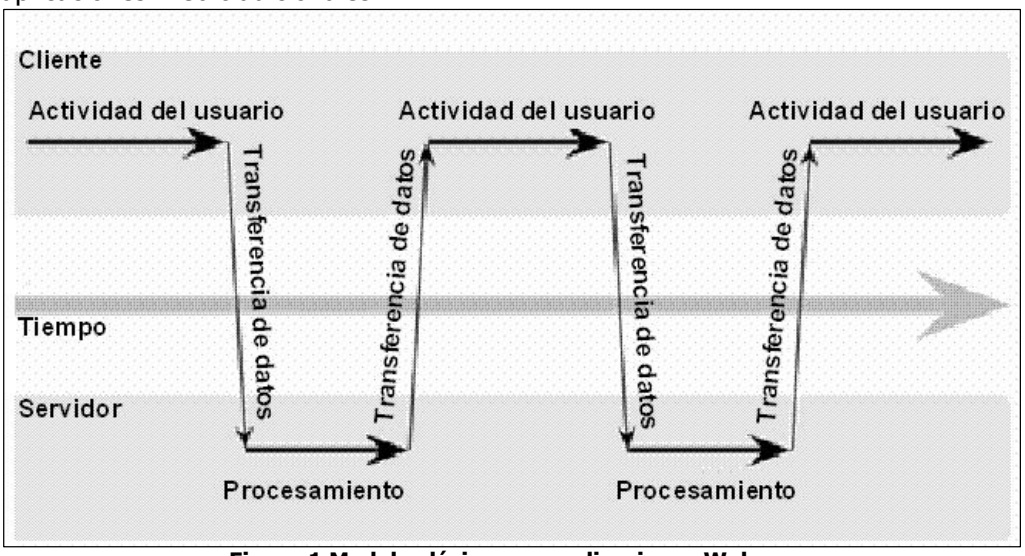


Figura 1 Modelo clásico para aplicaciones Web

Una limitación de este modelo es que mientras la petición viaja hacia el servidor, y éste la procesa, la interacción se detiene; es decir, el usuario se encuentra en espera de los resultados que arrojará su petición, lo cual, entre otros aspectos, produce la brecha que se mencionó anteriormente entre las aplicaciones de escritorio y las aplicaciones Web.

Sin embargo, esta brecha se ha ido cerrando cada vez más gracias a un nuevo esquema de desarrollo, el de las Aplicaciones Web Enriquecidas.

1.1. Aplicaciones de Internet Enriquecidas

RIA (*Rich Internet Application,* o Aplicación de Internet Enriquecida) es un tipo de aplicación Web que provee una mayor riqueza interactiva que las aplicaciones Web tradicionales, incorporando características muy similares a las que poseen las aplicaciones de escritorio, las cuales proveen una interacción dinámica y rica en elementos de interfaz.

El término *Rich Internet Application* fue introducido en marzo de 2002 por Macromedia sin embargo, este mismo concepto ya había sido manejado bajo otros nombres [1]:

- Remote Scripting, por Microsoft, en 1998
- X Internet, por Forrester Research, en octubre de 2000
- Rich Web Clients
- Rich Web Application

Como se mencionó anteriormente, el modelo tradicional de aplicaciones Web tiene una serie de limitaciones, como la poca capacidad multimedia que posee y la recarga continua de páginas. Todo esto se debe a que el cliente en las aplicaciones Web tradicionales sólo se limita a desplegar el contenido HTML. En cambio, las RIAs incorporan un motor como una nueva capa del lado del cliente, que sirve como intermediaria entre la interacción del cliente con el servidor, y tiene la responsabilidad de realizar los cambios sobre la interfaz de usuario.

Algunos de los beneficios que provee el uso de este entorno son los siguientes:

- Se aprovecha la capacidad de procesamiento del lado del cliente. El motor del lado del cliente puede tener la capacidad de realizar actividades solicitadas por el usuario sin la necesidad de realizar una petición al servidor. Estas

actividades podrían ser cálculos matemáticos, cambios en la interfaz de usuario (reposicionar objetos), etc.

- El motor del lado del cliente puede interactuar con el servidor asincrónicamente; por ejemplo, el motor podría solicitar datos al servidor para futuras peticiones del usuario, mejorando así el tiempo de respuesta para dichas peticiones.
- Las RIAs no necesitan hacer recargas continuas de toda la interfaz de usuario; en vista que el cliente es más sofisticado, es posible recargar la sección específica de la página que presente algún cambio.

A pesar de los múltiples beneficios que provee este entorno, las aplicaciones RIA presentan algunas limitaciones:

- Debido a que las RIAs se ejecutan en un sandbox[1], éstas tienen acceso restringido a los recursos del sistema. Una aplicación podría necesitar algún tipo de recurso particular, por ejemplo, tener acceso al sistema de archivos del cliente; en ciertas RIAs este acceso puede estar restringido por el sandbox, por lo tanto, su utilización para el desarrollo de esta aplicación no sería viable ya que no se podría satisfacer dicho requerimiento.
- Las RIAs son dependientes de un componente tecnológico para su correcto funcionamiento, el cual en muchos casos es el soporte de JavaScript o el uso de algún plugin específico. En algunos casos también son dependientes de la plataforma tecnológica (por ejemplo, el navegador).
- Según la compañía IBM [2], existe una serie de aspectos que son necesarios considerar a la hora de escoger una tecnología o un enfoque para desarrollar RIAs. A continuación se describe cada uno de estos aspectos:

Funcionamiento

Los usuarios esperan que el navegador continúe trabajando de la misma forma como si se estuviera interactuando con una aplicación Web tradicional. Esto quiere decir que el usuario espera que los botones atrás e historial del navegador mantengan el comportamiento que presentan en las aplicaciones Web tradicionales, así como también que las teclas de acceso rápido (cortar y pegar, buscar contenido en la página, entre otros) se puedan utilizar en la misma.

[1] Sandbox es un entorno seguro donde las aplicaciones tienen acceso restringido a determinados recursos del sistema.

Reducción de la recarga continua de páginas

Algunas solicitudes por parte del usuario podrían generar cambios en una pequeña parte de la interfaz de usuario; por consiguiente, la tecnología RIA debe facilitar la disminución del número de recargas de la página, permitiendo restringir los cambios únicamente a los sectores que deban ser modificados en la misma.

Elementos de interfaz de usuario disponibles

De debe tratar de utilizar los componentes de interfaz de usuario (menús, formularios, entre otros) para darle más comodidad al usuario, así como tratara de manejar eventos del lado del cliente e incorporar elementos que ofrezcan una mejor forma de interacción con el usuario.

Complejidad e interoperabilidad

La tecnología para desarrollar la RIA debe ser fácil de aprender y de usar. También debe tener la capacidad de ínter operar con tecnologías Web existentes.

Seguridad

Que provea características o mecanismos de seguridad que permitan mantener cierto control sobre la aplicación. Por ejemplo, sería conveniente estar al tanto de todas las conexiones que se establecen por medio del cliente, para así conocer la forma en la cual interactúa el motor del lado del cliente con el servidor y evitar así conexiones no generadas por acciones por parte del usuario de la aplicación (intrusiones).

Soporte para los paradigmas básicos de la Web

La tecnología debe soportar los paradigmas básicos existentes en la Web, tales como internacionalización, independencia del dispositivo de acceso, independencia del navegador, entre otros.

Utilidades

Verificar qué utilidades se encuentran disponibles para el desarrollo de RIAs. Estas utilidades pueden presentarse como *plugins* para los IDE de desarrollo, pudiéndose mencionar depuradores, herramientas de asistencia para la codificación, entre otras.

1.1.1. Tecnologías Utilizadas por las RIAs

Hasta el momento se han mencionado los beneficios y limitaciones que presenta el entorno RIA, así como los aspectos que deben tomarse en cuenta al momento de

escoger una tecnología o un enfoque para desarrollar RIAs. A continuación se mencionarán y describirán algunas de estas tecnologías, con excepción del enfoque AJAX, el cual será descrito con detalle en el capítulo 2 de este documento.

Laszlo (*www.openlaszlo.org/*)

Laszlo es una plataforma de código abierto para el desarrollo de RIAs basada en Flash y XML. Esta tecnología utiliza como motor del lado del cliente el *FlashPlayer* 6.x o una versión superior (éste es incorporado al navegador en forma de *plugin)*; como lenguaje de scripting utiliza LZX el cual es un lenguaje orientado a objetos basado en etiquetas que utiliza la sintaxis de JavaScript y XML para generar archivos Flash de forma dinámica.

El intercambio de datos entre el motor del cliente y el servidor es a través de XML. Los datos transmitidos son parseados en el servidor mediante funciones XPath que es uno de los lenguajes existentes para acceder a los elementos de un archivo XML; posteriormente, el archivo LZX es compilado en el servidor para generar un archivo Flash, que será enviado al cliente para realizar los cambios necesarios en la Interfaz de Usuario.

Mozilla XUL (*www.mozilla.org/projects/xul/*)

XUL (*XML-based User-interface Language*, o lenguaje basado en XML para la Interfaz de Usuario) es un lenguaje de marcado que utiliza la sintaxis de XML para definir componentes de Interfaz de Usuario en los navegadores Netscape y Mozilla. La diferencia entre HTML y XUL es que XUL tiene un conjunto extenso de componentes gráficos como menús, barras de herramientas, cajas de texto, entre otros; estos componentes gráficos son creados sin la necesidad de desarrollar código JavaScript. El motor del lado del cliente puede ser desarrollado utilizando JavaScript para realizar los cambios en la Interfaz de Usuario haciendo uso del DOM.

El intercambio de datos entre el motor del cliente y el servidor se realiza mediante XML.

XForms (*www.w3.org/TR/xforms/*)

Es un nuevo lenguaje de marcado para formularios Web, diseñado para ser el sustituto de los formularios tradicionales HTML. Este lenguaje permite a los desarrolladores de

formularios Web distinguir entre el propósito del formulario y su presentación, lo que ofrece una serie de ventajas en términos de:

- Reutilización: Los módulos XForms pueden reutilizarse independientemente de los datos que recogen.
- Independencia de Dispositivo: Gracias a que los controles de la interfaz de usuario son abstractos y sólo se indican sus características genéricas, es posible el despliegue de estos formularios en diferentes dispositivos.

XForms provee todas las funcionalidades de los formularios HTML y además permite:

- Comprobar automáticamente los datos mientras el usuario los introduce.
- Indicar que ciertos campos son obligatorios y que el formulario no podrá ser enviado sin esta información.
- Enviar formularios de datos como XML, ya que XForms esta basado en XML.
- Enviar el mismo formulario a diferentes servidores (por ejemplo, la búsqueda de una palabra se envía a diferentes motores de búsqueda).
- Guardar y restaurar valores en y desde un archivo (por ejemplo un archivo XML).
- Obtener los datos iniciales para un formulario a partir de un archivo externo.
- Forzar valores para que cumplan determinado criterio; por ejemplo, que los valores estén comprendidos en un rango determinado.
- Combinar tecnologías XML existentes (XML Events, XPath, entre otras).
- Facilitar la creación de formularios complejos.

Como motor del lado del cliente XForms necesita la instalación del *plugin* FormsPlayer. Usando XForms también es posible incorporar el uso de JavaScript y DOM para lograr que las aplicaciones establezcan comunicaciones con el servidor de manera asíncrona.

Adobe Flex (*www.adobe.com/es/products/flex/)*

Adobe Flex es una tecnología basada en Flash para la creación de páginas Web animadas e interactivas. De igual forma que Laszlo, los archivos Flash son generados en el servidor y enviados posteriormente al cliente para que sean mostrados. Adobe

Flex utiliza como motor del lado del cliente el reproductor Flash 6.x o una versión superior (éste es incorporado al navegador en forma de *plugin*).

Los componentes de la Interfaz de Usaurio son definidos en un lenguaje basado en sintaxis XML llamado MXML, que provee un conjunto amplio de librerías para definir componentes visuales.

Un lenguaje de scripting llamado *ActionScript 2* es embebido en MXML para manejar eventos de usuario o eventos del sistema. Éste es un lenguaje orientado a objetos, similar a JavaScript. Al igual que XForms, Flex también puede separar presentación, modelo de datos y servicios de datos (similar al patron Modelo-Vista-Controlador).

Con Flex, todas las peticiones son enviadas en forma de XML al servidor; estas peticiones son resueltas por el compilador de Flex y generan un archivo SWF, el cual será enviado al cliente.

AJAX

Es uno de los esquemas de creación de aplicaciones web enriquecidas mas utilizados en la actualidad. En el siguiente punto se hará un estudio más detallado para conocer en que consiste, su funcionamiento, ventajas, desventajas y características que presenta el mismo.

1.2. AJAX

AJAX (acrónimo para *Asynchronous JavaScript And XML.* JavaScript y XML Asíncronos) es un enfoque de desarrollo basado en un conjunto de tecnologías ya existentes, agrupadas para presentar información e interactuar dinámicamente, de manera asíncrona, con un servidor Web [3].

Entre las tecnologías que agrupa AJAX se destacan las siguientes como las principales:

- HTML y CSS: para la presentación, estructuración y formato del contenido.
- DOM (Document Object Model): Con el modelo de objetos del documento se logra obtener la estructura del documento HTML. Utilizando esta estructura se pueden agregar, eliminar y modificar, de manera dinámica, elementos de la página mediante el uso de la tecnología JavaScript.
- XML: Para el intercambio de datos entre el cliente (navegador Web) y el servidor.

- JavaScript: Mediante esta tecnología del lado del cliente se realizan las peticiones de manera asíncrona y, junto con el manejo del DOM, se logra la interacción dinámica con el usuario.

Con AJAX se busca entonces proveer de una mayor riqueza de interacción entre el usuario y la aplicación en comparación con la que proveen las aplicaciones Web tradicionales. Este enfoque cambia el esquema tradicional de interacción con las aplicaciones Web, procurando una mayor velocidad de respuesta hacia el usuario, permitiéndole mantener la interacción con la aplicación inclusive durante tiempos de procesamiento, con elementos de interfaz más dinámicos e interactivos.

El enfoque AJAX permite el envío de peticiones al servidor en segundo plano, es decir, sin interrumpir la interacción entre el usuario y la aplicación. Esto disminuye el típico tiempo de espera entre peticiones y permite mantener la continuidad en la interacción entre el usuario y la aplicación, lo que le permite al usuario realizar acciones como la modificación de campos de un formulario, despliegue de menús, visualización de datos, entre otros.

Luego de enviar una petición y obtener la información requerida (respuesta) del servidor, las aplicaciones AJAX tienen la capacidad de modificar la vista (estructura de la página) dinámicamente sin necesidad de solicitar una página distinta. Esta información obtenida del servidor está constituida típicamente de datos en formato XML o texto plano, no es necesario recibir en cada petición un documento HTML entero, ya que la aplicación AJAX utiliza estos datos para modificar la página sin necesidad de cambiarla por una nueva.

A continuación se profundiza en el contraste entre las aplicaciones AJAX y las aplicaciones Web tradicionales.

1.2.1. Contraste entre esquema clásico de aplicaciones Web y AJAX

Las aplicaciones Web tradicionales basan su esquema de interacción en el funcionamiento intrínseco del protocolo HTTP; el protocolo HTTP está basado en el paradigma petición-respuesta, donde participan dos actores principales, el cliente y el servidor [3].

El cliente envía peticiones de recursos al servidor, descartando la página desplegada actualmente. El servidor recibe la petición y la procesa, respondiendo seguidamente con una página HTML nueva, la cual es recibida y desplegada por el navegador. Este

proceso genera una interrupción en la interacción entre el usuario y la aplicación en cada petición.

A diferencia de las aplicaciones Web clásicas, las aplicaciones AJAX pueden enviar peticiones al servidor sin interrumpir la interacción, manteniendo la página actual en el navegador de tal manera que le permita al usuario seguir interactuando con la aplicación. Esto es posible gracias al uso de peticiones en segundo plano (peticiones asíncronas).

En el esquema clásico de aplicaciones Web, las páginas de respuesta enviadas por el servidor posiblemente contienen pocos cambios respecto a la página anterior, lo que puede producir una sobrecarga innecesaria entre cada petición. Con el uso de peticiones asíncronas, las aplicaciones AJAX pueden solicitar al servidor Web únicamente la información que represente un cambio en la página, sin solicitar datos que ya se encuentren en el cliente (como imágenes, encabezados, menús, etc.). Está característica permite disminuir considerablemente el impacto sobre la interacción del usuario con la aplicación que comúnmente se presenta en las aplicaciones Web tradicionales al momento de enviar peticiones al servidor.

Al obtener como respuesta del servidor los datos que representen un cambio en la aplicación, el cliente puede modificar la vista (estructura de la página) dinámicamente con la información obtenida, sin necesidad de solicitar una página distinta. Gracias a esta característica, AJAX permite un uso más eficiente del ancho de banda, ya que sólo se transmite del servidor al cliente la información necesaria y no páginas completas con información que ya reside en el cliente, como encabezados y pie de páginas, gráficos e imágenes, etc.

Las aplicaciones AJAX incorporan un componente adicional del lado del cliente denominado motor AJAX. El motor AJAX es un componente constituido principalmente por código JavaScript el cual se encarga de todo el procesamiento del lado del cliente y sirve como intermediario entre la interfaz de la aplicación y el servidor como se muestra en la Figura 2.

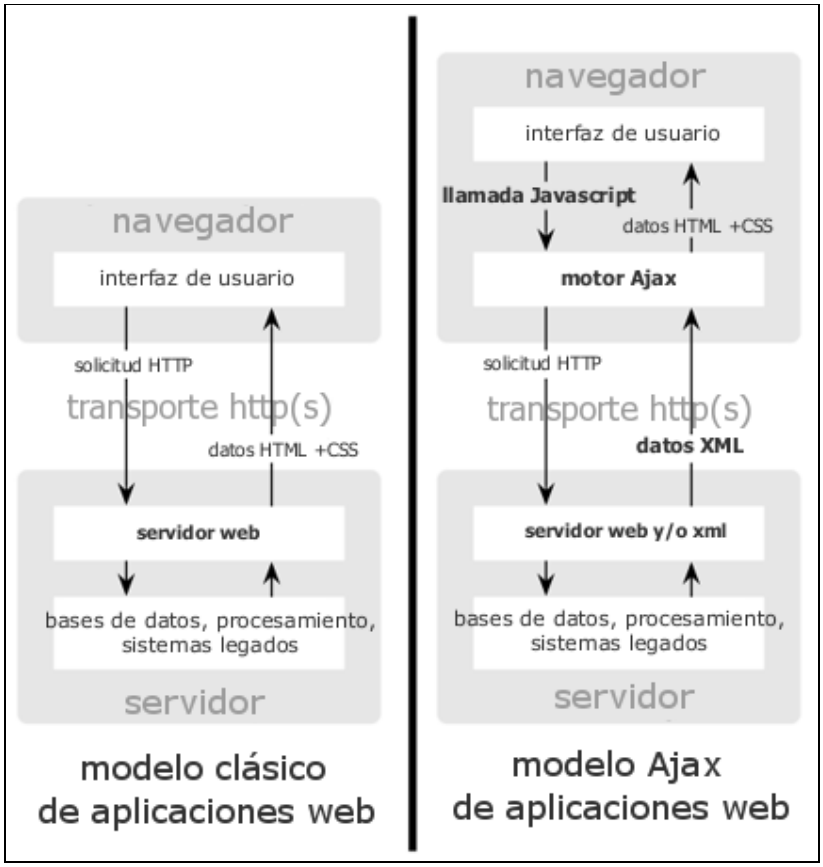


Figura 2 Comparación: Modelo clásico vs. Modelo AJAX

La Figura 2 muestra cómo en el modelo clásico las peticiones al servidor se originan directamente desde la interfaz como consecuencia de las acciones del usuario, en cambio, en el modelo AJAX las acciones del usuario sobre la interfaz son interceptadas por el motor AJAX el cual tiene la posibilidad de darles respuesta directamente o de generar peticiones al servidor (normalmente) en segundo plano.

En el modelo clásico de aplicaciones Web el servidor responde directamente con datos en formato HTML y CSS los cuales son desplegados directamente en el cliente. En el modelo AJAX el servidor responde comúnmente con datos en formato XML o texto plano, los cuales son obtenidos por el motor AJAX y son utilizados por éste para realizar los cambios dinámicos sobre la interfaz.

Luego de comparar el esquema de trabajo de las aplicaciones Web tradicionales y las aplicaciones AJAX, en la siguiente sección se profundiza en el funcionamiento del enfoque AJAX describiendo la base de todas las bondades que brinda y las diferencias con respecto al enfoque clásico de aplicaciones Web.

1.2.2. Funcionamiento

El funcionamiento de la Web sobre el protocolo HTTP explica las razones por las cuales las aplicaciones Web tradicionales presentan interrupciones por cada petición al servidor que se realice y porque éstas presentan una menor riqueza de interacción comparadas con las aplicaciones de escritorio. A continuación se describe cómo funcionan las distintas tecnologías que conforman el enfoque AJAX para incrementar la riqueza de interacción de las aplicaciones Web y obtener un mayor acercamiento al esquema de interacción de las aplicaciones de escritorio. Para esto es importante conocer el papel que juegan las tecnologías que conforman este enfoque [3].

Tecnologías

En la sección se identificaron las tecnologías que componen el enfoque AJAX, sin embargo el uso de cada una por su lado no construye una aplicación AJAX, cada tecnología cumple un papel importante y debe funcionar en conjunto con las otras tecnologías.

- Las Hojas de Estilo en Cascada (CSS) proveen un repositorio de atributos visuales predefinidos que pueden ser asignados a ciertos elementos de la página en cualquier momento.
- El Modelo de Objetos del Documento (DOM) provee una estructura de árbol que representa al documento, donde las distintas etiquetas HTML y los cuerpos de las mismas son los nodos del árbol, que a su vez pueden tener nodos hijos según como se encuentre estructurado el documento. El DOM expone la estructura del documento al motor JavaScript, el cual puede modificar sus elementos dinámicamente según se requiera. Esto se hace utilizando la variable global *document* la cual representa la raíz del árbol.
- JavaScript es el lenguaje utilizado para integrar todas las tecnologías, definiendo la lógica de la presentación (del lado del cliente) y el flujo de trabajo de la aplicación. JavaScript permite:
    - Manipular los elementos del DOM para modificar la interfaz.
    - Obtener la respuesta del servidor y usarla para transformar la vista.
    - Usar las hojas de estilos en cascada para aplicar, dinámicamente, diferente estilos a los elementos de la página.

Hasta ahora se han descrito las tecnologías que conforman el enfoque AJAX; existe un elemento que es el que hace posible la comunicación con el servidor de manera asíncrona, el `XMLHttpRequest`.

1.2.2.1. El objeto XMLHTTPRequest

El objeto `XMLHttpRequest` es el objeto que permite realizar las peticiones en segundo plano, para luego recibir las respuestas del servidor sin alterar el curso de la interacción entre el usuario y la aplicación.

En la Figura 3 se muestra como estas peticiones asíncronas en el enfoque AJAX modifican el esquema tradicional petición-respuesta de las aplicaciones Web mostrado en la Figura 1.

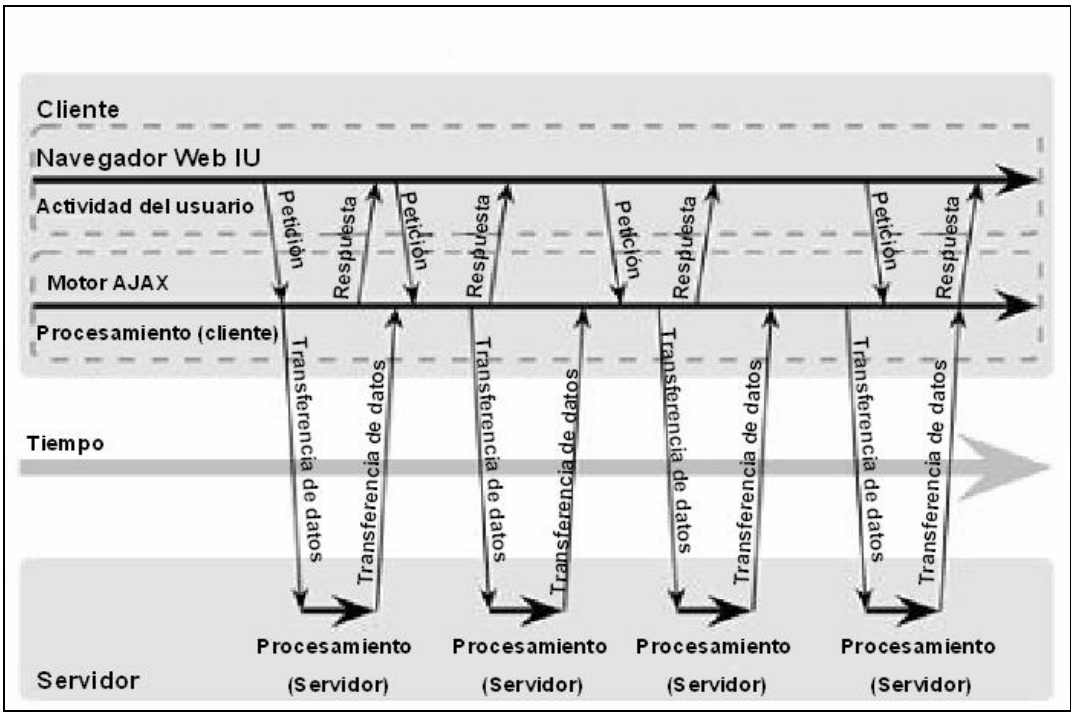


Figura 3 Modelo asíncrono de aplicaciones AJAX

El motor AJAX, creado utilizando el lenguaje JavaScript, es el encargado de enviar las peticiones asíncronas al servidor y a la vez permitir la interacción entre la aplicación y el usuario, todo esto mediante el uso de funciones JavaScript, manejo del DOM y Hojas de Estilo en Cascada. Al recibir la respuesta del servidor, el motor AJAX obtiene la data del objeto XMLHttpRequest y realiza los cambios correspondientes en la página.

Para enviar la petición asíncrona y obtener la data, el motor AJAX hace uso de un conjunto de métodos y atributos entre los cuales se encuentran los siguientes:

open(<tipo de petición>, <url> [, <asíncrono>, <nombre de usuario>, <clave>]): Este método inicializa una nueva petición al servidor, recibiendo como parámetros:

- **<tipo de petición>,** que indica el tipo de petición HTTP (GET, POST, HEAD, etc.).
- **<url>**, que indica la localización del recurso, pudiendo incluir los parámetros de la petición si ésta es de tipo GET.
- **<asíncrono>** (opcional), que puede tener los valores de *true*, si se realiza la petición de manera asíncrona, o *false*, si se quiere que sea síncrona. Este parámetro es opcional y su valor por defecto es *true*.
- **<nombre de usuario>** (opcional), que indica el nombre del usuario a autenticar.
- **<clave>** (opcional), que indica la clave de acceso de dicho usuario.

send(): Este método envía la petición al servidor luego de haber configurado la petición con el método open().

abort(): Este método aborta la petición actual.

Entre los atributos se destacan:

onreadystatechange: Este atributo es de suma importancia, pues indica la función JavaScript que será ejecutada cada vez que cambie el estado de la petición. Esta debe ser indicada antes de enviar la petición.

readyState: Este atributo del objeto provee el estado actual de la petición HTTP.

responseText: Mediante este atributo se obtiene el texto de respuesta del servidor al concluir el proceso de petición-respuesta.

responseXML: Este atributo mantendrá la respuesta del servidor como un objeto XMLDocument, que se puede recorrer usando las funciones DOM de JavaScript.

status: Este atributo provee el estado HTTP de la respuesta obtenida del servidor.

A continuación se describe cómo este objeto trabaja en conjunto con el resto de las tecnologías, desde que se realiza una petición hasta que se recibe la respuesta y se realizan las modificaciones sobre la aplicación.

1.2.2.2. Uso del objeto XMLHttpRequest

Para hacer el proceso de petición-respuesta asíncrono mediante el uso del objeto `XMLHttpRequest` es necesario seguir un conjunto de pasos utilizando los métodos y atributos que provee este objeto, los cuales se describen a continuación.

Creación del objeto XMLHttpRequest

Cuando se desarrolla una aplicación Web, es deseable que ésta funcione en el mayor número de navegadores posibles, sobre todo en los más populares. La creación de una instancia de `XMLHttpRequest` no es independiente del navegador, sino que ésta varía en cada uno de ellos. Por ejemplo, en Internet Explorer la forma de crear un objeto de este tipo varía incluso entre sus distintas versiones, a pesar de ser el mismo navegador. Para la mayoría del resto de los navegadores (Mozilla, Firefox, Opera, Safari, etcétera) se crea de la misma forma; sin embargo, las versiones más antiguas podrían no soportarlo.

Tomando en cuenta estas diferencias, si se desea hacer una aplicación robusta que funcione correctamente en distintos navegadores, es necesario tomar en cuenta todos los casos posibles. En el siguiente fragmento de código se muestra cómo realizar la declaración del `XMLHttpRequest` contemplando distintos navegadores.

```
<script language="javascript" type="text/javascript">
      var request = false;
      try {
       //para la mayoría de los navegadores
       request = new XMLHttpRequest();
      }catch (trymicrosoft) {
       try{
       //IE ultimas versiones
       request = new ActiveXObject("Msxml2.XMLHTTP");
       }catch (othermicrosoft) {
       try{
       //IE versiones anteriores
       request = new ActiveXObject("Microsoft.XMLHTTP");
       }catch (failed) {
       request = false;
       }
       }
      }
      if (!request)
       alert("Error inicializando el XMLHttpRequest!");
</script>
```

Como se observa en el código, se tienen tres formas de declarar un objeto `XMLHttpRequest`:

- Para la mayoría de los navegadores actuales excluyendo Internet Explorer se realiza de la siguiente forma:

```
var request = new XMLHttpRequest();
```

- Para las versiones más modernas del navegador Internet Explorer la declaración se realiza a través de controles Activex de la siguiente forma:

```
var request = new ActiveXObject("Msxml2.XMLHTTP");
```

- Para las versiones menos modernas de Internet Explorer que soportan el `XMLHttpRequest`, la declaración se realiza de la siguiente forma utilizando también controles Activex:

```
var request = new ActiveXObject("Microsoft.XMLHTTP");
```

Envío de la petición

Luego de crear la instancia del objeto `XMLHttpRequest`, es necesario abrir la conexión utilizando el método `open()`, mencionado anteriormente. Cabe destacar que,

por razones de seguridad, las peticiones sólo pueden ser realizadas hacia el mismo dominio en el que la aplicación se está ejecutando; por lo tanto, si se desea realizar una petición externa, no puede hacerse a través del objeto `XMLHttpRequest`.

Antes de enviar la petición, y luego de haber usado el método `open()`, es necesario indicar quién será el encargado de manejar la respuesta, es decir, qué función será invocada cuando se realice un cambio de estado en la petición. Para esto, se le asigna el nombre de la función al atributo `onreadystatechange`; seguidamente, se utiliza el método `send()` para enviar la petición.

El siguiente código muestra un ejemplo de cómo realizar una petición siguiendo los pasos mencionados.

```
request.open('GET', '/un/url/interno?parametro1=valor1');
request.onreadystatechange = funcionCallback;
request.send(null);
```

Manejo de la respuesta

Como se indicó en el paso anterior para manejar la respuesta del servidor se utiliza una función JavaScript cuyo nombre ha sido asignado al atributo `onreadystatechange` como se mencionó en el paso anterior. Cuando la petición cambia de estado, automáticamente se realiza una llamada a esta función, la cual debe manejar el procesamiento que se desee realizar según el estado de la petición.

Según el fragmento de código anterior, la función *`callback`*[2] designada para manejar la respuesta se llama `funcionCallback()`. Dentro de esta función se debe incluir todo el código correspondiente al manejo de la respuesta.

El estado de la petición se verifica en la propiedad `readyState` del objeto `XMLHttpRequest` pudiendo ser alguno de los siguientes:

- **0:** La petición no ha sido inicializada, es decir, no se ha invocado el método `open()`.
- **1:** La petición ya ha sido inicializada, pero no ha sido enviada.
- **2:** La petición ha sido enviada y está siendo procesada. En este estado es posible obtener cabeceras contenidas en la respuesta (*response*).

[2] Función que maneja la respuesta de la petición asíncrona

- **3:** La petición se está procesando y se pueden obtener algunos datos, pero la respuesta no está completa.
- **4:** La petición ha finalizada y se ha obtenido la respuesta completa; es hora de trabajar con ella.

Como se puede observar, el estado de mayor importancia es el 4, ya que este indica el momento en que se debe trabajar con la respuesta que se obtuvo del servidor; sin embargo, en ciertos casos es de utilidad trabajar con el resto de los estados.

Hay que tomar en cuenta que no todos los navegadores trabajan de igual forma con los estados; de hecho, no todos los navegadores reportan todos estos estados mencionados sino que reportan algunos de ellos. Por ejemplo, Internet Explorer no reporta el estado 2, mientras que *Mozilla Firefox* reporta del 1 al 4. El siguiente fragmento muestra cómo puede ser una función *callback*.

```
function funcionCallback() {
 if (request.readyState == 4) { //verificar estado de la petición
 var response = request.responseText;
 ...
 //trabajo con la respuesta
 ...
 }
 }
```

En el extracto de código mostrado anteriormente se puede observar que la función *callback* realiza el manejo de la respuesta del servidor en el momento en que el estado de la petición sea 4. Sin embargo, para manejar la respuesta no basta con conocer el estado de la petición, sino que es de suma importancia verificar también cuál es el estado de la respuesta en sí, obteniendo el código de respuesta HTTP. Este código indica el resultado de la petición, y se obtiene por medio del atributo status del objeto `XMLHttpRequest`.

Entre los códigos de mayor importancia tenemos:

- **401 y 403:** Datos de uso restringido.
- **404:** Recurso no encontrado.
- **200:** Petición completada exitosamente.

En el siguiente fragmento de código se toma en cuenta el estado de la respuesta antes de realizar alguna operación.

```
function metodoCallback() {
 if (request.readyState == 4) { //verificar estado de la petición
 if (request.status == 200) {//verificar estado de la respuesta
 var response = request.responseText;
 ...
      //trabajo con la respuesta
      ...
 } else
 alert("El estado es: " + request.status);
 }
 }
```

Luego de verificar que la petición esta lista (`readyState==4`) y que se haya procesado sin ningún problema, obteniendo los recursos solicitados (`status==200`), dentro de la función *callback* se obtiene la respuesta por medio del atributo `responseText` (en caso que la respuesta sea texto plano sin formato) del objeto `XMLHttpRequest` o del atributo `responseXML` (en caso que la respuesta sea un XML).

Hasta este punto, luego de obtener la respuesta del servidor, llega el trabajo del objeto `XMLHttpRequest` para una petición, dándole paso al resto de las tecnologías involucradas como CSS y DOM, las cuales son manejadas e integradas con el uso de código JavaScript. Utilizando los datos obtenidos en el cliente, para lograr la interacción dinámica con el usuario se hace uso de código JavaScript para manipular el DOM y realizar cualquier cambio en los elementos de la página de una manera rápida, brindándole al usuario un cambio en la vista sin que se produzca perdida de interacción.

1.2.3.Manejo del DOM

El modelo de objetos del documento DOM es una estructura que reside en memoria, la cual representa el contenido de un documento HTML o XML en forma de árbol. En la Figura 4 se observa la representación en estructura de árbol para el siguiente extracto de código HTML:

```
<html>
 <head>
 <title>Ejemplo DOM</title>
 </head>
 <body>
 <h1>Ejemplo DOM</h1>
 <p>Esta pagina muestra un <em>ejemplo</em> de la estructura de arbol DOM.
 </p>
 <div id="ejemplo">Esto es un texto de ejemplo dentro de una etiqueta DIV
 </div>
 </body>
</html>
```

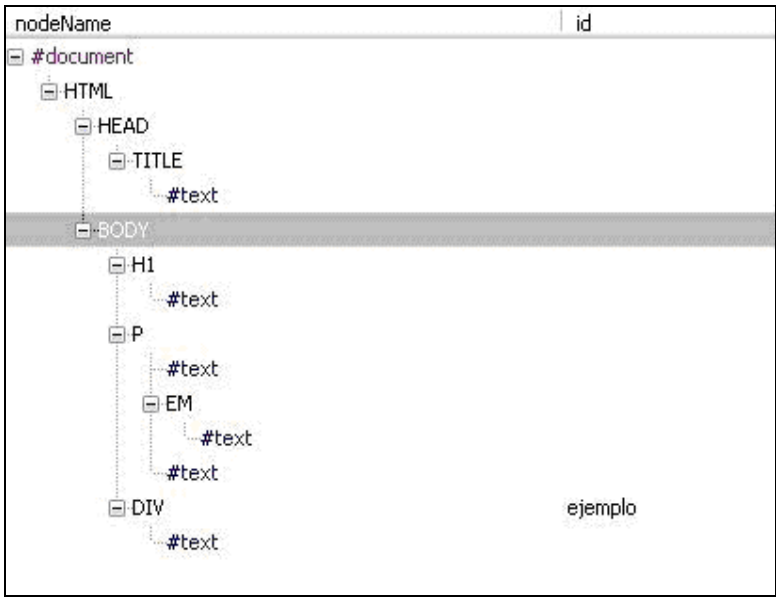


Figura 4 Estructura de árbol DOM

Por medio de esta estructura de árbol es posible realizar cambios en la Interfaz de Usuario sin la necesidad de generar una recarga completa de la página Web; la forma de realizar estos cambios en la Interfaz de Usuario es mediante la API que provee JavaScript para el manejo del DOM. A continuación se describen los métodos más importantes según [4] que provee JavaScript para el manejo del DOM.

Después de obtener la respuesta del servidor en el objeto `XMLHttpRequest`, la primera acción que se debe realizar es crear una variable y asignarle la estructura de árbol del DOM de la página o de la respuesta obtenida.

- Obtener el árbol DOM de la página (HTML):

  ```
  var arbolDOM = document;
  ```

- Obtener el árbol DOM de la respuesta (XML):

  ```
  var arbolDOM = XMLHttpRequest.responseXML.documentElement;
  ```

De esta manera, ya es posible recorrer los elementos del árbol y manipularlos para lograr la apariencia que se desee (agregar, eliminar o modificar elementos).

Los elementos del árbol se pueden obtener de dos formas: mediante el uso del método getElementById o mediante el uso del método getElementsByTagName. A continuación se describen estos dos métodos en las Tablas 1 y 2 respectivamente:

Tabla 1 método getElementById

Método **getElementById**	
Sintaxis	`var elemento = arbolDOM.getElementById(idElemento);`
Parámetros	- **idElemento**: String que contiene el id del elemento. Este parámetro es requerido.
Retorno	Retorna una referencia al primer elemento encontrado que se corresponda con el id especificado por el parámetro idElemento.
Observaciones	Observaciones: En caso de que exista más de un elemento con el mismo id solo será retornado el primero que se encuentre (este método no retorna una colección de elementos).

Tabla 2 método getElementsByTagName

Método **getElementsByTagName**	
Sintaxis	`var elemento = arbolDOM.getElementByTagName(etiqueta);`
Parámetros	- **etiqueta**: String que contiene el nombre de la etiqueta que se desea encontrar.
Retorno	Retorna una colección con los elementos encontrados que se corresponda con el nombre de etiqueta especificada por el parámetro etiqueta
Observaciones	El nombre de etiqueta que se especifica en el parámetro etiqueta no es sensible a mayúsculas y minúsculas. En caso de que el valor del parámetro etiqueta sea "*" se devuelve toda la estructura de árbol del documento.

Una vez obtenido el elemento deseado, para obtener su valor se hace lo siguiente:

```
var valor = elemento.value;
```

Para hacer cambiar el valor del elemento:

```
elemento.value = "nuevo valor";
```

Para insertar nuevos elementos en la estructura del árbol se utilizan los métodos createElement y createTextNode, que se describen en las tablas 3 y 4 respectivamente:

Tabla 3 método createElement

Método **createElement**	
Sintaxis	`var elemento = arbolDOM.createElement(nElemento);`
Parámetros	- **nElemento:** String que contiene el nombre del elemento que se desea crear (table, div, etc.). Este parámetro es requerido.
Retorno	Retorna la referencia al elemento creado.
Observaciones	En el navegador Microsoft IE 4.0 los únicos elementos que pueden ser creados son img, area y option. Para la versión de IE 5.0 y posteriores es posible crear cualquier elemento con la excepción de los elementos frame o iframe.

Tabla 4 método createTextNode

Método **createTextNode**	
Sintaxis	`var elemento = arbolDOM.createTextNode(texto);`
Parámetros	- **texto**: String que contiene el texto que se desea agregar al elemento. Este parámetro es opcional.
Retorno	Retorna la referencia al elemento de texto.
Observaciones	(ninguna)

Para agregar los elementos creados a la estructura de árbol se utiliza el siguiente método presentado en la Tabla 5:

Tabla 5 método appendChild

Método **appendChild**	
Sintaxis	`var element = arbolDOM.appendChild(elemento);`
Parámetros	- **elemento:** variable que contiene la referencia del elemento que se desea agregar. Este parámetro es requerido.
Retorno	Retorna la referencia al elemento agregado.
Observaciones	Si el elemento al cual se desea agregar el nuevo elemento ya posee otros elementos, éste es agregado al final de los mismos.

Son muchos los cambios que se pueden realizar al documento mediante el uso del árbol DOM, pudiéndose agregar, eliminar o modificar elementos, así como modificar las propiedades de los mismos. Es importante mencionar que los métodos descritos anteriormente también aplican para documentos XML.

A continuación se muestra un extracto de código HTML que hace uso de la API de JavaScript para agregar elementos al DOM mostrado en la Figura 4:

```
<html>
 <head>
 <title>Ejemplo DOM</title>
 <script language="JavaScript">
 function agregarElementos(){

 var arbolDOM=document;
 var elementoNuevo=arbolDOM.createElement("div");
 var elementoViejo =arbolDOM.getElementById("ejemplo");
 var texto="Este es un nuevo elemento";
 var txtNodo=document.createTextNode(texto);
 elementoNuevo.appendChild(txtNodo);
 elementoViejo.appendChild(ElementoNuevo);

 var body=document.getElementsByTagName("BODY");
 texto="Este es un nuevo elemento dentro del body";
 var txtNodo=document.createTextNode(texto);
 body[0].appendChild(txtNodo);

 }
 </script>
 </head>
 <body onload="agregarElementos()">
 <h1>Ejemplo DOM</h1>
 <p>Esta pagina muestra un <em>ejemplo</em> de la estructura de arbol
DOM.</p>
 <div id="ejemplo">Esto es un texto de ejemplo dentro de una etiqueta
DIV</div>
 </body>
</html>
```

Tabla 6 Extracto de codigo HTML y JavaScript de Ejemplo

En la figura 5 se muestra el árbol DOM asociado al extracto de código de la figura 6:

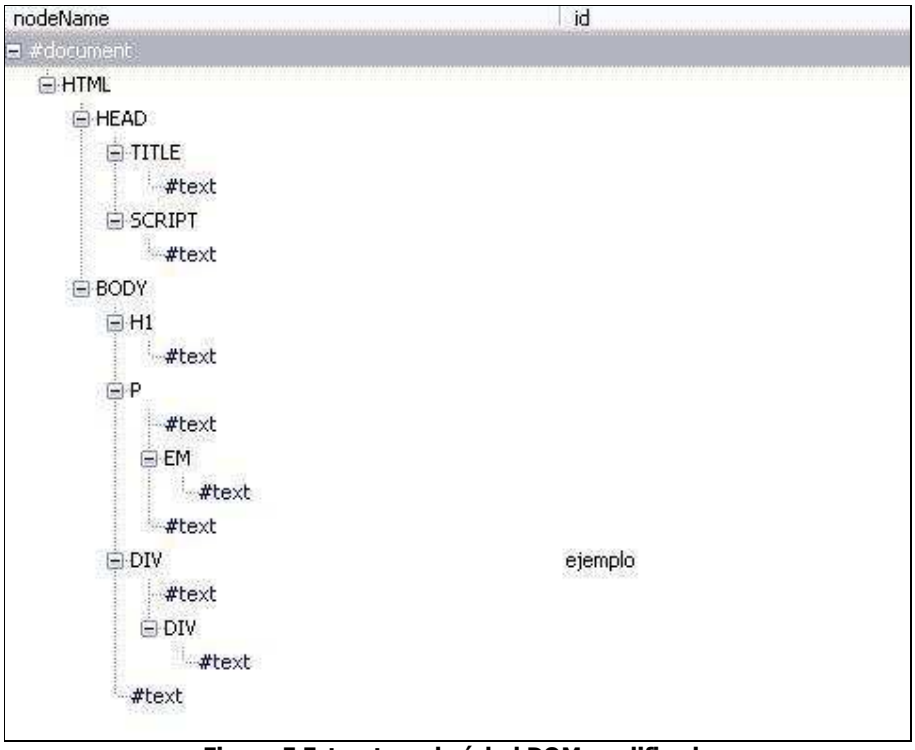


Figura 5 Estructura de árbol DOM modificada

Luego de conocer las principales características y el funcionamiento del enfoque AJAX se pueden encontrar cuatro principios que engloban estas características y permiten conocer la diferencia entre una aplicación AJAX y una aplicación Web tradicional los cuales se describen a continuación.

1.2.4. Algunos principios de AJAX

A continuación se describen cuatro principios de AJAX descritos en [4], que resumen un conjunto de características deseables en el enfoque AJAX que lo hacen diferenciarse de las aplicaciones Web tradicionales.

En el navegador reside una aplicación, no sólo estructura y contenido

En las aplicaciones Web basadas en AJAX, a diferencia de las aplicaciones Web tradicionales, existe un componente del lado del cliente denominado motor AJAX, el cual tiene la capacidad de saber cuándo responder a las acciones del usuario, es decir, éste puede decidir cuándo ocuparse de manejar estas acciones dándoles una respuesta directa y cuándo hacer peticiones al servidor Web para obtener la respuesta.

Con el enfoque AJAX la interacción con la aplicación tiende a enriquecerse, gracias a que gran parte de la lógica de presentación es delegada en el cliente, otorgándole a éste una mayor responsabilidad que la que tiene normalmente en las aplicaciones Web clásicas.

Cuando el usuario accede a una aplicación basada en AJAX, el navegador recibe un documento más complejo que el que se puede recibir en las aplicaciones Web tradicionales. Este documento puede contener una gran parte de código JavaScript, que puede permanecer en uso en la sesión del usuario durante varias peticiones al servidor; incluso puede modificarse considerablemente a sí mismo para presentar cambios en la interfaz de usuario.

El servidor transfiere datos relevantes, la estructura y formato se encuentran en el cliente

En el esquema clásico de aplicaciones Web el servidor envía en cada respuesta una típica mezcla de contenido e información de formato.

Las aplicaciones AJAX cambian este comportamiento, permitiendo hacer un mejor uso de los recursos y obtener un mejor rendimiento, utilizando peticiones asíncronas al servidor y obteniendo de este último solamente los datos relevantes, manteniendo la estructura del documento y la información de formato en el cliente. Las aplicaciones AJAX pueden realizar esto retornando código JavaScript, texto plano o documentos XML.

La interacción con la aplicación debe ser fluida y continua

Para obtener cierto comportamiento dinámico las aplicaciones Web tradicionales hacen uso de JavaScript para ciertas tareas, como validación de campos de formularios y modificación de elementos de la página. Sin embargo, este código está presente únicamente mientras la página lo esté, y se pierde entre las peticiones al servidor. Para interactuar con el servidor, estas aplicaciones Web hacen uso de peticiones síncronas, que interrumpen el flujo de interacción entre el usuario y la aplicación, sustituyendo la página actual por la respuesta del servidor en cada petición.

Las aplicaciones AJAX funcionan de una forma distinta, ya que éstas pueden realizar peticiones al servidor sin necesidad de perder el código JavaScript que reside en la página en uso (gracias a las peticiones asíncronas); además de esto, las aplicaciones

AJAX pueden modificar la interfaz dinámicamente sin solicitar una nueva página al servidor manteniendo mayor continuidad en la interacción.

Con el enfoque AJAX, conceptos más sofisticados de interfaces de usuario llegan a ser factibles, tales como *drag & drop*, menús desplegables, paneles, entre otros, llegando a hacer la experiencia del usuario más parecida a la que obtiene con las aplicaciones de escritorio.

Es codificación real y requiere disciplina

El uso que se le ha dado a JavaScript en muchas aplicaciones Web tradicionales le ha dado una mala reputación al lenguaje, llegando a ser visto por muchos desarrolladores como un lenguaje trivial o peligroso.

Sin embargo, esto no es así. El código JavaScript de una aplicación AJAX comúnmente es complejo, siendo una pieza de suma importancia la cual debe funcionar hasta que la aplicación sea cerrada, sin causar retrasos, colapsos de memoria y sin romper el flujo de interacción con el sistema, para comunicarse de manera eficiente con el servidor y a su vez proveer una buena interacción con el usuario.

1.2.5. Visión general del funcionamiento

la Figura 6 presenta un diagrama de secuencia que representa el funcionamiento de una aplicación AJAX, donde se muestra la interacción entre los distintos componentes de la misma.

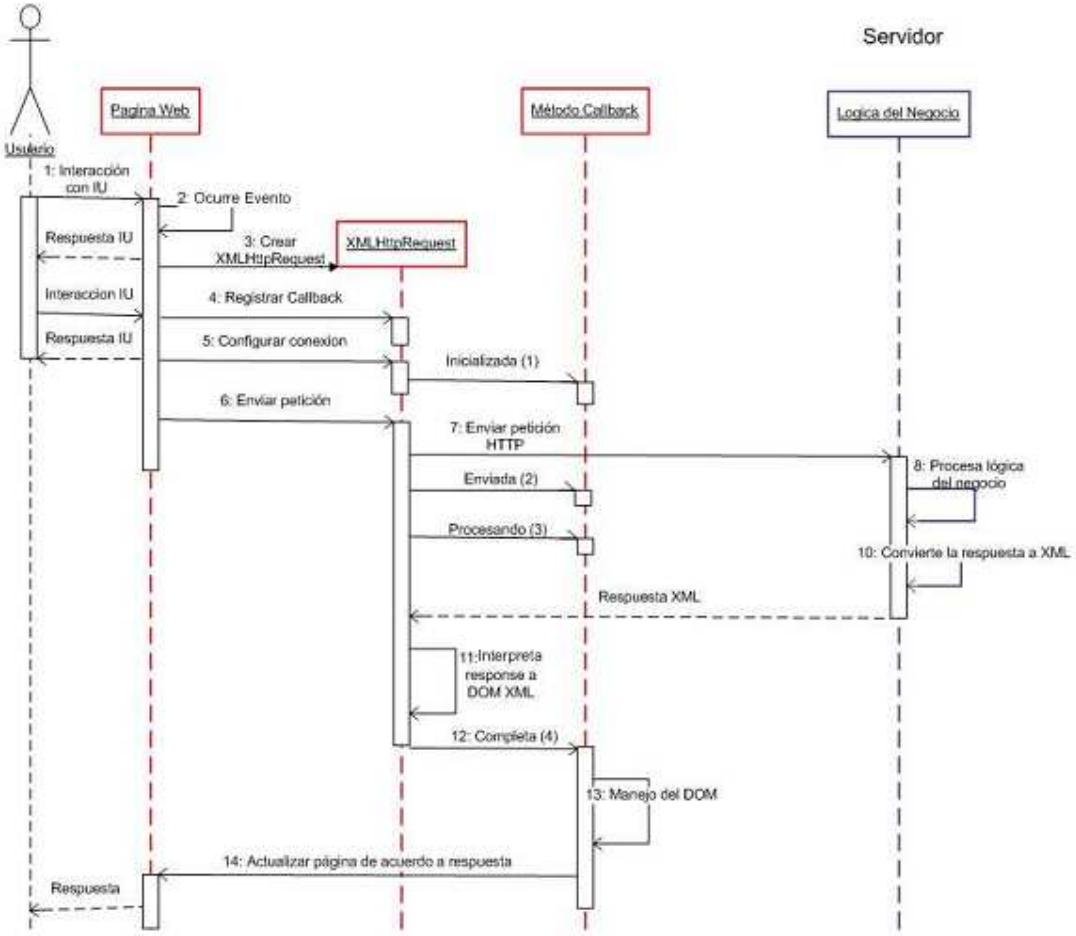


Figura 6 Diagrama de secuencia de aplicación AJAX

En la figura 6 se muestra la interacción entre los distintos componentes de una aplicación AJAX. Los componentes Pagina web, XMLHttpRequest y el metodo callback conforman el motor AJAX, mientras que el componente de logica de negocios se encuentra del lado del servidor. Cuando el usuario interactúa con la página Web, ésta produce un evento que es capturado por una función JavaScript, la cual se encarga de crear y configurar una petición según el evento capturado, retornando al mismo tiempo una respuesta al usuario. Después de configurar la petición, el motor AJAX se encarga de enviarla al servidor y de seguir interactuando con el usuario mientras ésta se procesa. El usuario puede realizar diferentes acciones sobre la página mientras la

aplicación espera la respuesta del servidor, y el motor AJAX puede darle respuesta a muchas de estas acciones antes que llegue la misma.

1.2.6. Consideraciones de interacción con el usuario en el desarrollo de aplicaciones AJAX

A lo largo de este capítulo se ha hecho referencia a una serie de bondades que ha traído el uso de AJAX al desarrollo de aplicaciones Web. Sin embargo, existen algunos aspectos a considerar respecto a como se comportan las aplicaciones Web basadas en AJAX que pueden representar los puntos débiles de este esquema. A continuación se mencionan dichos puntos y sus implicaciones, así como sus posibles soluciones:

El botón atrás/adelante

Uno de los argumentos contra el uso de AJAX en aplicaciones Web, es que puede fácilmente acabar con el comportamiento normal del botón atrás/adelante del navegador. Los usuarios, normalmente, esperan que haciendo clic en el botón atrás del navegador, mientras utilizan una aplicación Web, volverán a la última página cargada; en las aplicaciones AJAX lo más seguro es que esto no ocurra, debido a que gran parte de las páginas que conforman la aplicación Web son generadas dinámicamente del lado del cliente mediante el uso JavaScript, con base en una misma página HTML.

Los desarrolladores han implementado varias soluciones a este problema, muchas de las cuales giran en torno al uso de iframes escondidos para invocar los cambios que han tenido lugar, emulando el historial usado por el botón atrás/adelante del navegador.

Uso de marcadores y favoritos

Otro problema relacionado con lo planteado anteriormente es que las actualizaciones dinámicas hacen difícil al usuario agregar a los marcadores/favoritos un contenido particular de la aplicación: el URL marcado puede no llevar al contenido esperado si es accedido en otro momento.

Existen soluciones para este problema, muchas de las cuales utilizan el fragmento identificador del URL (la parte de un URL precedida por el signo numeral, "#") para no perder las páginas de vista, y permitir a los usuarios volver a ese momento exacto.

Esto es posible porque muchos navegadores permiten al código JavaScript actualizar dinámicamente el fragmento identificador del URL, por lo que las aplicaciones AJAX pueden mantenerlo a medida que el usuario va cambiando el estado de las páginas. Esta solución también mejora el funcionamiento del botón atrás.

Uso de los formularios

Los formularios basados en el objeto XMLHttpRequest no actúan de la manera a la que el usuario está acostumbrado. En un formulario clásico, es posible hacer y deshacer los cambios que se realicen sobre algún campo particular (cuadros de texto, cajas de chequeo, botones de selección, entre otros) pues se sabe que, hasta que no se pulse el botón *enviar*, se pueden hacer los cambios que hagan falta sobre estos datos sin que esto se refleje del lado del servidor. Pero AJAX no funciona así, ya que los datos pueden enviarse campo a campo; esto desvirtúa el concepto clásico de formulario e introduce un elemento que puede generar perturbación y confundir al usuario.

Una posible solución sería no mandar al servidor los datos del formulario hasta que el usuario presione el botón de *enviar*, pero esto rompería con el esquema asíncrono de las aplicaciones AJAX y se estaría en presencia de una aplicación Web síncrona. Así, otra posible solución que se plantea es enviar, de manera asíncrona, únicamente el contenido de los campos del formulario que necesiten una validación por parte del servidor.

Notificación de cambios

La carga instantánea de diversas partes de una misma página puede producir en el usuario la sensación de que nada ha pasado, pues la carga se ha dado de una forma muy rápida como para que pudiera percibirla, teniéndose además que el URL no ha variado. Mediante el uso de AJAX la actualización de la página es parcial y la percepción del cambio es mucho menos notoria, de ahí que ante cambios tenues de la interfaz sea necesario incorporar señales sutiles para que el usuario perciba que algo ha cambiado en la página.

Una posibilidad es utilizar colores para denotar los cambios que se hayan producido; sin embargo, el uso de colores para denotar la interacción deberá contar con el contraste suficiente para que el cambio sea percibido. Otra alternativa es introducir un pequeño retardo de forma artificial a la carga de datos en la página, para ofrecer la

habitual sensación de una página que se descarga lentamente. Sin embargo, el retraso debe ser lo suficientemente leve como para no disminuir la velocidad general del sistema y lo suficientemente largo como para que el usuario perciba correctamente el cambio en el estado del sistema (la sensación de recarga de la página).

Validación de datos

El desacoplamiento que ofrece AJAX del cliente respecto al servidor puede tentar al desarrollador a mantener todas las validaciones de los datos del lado del cliente, dejando al servidor únicamente la tarea de procesar y/o almacenar los mismos. Implementar el control de seguridad únicamente del lado del cliente es una técnica inadecuada y peligrosa, debido a que existe la posibilidad de que intrusos (atacantes) puedan modificar el código de manera maliciosa y esquivar dichos controles.

Una práctica recomendada para la implementación de controles de seguridad es que las validaciones de los datos sean hechas en ambos lados, es decir, tanto en el cliente como en el servidor.

Notificación de anomalías

Debido al comportamiento asíncrono de las aplicaciones AJAX, podría estar ocurriendo alguna actividad no deseada por el usuario sin que el mismo se percate de este hecho, percibiéndose únicamente un retardo en la respuesta. En una aplicación Web tradicional, el usuario observaría por un intervalo de tiempo prolongado un documento HTML en blanco; sin embargo, en las aplicaciones Web basadas en AJAX esto no ocurriría, y el usuario no se percataría de la condición anormal.

Debido a esta situación, es necesario implementar un mecanismo para informar al usuario sobre la condición anormal que esta ocurriendo.

Luego de conocer el funcionamiento y las bondades que brinda el enfoque AJAX, y cómo éste revoluciona el comportamiento de las aplicaciones Web, se presenta a continuación un conjunto de patrones y buenas prácticas de programación en el desarrollo de aplicaciones bajo este enfoque.

1.2.7. Comparación de tecnologías

Luego de haber realizado una descripción del enfoque AJAX se procede a realizar una comparación entre este enfoque y las tecnologías para el desarrollo de RIAs descritas en el capítulo 1.

Enfoque	**Tecnología en el cliente**	**Lenguaje de Scripting**	**Uso de *plugin***	**Dependiente del navegador**
Laszlo	Flash y XML	LZX	Si	No
Mozilla XUL	Lenguaje XUL (XML) y DOM	JavaScript	No	Si
XForms	Xform (XML) y DOM	JavaScript	Si	No
Macromedia Flex	Flash y MXML (XML)	ActionScript 2	Si	No
AJAX	XML, DOM y CSS	JavaScript	No	No

Tabla 7 Comparación AJAX-tecnologías RIAs

En la tabla 7 se pueden apreciar las distintas características correspondientes a cada enfoque y se aprecia que el enfoque AJAX destaca su independencia de los navegadores y que no se necesita de ningún plugin especial para su correcto funcionamiento, por lo hace que este sea uno de los esquemas mas versátiles al momento de ser implementado y de aceptar por cualquier usuario. Razones por la cual fue escogido como el esquema a desarrollar en el caso de estudio de este Trabajo Especial de Grado.

1.3. Metodología de Desarrollo

El proceso de desarrollo software es una de las áreas de investigación más importantes para la comunidad de ingeniería del software. Continuamente aparecen nuevos trabajos y propuestas que definen distintas aproximaciones para el proceso de desarrollo de software. Sin embargo, es difícil que satisfagan todas las necesidades de un proyecto específico. Teniendo en cuenta que dos proyectos pueden ser muy diferentes, el proceso aplicado con éxito en uno de ellos puede ser un completo fracaso en el otro. Por eso, el proceso software debe ser adaptado al contexto y características específicas de cada caso. A Continuación se explicara mas a fondo el método de desarrollo del Proceso unificado el cual es un método basado en estos fundamentos.

1.3.1. El Proceso Unificado

El Proceso Unificado de Desarrollo Software o simplemente Proceso Unificado es un marco de desarrollo de software que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El refinamiento más conocido y documentado del Proceso Unificado es el Proceso Unificado de Rational o simplemente RUP. El Proceso Unificado no es simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos. El nombre Proceso Unificado se usa para describir el proceso genérico que incluye aquellos elementos que son comunes a la mayoría de los refinamientos existentes [8].

Este provee un enfoque disciplinado en la asignación de tareas y responsabilidades dentro de una organización de desarrollo. Su meta es asegurar la producción de software de muy alta calidad que satisfaga las necesidades de los usuarios finales, dentro de un calendario y presupuesto predecible.

1.3.2. Características

Las características principales del proceso unificado son las siguientes

Iterativo e Incremental

El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones. Estas iteraciones ofrecen

como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

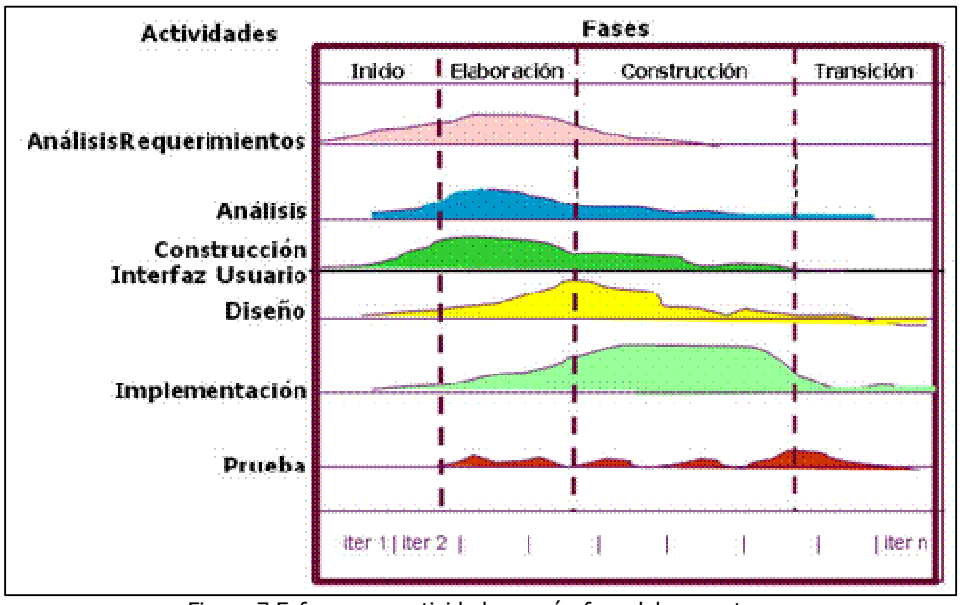


Figura 7 Esfuerzo en actividades según fase del proyecto

Dirigido por los casos de uso

En el Proceso Unificado los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc. el proceso dirigido por casos de uso es el RUP.

Centrado en la arquitectura

El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.

Enfocado en los riesgos

El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración, deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

1.3.3. Etapas del Proceso Unificado

El proceso unificado podria dividirse en dos importantes fases como son las siguientes:

Etapa de ingeniería

Esta etapa agrupa las fases de concepción y de elaboración, lo que básicamente le da por objetivos la conceptualización del sistema y el diseño inicial de la solución del problema.

Se inicia el proceso de administración de los requerimientos con la identificación y especificación de casos de usos, así como el proceso de aseguramiento de la calidad a través de los casos de prueba.

Se identifican los riesgos y se establece su plan de manejo, para determinar en qué orden y en qué iteraciones se desarrollarán los artefactos de software que son la solución a los casos de uso.

Se identifican los recursos necesarios, tanto económicos como humanos, acordes con las necesidades del proyecto. Se da comienzo al proceso de estimación y planificación inicia la un nivel macro para todo el proyecto y posteriormente se realiza una estimación detallada de tiempos y recursos de las fases de concepción y elaboración.

- **Fase de concepción**

  Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones.

- **Fase de elaboración**

  Los casos de uso seleccionados para desarrollarse en esta fase permiten definir la arquitectura del sistema, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la

solución preliminar del problema y comienza la ejecución del plan de manejo de riesgos, según las prioridades definidas en él. Al final de la fase se determina la viabilidad de continuar el proyecto y si se decide proseguir, dado que la mayor parte de los riesgos han sido mitigados, se escriben los planes de trabajo de las etapas de construcción y transición y se detalla el plan de trabajo de la primera iteración de la fase de construcción.

Etapa de producción

En esta etapa se realiza un proceso de refinamiento de las estimaciones de tiempos y recursos para las fases de construcción y transición, se define un plan de mantenimiento para los productos entregados en la etapa de ingeniería, se implementan los casos de uso pendientes y se entrega el producto al cliente, garantizando la capacitación y el soporte adecuados.

- **Fase de construcción**

  El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar el cambio de los artefactos construidos, ejecutar el plan de administración de recursos y mejoras en el proceso de desarrollo para el proyecto.

- **Fase de transición**

  El propósito de esta fase es asegurar que el software esté disponible para los usuarios finales, ajustar los errores y defectos encontrados, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto al inicio del mismo

1.3.4.Principios fundamentales del Proceso Unificado

RUP está basado en 5 principios clave que son:

Adaptar el proceso

El proceso deberá adaptarse a las características propias del proyecto u organización. El tamaño del mismo, así como su tipo o las regulaciones que lo condicionen, influirán en su diseño específico. También se deberá tener en cuenta el alcance del proyecto.

Balancear prioridades

Los requerimientos de los diversos participantes pueden ser diferentes, contradictorios o disputarse recursos limitados. Debe encontrarse un balance que satisfaga los deseos de todos. Debido a este balanceo se podrán corregir desacuerdos que surjan en el futuro.

Demostrar valor iterativamente

Los proyectos se entregan, aunque sea de un modo interno, en etapas iteradas. En cada iteración se analiza la opinión de los inversores, la estabilidad y calidad del producto, y se refina la dirección del proyecto así como también los riesgos involucrados

Elevar el nivel de abstracción

Este principio dominante motiva el uso de conceptos reutilizables tales como patrón del software, lenguajes marcos de referencia (frameworks) por nombrar algunos. Esto evita que los ingenieros de software vayan directamente de los requisitos a la codificación de software a la medida del cliente, sin saber con certeza qué codificar para satisfacer de la mejor manera los requerimientos y sin comenzar desde un principio pensando en la reutilización del código. Un alto nivel de abstracción también permite discusiones sobre diversos niveles y soluciones arquitectónicas. Éstas se pueden acompañar por las representaciones visuales de la arquitectura, por ejemplo con el lenguaje UML.

Enfocarse en la calidad

El control de calidad no debe realizarse al final de cada iteración, sino en todos los aspectos de la producción. El aseguramiento de la calidad forma parte del proceso de desarrollo y no de un grupo independiente.

1.3.5. Adaptación al Contexto del Proceso Unificado

El Proceso Unificado no es un sistema con pasos firmemente establecidos, sino un conjunto de metodologías adaptables al contexto y necesidades de cada proyecto, por ende este se puede adaptar, a un tipo de proyecto especifico.

La peculiaridad principal del proceso unificado es que sea orientado a casos de uso y a su vez se cumplan las etapas correspondientes a la creación de artefactos de software (Iteraciones) y se sigan los principios de la metodología.

El uso de diagramas para apoyar el diseño de la aplicación así como el uso de elementos externos que den soporte a un mejor entendimiento del sistema a desarrollar queda a conveniencia del grupo de desarrolladores.

Una adaptación particular de esta metodología será la utilizada para el desarrollo del sistema que constituirá este Trabajo Especial de Grado y la cual será explicada con mayor detalles en capitulo 2 del marco aplicativo.

Capítulo 2. Marco Aplicativo

El tiempo empleado por el los miembros de una institución educativa representa una importante inversión, aparte de los materiales empleados para hacer posible todos los procesos dentro de una institución (Papel, tinta, etc.). Por lo que cada vez se busca agilizar los procesos que ocurren a diario en el entorno escolar, en este caso el de control, creación, aprobación y distribución de los planes de evaluación del periodo escolar. En este capítulo se presenta un análisis sobre el proceso de gestión de planes de evaluación así como la entrega de notas de las actividades asignadas durante el periodo escolar correspondiente, proponiendo una solución para el desarrollo de una aplicación Web enriquecida. Dicha solución está basada en el uso del enfoque de desarrollo AJAX, con la intención de obtener una mayor riqueza en la interacción del usuario con la aplicación.

En la sección 3.1 se plantea la metodología utilizadaza para el desarrollo del sistema, y cuales son las fases de desarrollo que se llevaran a cabo, en la sección 3.2 se realiza la captura de los requerimientos para la solución a desarrollar, proporcionando una visión global del escenario. La sección 3.3 se dedica al análisis de los requerimientos capturados para cubrir el escenario planteado. Para culminar el capítulo, la sección 3.4 cubre lo referente al diseño de la solución, base fundamental para la implementación.

2.1. Metodología utilizada

El objetivo de esta sección consiste en definir la metodología utilizada para el desarrollo e implementación de El Sistema de Gestión Estudiantil.

2.1.1. Adaptación del Proceso Unificado

En el proceso unificado se plantea que dado el tipo de aplicación, se puede instanciar esta metodología para ajustar el proceso a las exigencias de los desarrolladores. En este trabajo se realiza los diferentes modelos del análisis de requerimientos, a partir de estos se desarrolló el prototipo de interfaz y posteriormente, se procedió a implementar cada uno de los escenarios, adaptándose en esta etapa a un proceso de desarrollo ágil. Se establecieron las siguientes etapas [7]:

Etapa de Ingeniería, se inició el proceso de captura de los requerimientos para el sistema de gestión escolar, una etapa de análisis donde se identificaron los Actores y los Casos de Uso. Se realizó un refinamiento de los casos de uso principales y se

desarrolló el Modelo de Datos a ser utilizado, seguido de una etapa de diseño en la que se identificaron las tecnologías a utilizar y se realizaron los diagramas de componentes y de despliegue, determinando luego en qué orden y en qué iteraciones se desarrollaron los artefactos de software que son la solución de los Casos de Uso.

Etapa de Producción, según el plan establecido en la etapa anterior se empezó con la implementación de cada uno de los escenarios, hasta completar las funcionalidades del sistema. Por último, se comenzó con una fase de transición la cual consistió en asegurar la aceptación del producto por los usuarios finales, ajustar los errores y defectos encontrados para asegurar que el producto cumpla con los requerimientos del usuario.

2.2. Captura de requerimientos

La institución C.E.A.P.U.C.V. tiene la necesidad de planificar y controlar la creación y aprobación de los planes de evaluación durante el periodo escolar, así como la entrega de notas por parte de los profesores, basándose en como se maneja este proceso en la actualidad. Este proceso es realizado de forma manual y presencial por los profesores y la Oficina de Control y Evaluación.

En la búsqueda por automatizar este proceso y haciendo uso de las tecnologías actuales, como el Internet, las aplicaciones Web y el surgimiento de los nuevos esquemas de interacción que provee el entorno RIA se puede desarrollar una aplicación que solucione este problema.

Dados estos factores y con base en un análisis constituido por una serie de entrevistas con la comunidad y posibles usuarios finales de la aplicación surgen un conjunto de requerimientos, que serán desarrollados bajo la plataforma planteada en la propuesta de TEG.

2.2.1. Requerimientos

Luego de una serie de reuniones con el personal del C.A.P.U.C.V y un estudio del proceso de creación y aprobación de los planes de evaluación dentro del plantel se hizo un levantamiento de los siguientes requerimientos para el sistema.

- Se quiere que los profesores puedan crear, modificar y eliminar, el plan de evaluación correspondiente a cada lapso de cada materia y curso que están dictando, los cuales debe cumplir con los parámetros establecidos por la institución (como que el total de porcentaje a ser evaluado en un lapso debe

cumplir un 100% y debe tener mínimo un examen de lapso, etc.), una ves terminado el plan de evaluación el profesor debe poder colocarlo para su posterior revisión por parte de la Oficina de Coordinación y Evaluación y esperar por su aprobación.

- Se quiere que una vez que los planes de evaluación hayan sido aprobados por la oficina de Coordinación y Evaluación, el profesor pueda ingresar en el sistema las notas correspondientes a cada evaluación de cada alumno, una ves cargadas todas las notas el profesor deberá enviarlas por el sistema a la oficina de Coordinación y Evaluación, el profesor debe enviarlas antes de una fecha fijada por la Oficina la cual se le mostrara a los profesores por medio del sistema y se le enviaran alertas cuando la fecha tope de publicación de notas se encuentre próxima.
- Se quiere que los representantes puedan ver el resultado de las evaluaciones, una vez que estas hayan sido aprobadas por la oficina de evaluación, durante todo el periodo del año escolar.
- Se quiere que los usuarios de la oficina de control y evaluación puedan ver, aprobar, evaluar o rechazar los planes de evaluación sugeridos por los profesores al principio del año escolar o cada lapso, así como cualquier modificación que se le haga a los mismos.
- Se quiere que los usuarios de la oficina de control y evaluación puedan fijar las fechas de entrega de notas por parte de los profesores, fecha de exámenes de reparación, y de publicación de planes de evaluación etc.

Una vez conocidos los requerimientos que debe cumplir el sistema para llevar a cabo el proceso de creación de planes de evaluación y entrega de notas se pasa a la etapa de análisis.

2.3. Fase de Análisis

Luego de plantear el escenario global y las funcionalidades básicas de la solución que se desarrollará, se presenta un análisis más profundo del escenario, planteando los sub-escenarios presentes y cómo se enfocará cada requerimiento. Para realizar este análisis se hará uso del modelo funcional y del modelo estructural que provee el lenguaje de modelado unificado (UML), mediante la realización de los siguientes pasos:

- Construcción de diagramas de Casos de Uso.
- Construcción de Modelo de Datos.

Para la construcción de los diagramas se utilizó la versión 2.0 de UML para aprovechar las nuevas características que esta versión provee [10].

2.3.1. Modelo de Casos de Uso

En esta sección se presenta los casos de uso del Sistema de Gestion Escolar, en términos de los puntos de interacción del usuario con la aplicación; para ello se precisarán los actores, es decir, los tipos de usuarios que utilizan el sistema, luego se presenta cada uno de los casos clasificados por niveles y seguido la descripción particular de los casos de uso presentes en el mismo.

Actores

En el diagrama de casos de uso se identifican los siguientes actores:

Profesor: usuario encargado de la creación de los planes de evaluación para cada materia, así como la carga de notas de cada actividad presente en el plan de evaluación.

Oficina de Control y Evaluación: usuario con privilegios para aprobar o rechazar los planes de evaluación creados por los usuarios profesores y enviados a ser evaluados por este usuario.

Representante: usuario que forma parte de la comunidad estudiantil, el cual puede consultar las notas de todas las actividades de los alumnos durante el periodo escolar.

Administrador: usuario que se encarga del manejo de la aplicación, con tareas como agregar usuarios, cursos, materias, asignar profesores a cada materia, asignar alumnos a cada curso, etc.

Diagrama de casos de Uso

En la figura 9 se muestra el diagrama de casos de uso correspondiente al levantamiento de requerimientos realizado para la aplicación que se desea desarrollar.

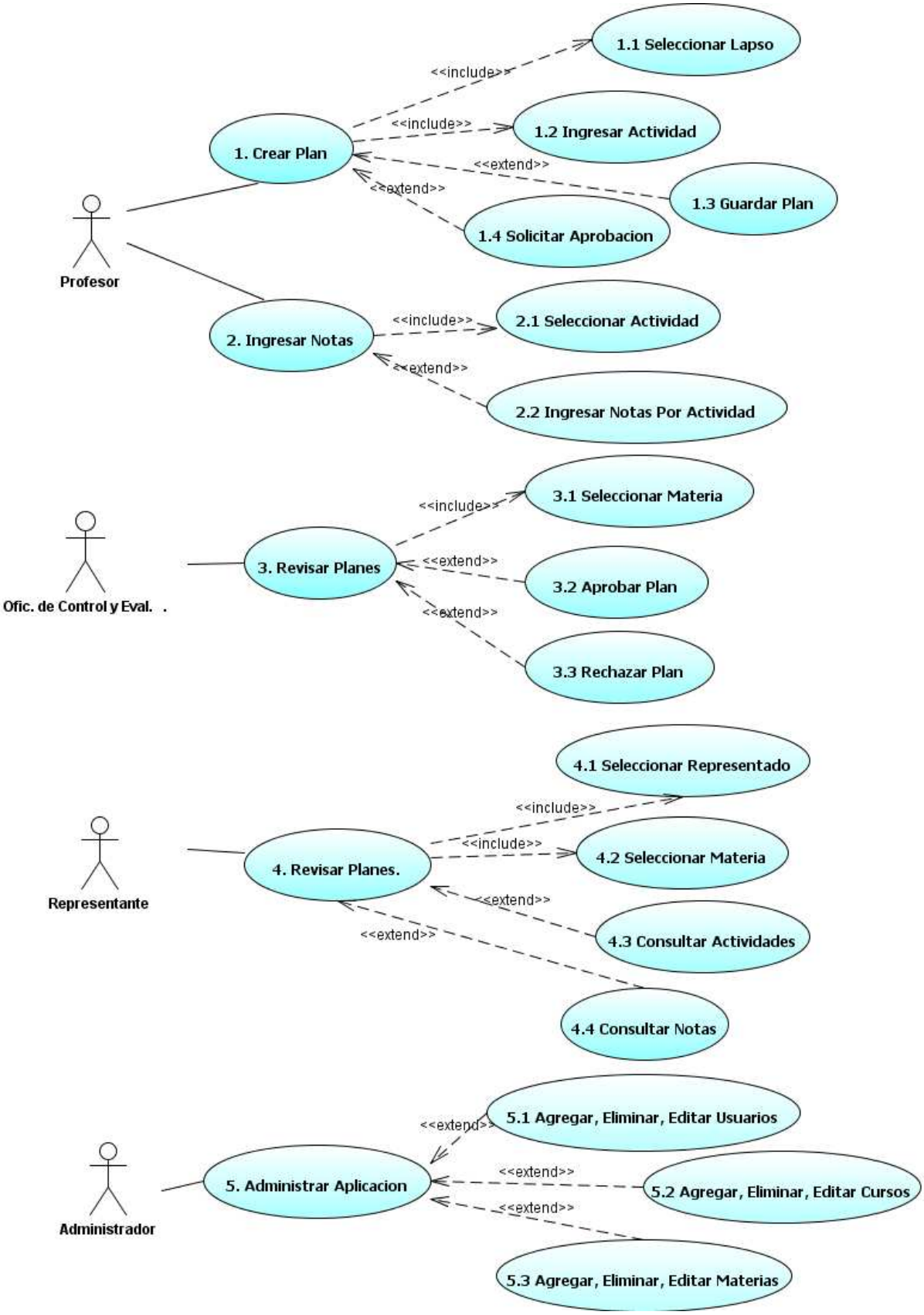


Figura 8 Diagrama de Casos de Uso del Sistema

Descripción de los casos de uso

Caso de Uso	1. Crear Plan de Evaluación
Actor	Profesor
Descripción	El usuario profesor puede crear un plan de evaluación, guardarlo o solicitarle su aprobación a la oficina de control y evaluación
Precondiciones	El Profesor debe Estar Autenticado
Condición Final Exitosa	Se muestran las opciones para agregar actividades al plan de evaluación, así como opciones para guardarlo o solicitar su aprobación

Caso de Uso	1.1 Seleccionar lapso
Actor	Profesor
Descripción	El usuario profesor debe seleccionar el lapso (periodo escolar) de la materia en la que desea crear el plan de evaluación
Precondiciones	Debe estar creada una materia en la base de datos y estar relacionada con el profesor
Condición Final Exitosa	Se muestran las opciones para agregar actividades al plan de evaluación, así como opciones para guardarlo o solicitar su aprobación

Caso de Uso	1.2 Ingresar Actividades
Actor	Profesor
Descripción	El usuario profesor debe ingresar todas las actividades a ser evaluadas durante el lapso, debe ingresar su descripción, porcentaje, fecha y tipo de actividad
Precondiciones	Debe estar creado o guardado un plan de evaluación
Condición Final Exitosa	Se puede consultar en momento todas las actividades

	ingresadas al plan de evaluación.

Caso de Uso	1.3 Guardar Plan de Evaluación
Actor	Profesor
Descripción	EL usuario puede guardar el plan de evaluación para modificarlo, agregar actividades posteriormente o ser enviado a revisión en otra ocasión
Precondiciones	Se debe haber agregado una o más actividades al plan de evaluación.
Condición Final Exitosa	Se puede consultar en momento todas las actividades ingresadas al plan de evaluación.

Caso de Uso	1.5 Solicitar Aprobación
Actor	Profesor
Descripción	EL usuario envía el plan de evaluación para que sea revisado por la oficina de control y evaluación.
Precondiciones	Se deben haber agregado actividades al plan de evaluación y que todas ellas en total sumen un 100% de ponderación a evaluar durante el lapso
Condición Final Exitosa	Se coloca el plan de evaluación en un estado de espera a ser aprobado.

Caso de Uso	2. Ingresar Notas
Actor	Profesor
Descripción	El usuario puede ingresar notas de los alumnos en las actividades correspondientes al plan de evaluación del lapso correspondiente.
Precondiciones	El plan de evaluación debe haber sido aprobado por la oficina de control y evaluación.
Condición Final Exitosa	Las notas son cargadas al sistema y pueden ser vistas y

	modificadas por el usuario profesor y ser consultadas por el usuario representante y la Oficina de Control y Evaluación.

Caso de Uso	2.1 Seleccionar plan de evaluación
Actor	Profesor
Descripción	El usuario debe seleccionar el plan de evaluación en el cual desea ingresar las notas.
Precondiciones	El plan de evaluación debe haber sido aprobado por la oficina de control y evaluación.
Condición Final Exitosa	Se muestra un listado con las actividades correspondientes a ese plan de evaluación.

Caso de Uso	2.2 Ingresar Notas de actividades
Actor	Profesor
Descripción	El usuario puede ingresar las notas de una actividad que haya seleccionado de un plan de evaluación
Precondiciones	Debe haber seleccionado un plan de evaluación para poder ver el listado de actividades.
Condición Final Exitosa	Se guardan en el sistema las notas correspondientes a los alumnos que presentaron la actividad, en que se introdujeron las notas

Caso de Uso	3. Revisar Planes de Evaluación
Actor	Oficina de Control y Evaluación
Descripción	El usuario puede revisar los planes de evaluación que han sido enviados por los profesores con la finalidad de ser aprobados.
Precondiciones	El plan de evaluación debe haber sido enviado a revisión

	por parte de un usuario de tipo profesor.
Condición Final Exitosa	Se guardan en el sistema las notas correspondientes a los alumnos que presentaron la actividad, en que se introdujeron las notas

Caso de Uso	3.1 Seleccionar materia
Actor	Oficina de Control y Evaluación
Descripción	El usuario debe seleccionar la materia a la cual desea hacerle revisión de el plan de evaluación
Precondiciones	El plan de evaluación de la materia seleccionada debe haber sido enviado a revisión por parte del usuario de tipo profesor asociado a la materia.
Condición Final Exitosa	Se muestra el plan de evaluación y las actividades correspondientes a el.

Caso de Uso	3.2 Aprobar Plan
Actor	Oficina de Control y Evaluación
Descripción	El usuario puede aprobar el plan de evaluación de la materia seleccionada.
Precondiciones	El plan de evaluación de la materia seleccionada debe haber sido enviado a revisión por parte del usuario de tipo profesor asociado a la materia.
Condición Final Exitosa	Se le cambia el estado al plan de evolución y se le notifica al profesor que ha sido aprobado el plan.

Caso de Uso	3.3 Rechazar Plan
Actor	Oficina de Control y Evaluación
Descripción	El usuario puede rechazar el plan de evaluación de la materia seleccionada.

Precondiciones	El plan de evaluación de la materia seleccionada debe haber sido enviado a revisión por parte del usuario de tipo profesor asociado a la materia.
Condición Final Exitosa	Se le cambia el estado al plan de evolución y se le notifica al profesor que ha sido rechazado el plan opcional el usuario de la Oficina de Control y Evaluación puede enviarle una observación al usuario.

Caso de Uso	4. Consultar Notas
Actor	Representante
Descripción	El usuario puede consultar las notas acumuladas de su(s) representado(s) durante el transcurso del periodo escolar
Precondiciones	Se deben haber cargado las notas por parte del usuario profesor al alumno asociado al usuario representante
Condición Final Exitosa	Se le muestran las notas correspondientes a cada alumno de cada actividad de el lapso seleccionado

Caso de Uso	4.1 Seleccionar representado
Actor	Representante
Descripción	El usuario debe seleccionar el representado del cual desea consultar sus notas.
Precondiciones	Se deben haber cargado las notas por parte del usuario profesor al alumno asociado al usuario representante
Condición Final Exitosa	Se le muestran las notas correspondientes al representado escogido por el representante.

Caso de Uso	4.2 Consultar actividades
Actor	Representante
Descripción	El usuario puede consultar las actividades del año escolar de cada plan de evaluación de las materias cursadas por su representado a fin de conocer su ponderación y fecha en que va a ser realizada
Precondiciones	Se debe haber aprobado el plan de evolución que desea consultar
Condición Final Exitosa	Se muestra la información de las actividades del plan de evaluación seleccionado.

Caso de Uso	5. Administrar Aplicación
Actor	Administrador
Descripción	El usuario puede administrar todos los componentes principales que constituyen la aplicación, como son los usuarios, los cursos, y las materias
Precondiciones	El Usuario administrador debe estar autenticado
Condición Final Exitosa	Se agregan. Editan o eliminan los componentes de la aplicación

Caso de Uso	5.1. Agregar, Eliminar, Editar Usuarios
Actor	Administrador
Descripción	El usuario puede agregar, eliminar o editar cualquier usuario de la aplicación, ya sea estudiante, profesor, de la oficina de control, representante, etc…
Precondiciones	Para los usuarios estudiantes debe haber un usuario representante para poder asociárselo, así como un curso, para el usuario profesor debe existir una materia que se le asocie.

Condición Final Exitosa	Se actualiza en la base de datos y en el sistema la acción realizada, en caso de estar agregándose por primera ves el usuario, se le envía toda la información por correo al nuevo usuario

Caso de Uso	5.2. Agregar, Eliminar, Editar Cursos
Actor	Administrador
Descripción	El usuario puede agregar, eliminar o editar cualquier curso que vaya a existir en la institución durante el año escolar.
Precondiciones	Ninguna
Condición Final Exitosa	Se actualiza en la base de datos y en el sistema la acción realizada

Caso de Uso	5.3 Agregar, Eliminar, Editar Materias
Actor	Administrador
Descripción	El usuario puede agregar, eliminar o editar cualquier materia que se vaya a cursar durante el periodo escolar
Precondiciones	Se deben haber agregado cursos al sistema para que se pueda asociar la materia con el curso donde va a ser dictada
Condición Final Exitosa	Se actualiza en la base de datos y en el sistema la acción realizada

2.3.2. Modelo de Datos

En esta sección se muestra el modelo de datos de la aplicación y las relaciones de entre cada tabla resultado de la identificación de actores y flujo de información de los casos de uso.

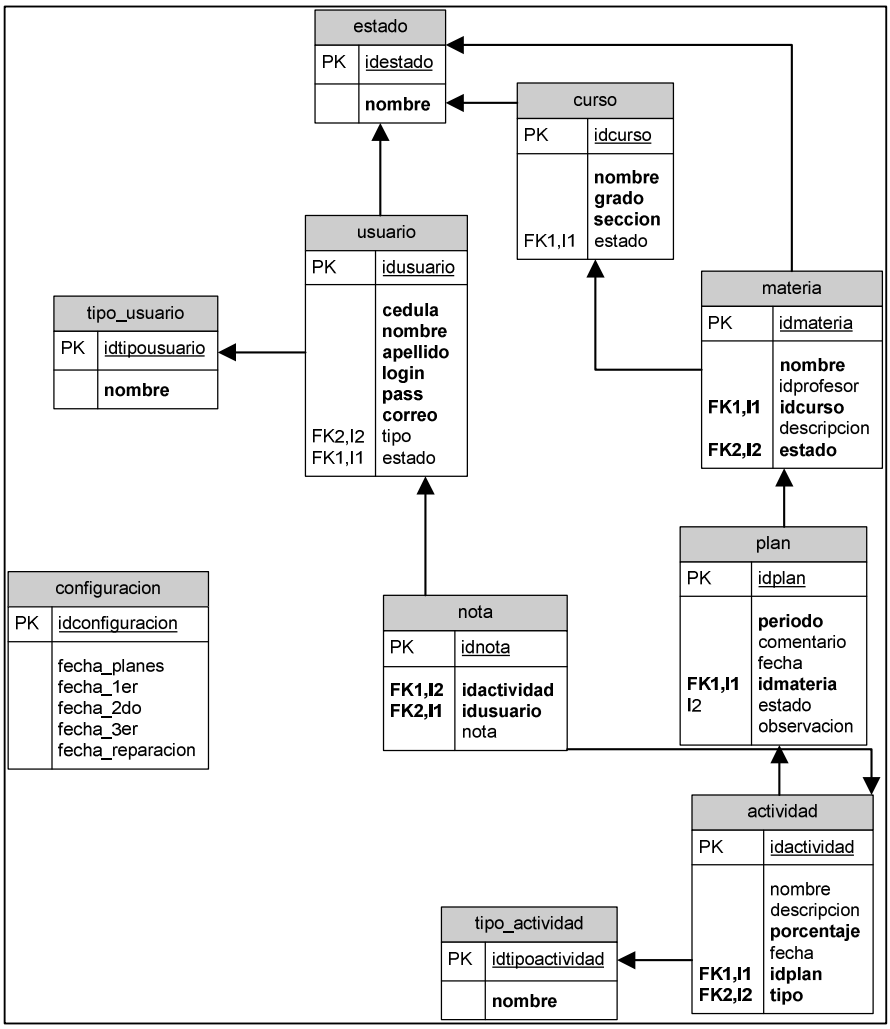


Figura 9 Modelo de Datos del Sistema

En la figura 10 se puede observar las tablas correspondientes a cada uno de los componentes de la aplicación y las cuales van a ser accedidas usando el framework de persistencia hibernate, el cual será explicado mas adelante en el capitulo de implementación.

2.4. Fase de Diseño

Una vez hecho el levantamiento de requerimientos y realizado el análisis correspondiente, que permitió determinar los casos de uso y el flujo de tareas, se logra conocer con mayor profundidad qué necesidades se desean cubrir, qué funcionalidades debe tener la aplicación y la composición de éstas.

A continuación se procede a realizar los diagramas de diseño lógico que permiten especificar cómo se dará solución a los requerimientos encontrados y cómo se realizarán las distintas tareas que conforman las funcionalidades requeridas, con lo cual se obtiene una base fundamental para la implementación de la aplicación.

Antes de proceder a realizar el diseño de la aplicación, se efectuó una investigación comparativa para la selección de un conjunto de frameworks, tanto para el uso del enfoque AJAX, foco principal del trabajo especial de Grado. Así como Para la comprensión de los diagramas de diseño es necesario conocer que, basándose en esta investigación, se decidió hacer uso de los siguientes frameworks:

- Capa de presentación:
    - DWR (http://getahead.ltd.uk/dwr)
    - scriptaculos (http://script.aculo.us/)
    - Prototype (http://www.prototypejs.org)
- Capa de persistencia:
    - Hibernate (http://www.hibernate.org)

El estudio realizado sobre las distintas librerías y frameworks se presenta en la sección 4.1 Comparación de librerías y frameworks, Los diagramas se realizaron utilizando el Lenguaje de Modelado Unificado (UML) en su versión 2.0. Para el diseño se modelan los siguientes diagramas:

- Diagrama de componentes
- Diagrama de despliegue

2.4.1. Diagrama de componentes

Antes de realizar la implementación, es de suma importancia planificar desde un alto nivel las partes que conformarán el sistema, para establecer la arquitectura y las dependencias de éste. El diagrama de componentes UML permite modelar los componentes del sistema, para organizarlo en piezas manejables, re-usables y reemplazables.

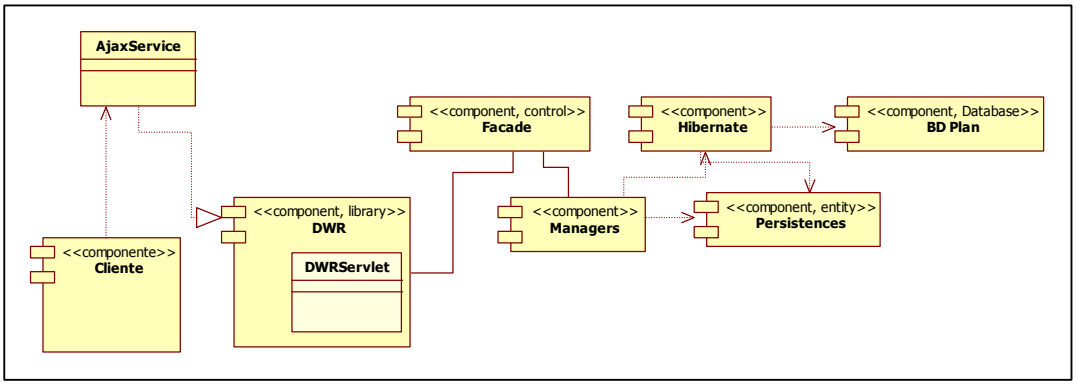


Figura 10 Diagrama de Componentes del Sistema

La Figura 10 muestra el diagrama de componentes elaborado para el desarrollo de la aplicación. En este diagrama se observa el uso de la librería DWR como framework para el uso de AJAX, así como el uso el framework Hibernate para el manejo de persistencia; el estudio de estos frameworks se describe en el Capítulo 4. DWR provee dos clases principales: AJAXService, la cual es una clase JavaScript que encapsula las invocaciones del cliente a métodos remotos, y el servlet DWRServlet, el cual recibe todas las peticiones realizadas a través del framework.

Del lado del servidor se encuentran las todos los componentes divididos usando lo siguientes 3 patrones de diseño de J2EE especificados en http://java.sun.com/blueprints/corej2eepatterns/Patterns/index.html y explicados mas detalladamente en [], a continuación una breve explicación de cómo funcionan y se presentan estos patrones en la arquitectura de nuestro sistema:

- **Front Controller:** Provee un punto de acceso centralizado para la manipulación de peticiones por parte de la capa de presentación, y es implementado por el servlet provisto por DWR. El DWRServlet se encarga de resolver las peticiones invocando los métodos correspondientes del componente que implementa el patrón Facade.
- **Facade:** Provee una capa intermedia (fachada) para los objetos del negocio, separando la lógica de procesamiento de peticiones (Front Controller) de la

lógica del negocio. El componente Facade invoca al objeto del negocio correspondiente con el objeto de realizar la acción requerida (lógica del negocio); estos objetos de negocios implementaran el patrón Business Objects.

- **Business Objects:** La idea es proveer una separación entre la lógica y los datos de negocio, usando un modelo de objetos con una lógica sofisticada, validaciones y reglas de negocio bien definidas. Son simples POJOs (Plain Old Java Objects) que permiten encapsular y manejar los datos de negocio y su comportamiento, con la ayuda de algún mecanismo la persistencia de estos datos. Para este trabajo especial de grado se hace uso de la del framework Hibernate para el manejo de persistencia de los objetos. Los Business Objects hacen uso del EntityManager (provisto por Hibernate) para realizar el manejo de la persistencia de las entidades.

El diagrama de componentes nos brinda una imagen de la aplicación basada en las distintas piezas que la componen; sin embargo, es de relevancia tener una visión del sistema, donde se observen los entornos sobre los cuales se soportan estos componentes. Con este fin se modela el diagrama de despliegue, mostrado en la siguiente sección

2.4.2. Diagrama de despliegue

El diagrama de despliegue que se presenta a continuación (Figura 11), brinda una visión física del sistema, mostrando cómo se encuentra el software con respecto al hardware y como las piezas del sistema se comunican. En este diagrama se muestran tres nodos principales: el cliente, el servidor Web y el servidor de base de datos.

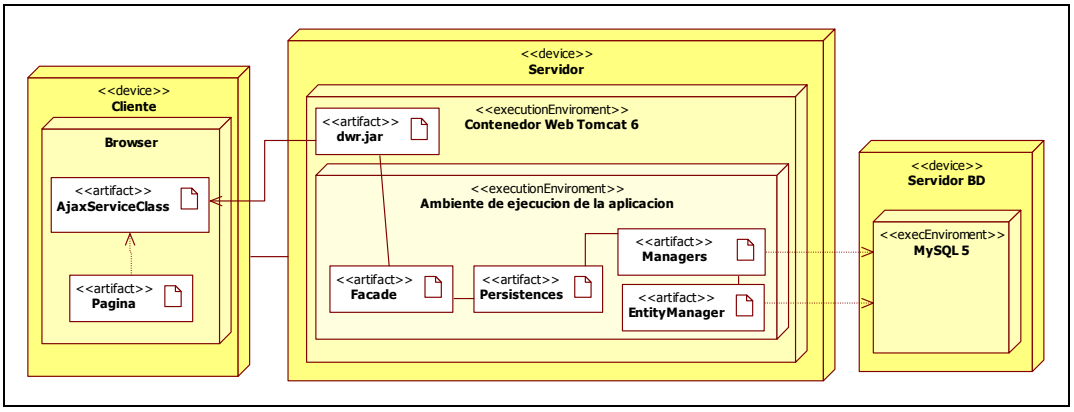


Figura 11 Diagrama de Despliegue del Sistema

Del lado del cliente se presenta el navegador, en el cual residen las páginas de la aplicación y la clase AJAXService, generada por DWR para encapsular las peticiones asíncronas.

Del lado del servidor se utiliza como contenedor Web Apache Tomcat 6.0.16, un contenedor Web de código abierto desarrollado bajo el auspicio de la Apache Software Foundation. En este contenedor reside el framework DWR, las entidades de la aplicación, el componente Facade, los objetos del negocio (Business Objects), y el SessionFactory que provee el framework Hibernate.

En el tercer nodo se ejecuta el manejador de base de datos MySQL 5, en el cual residirá la base de datos de la aplicación. Culminado el diseño de la aplicación y la descripción de sus distintos componentes, se presenta a continuación el capítulo concerniente a la implementación de la solución, donde se muestra el proceso de desarrollo de los distintos componentes y se materializa todo lo presentado en el diseño de la solución.

2.5. Comparación de librerías y frameworks

Para la implementación de la aplicación propuesta se realizó un estudio previo de un conjunto de librerías y frameworks que podrían ser utilizados en el desarrollo de la capa de presentación y la capa de persistencia. Para la capa de presentación se consideraron los frameworks *Tibco, Prototype, DWR y Scriptaculous*.

Tibco (http://www.tibco.com/software/rich_internet_application/default.jsp) permite el desarrollo de aplicaciones Web enriquecidas, aportando un conjunto amplio de componentes gráficos, la posibilidad de realizar peticiones en segundo plano y un entorno de desarrollo integrado (IDE). Este framework centra sus objetivos en el desarrollo de interfaces que contengan componentes llamativos (menús, botones, banner, etc.).

Prototype (http://www.prototypejs.org/) provee métodos o funciones de alto nivel que permiten simplificar la manipulación del DOM y de las peticiones asíncronas. Está basado en el lenguaje JavaScript, el cual es soportado de forma nativa por parte de los navegadores.

DWR (http://getahead.ltd.uk/dwr) permite realizar peticiones asíncronas con objetos Java que residen en el servidor. Está fundamentado en el uso de JavaScript o algún framework basado en este lenguaje del lado del cliente, y objetos Java del lado del servidor. Este framework facilita el desarrollo de aplicaciones Web que utilizan el

lenguaje Java del lado del servidor, permitiendo invocar los métodos pertenecientes a los objetos de manera directa, sin necesidad de agregar algún componente adicional implementado por parte del desarrollador que capture las peticiones y las redireccione a los objetos.

Scriptaculous (http://script.aculo.us/) provee un conjunto de efectos que pueden ser incorporados a los componentes de la interfaz (efectos de rotación, drag & drop, aparecer, desaparecer, etc.); además, este framewok tiene incorporada la librería Prototype, la cual ofrece los beneficios mencionados anteriormente. Scriptaculous está basado en el lenguaje JavaScript.

Luego de estudiar y realizar pruebas con estos frameworks, y sobre la base de que Java será el lenguaje de programación que se utilizará del lado del servidor, consideramos necesario combinar el uso de todos estos frameworks, con la excepción de Tibco.

Tibco centra sus objetivos en el desarrollo de componentes de interfaz llamativos, pero basando su interacción con el servidor principalmente en el uso de Web Services, tecnología que no forma parte de la arquitectura plateada en este trabajo especial de grado; además, el IDE que provee no es de fácil utilización, lo que, por razones prácticas, determinó su descarte como framework AJAX a favor de los mencionados anteriormente.

En la siguiente tabla 8 se hace una comparación de las principales caracateristicas de los frameworks de AJAX estudiados.

Tabla 8 Comparacion entre Frameworks de AJAX

Framework	Libre	Lenguaje de Script	Interoperabilidad	Documentación
Tibco	Si	JavaScript	Cualquier lenguaje	Tutoriales (video)-accesible
Prototype	Si	JavaScript	Cualquier lenguaje	Extensa-accesible
DWR	Si	JavaScript	Java	Extensa-dispersa
Scriptaculous	Si	JavaScript	Cualquier lenguaje	Poca

Para la capa de persistencia se consideraron los frameworks de persistencia Hibernate, TopLink Essentials y Glassfish Persistente.

TopLink Essentials (http://www.oracle.com) es una versión de código abierto de la herramienta de manejo de persistencia de Oracle, TopLink. Esta versión provee una implementación de la API de Persistencia Java y funciona con cualquier base de datos o servidor de aplicaciones. TopLink Essentials fue creado por una compañía reconocida que provee un buen soporte, Oracle.

A pesar de todas las ventajas que provee TopLink Essentials, esta herramienta fue descartada por el riesgo que puede presentar el uso de una versión limitada perteneciente a una compañía dedicada a desarrollar principalmente sistemas propietarios.

Glassfish Persistence (https://glassfish.dev.java.net) es una implementación de la API de Persistencia Java, desarrollada por java.net bajo el proyecto de Glassfish. La implementación Glassfish Persistente funciona en ambientes stand-alone y en servidores de aplicaciones diferentes a Glassfish, es de uso libre y su documentación está soportada por la comunidad de java.net. Sin embargo, en un principio la implementación de Glassfish Persistente se desarrolló para su uso en el servidor de aplicaciones Glassfish, y está basada en la implementación libre de Oracle TopLink Essential; estos factores condujeron al uso de otra herramienta para el manejo de persistencia.

Hibernate (http://www.hibernate.org) es un framework de persistencia objeto/relacional y servicios de consultas, el cual permite desarrollar clases persistentes basadas en el paradigma orientado a objetos, incluyendo el manejo de la herencia, asociaciones, polimorfismo, composición y colecciones. Hibernate provee una implementación que funciona tanto en ambientes stand-alone como en diferentes contenedores Web y servidores de aplicaciones. Este framework es de código abierto, permite la portabilidad de código entre distintos manejadores de base de datos, y provee una extensa documentación organizada y de fácil acceso. Además de estas ventajas existen un conjunto de herramientas que facilitan el uso de este framework, así como su integración con otros IDEs o frameworks.

Fundamentándose en este estudio se tomó la decisión de utilizar Hibernate como framework para el manejo de persistencia en el trabajo que se desarrollará, por sus importantes cualidades, tales como: ser software libre, que sea una version estable del framework, no debe ser una versión limitada de una aplicación comercial, y por poseer una amplia documentación.

Para resumir esta comparación, a continuación en la tabla 9 se muestra un cuadro que presenta las principales características de los frameworks y herramientas estudiadas para el manejo de la capa de persistencia de datos.

Tabla 9 Comparacion entre Frameworks de Persistencia

Producto	**Libre**	**Limitado**	**Pertenece a**	**Documentación**
TopLink Essential	Si	Si	Oracle	Tutoriales-accesible
Glassfish	Si	No	Java.net	Extensa-dispersa
Hibernate	Si	No	JBoss	Extensa-accesible

Capítulo 3. Implementación

Con base en el análisis y diseño presentado en el capítulo 2 y siguiendo con la metodología propuesta, en este capítulo se presenta la implementación de la solución para los requerimientos planteados.

El capítulo está dividido en 2 secciones que presentan tanto la implementación de la lógica del lado del cliente como la implementación de la lógica del lado del servidor.

La sección 3.1 presenta la plataforma de hardware y software utilizada para el desarrollo. En la sección 3.2 se describe la implementación concerniente al lado del cliente, describiendo los artefactos de software creados en las diferentes iteraciones.

3.1. Plataforma de Hardware y Software

En esta sección se describe la plataforma de hardware y software utilizada durante la implementación de la solución de este trabajo especial de grado.

3.1.1. Plataforma de Hardware

Para el desarrollo de la aplicación se utilizaron dos equipos con la siguiente configuración:

- Procesador Intel Pentium IV 3.2 GHZ
- 1 GB de memoria RAM
- Espacio en disco utilizado para la aplicación y software de desarrollo y ejecución: 1 GB

3.1.2. Plataforma de software

A continuación se describen todos los componentes que conforman la plataforma de software utilizada para el desarrollo de la aplicación de Gestión Escolar.

Sistema Operativo

Para el desarrollo de la aplicación se utilizó sobre Windows XP Professional; sin embargo, se han realizado varias pruebas sobre la distribución Ubuntu, de Linux. La aplicación puede ejecutar sobre cualquier sistema operativo que soporte la maquina virtual Java versión 5 y el resto de los componentes, tales como el SMBDR[3], MySQL 5 y el contenedor Web Tomcat 6.0.

Máquina virtual Java

La aplicación fue desarrollada utilizando el lenguaje de programación Java; por lo tanto, es necesario tener una JVM que soporte la aplicación. La versión a utilizar de la JVM debe ser de la 5 en adelante, ya que las anotaciones (utilizadas por el framework DWR y Hibernate) son soportadas sólo a partir de Java 5. Además de las anotaciones se aprovecharon ciertas bondades de Java 5 como los cambios en la API de Collections, utilizando colecciones restringidas, y la forma de iterar sobre las colecciones. Para el desarrollo se utilizo el JDK (Java Development Kit) 1.5.0 revisión 15.

Contenedor Web

Una pieza fundamental para la ejecución de la aplicación es el contenedor Web, que brindará todos los servicios sobre los que se apoyará la aplicación para su acceso a través de la Web.

Como contenedor Web se utilizó Apache Tomcat en su versión 6.0.16.

Sistema manejador de base de datos

Como manejador de base de datos se escogió MySQL 5, ya que es un manejador Open Source muy utilizado que cuenta con una amplia documentación, además de su eficiencia y rapidez.

3.2. Implementación del lado del cliente

Esta sección describe los componentes utilizados para la implementación de la lógica del lado del cliente. Para esta implementación se utilizaron los frameworks DWR y Prototype, así como también la librería scriptaculous. La sección comienza describiendo cómo se realizó la instalación y qué utilidad se dio a cada uno de estos frameworks; seguidamente, se describen las páginas (html) que se desarrollaron, y qué papel

[3] Siglas para Sistema Manejador de Bases de Datos Relacionales

juegan en la aplicación.

Por último, se presentan las funciones JavaScript más importantes desarrolladas en la solución.

3.2.1. Prototype

Por medio de este framework se realizaron las modificaciones dinámicas sobre el modelo de objetos del documento de las páginas de la aplicación.

La versión utilizada fue la 1.6

Instalación

Este framework se encuentra disponible en (http://www.prototypejs.org). Para incorporarlo como parte de la aplicación, se debe escribir dentro de la etiqueta <HEAD>, en las páginas donde se desee utilizar el framework, la siguiente línea:

```
<script type='text/javascript' src='[ruta]/prototype.js'></script>
```

Ya que este framework está basado en JavaScript, su incorporación en la aplicación es sencilla. La ruta debe seguir la jerarquía de carpetas donde se encuentre almacenado el archivo prototype.js; en nuestro caso, esta ruta corresponde a PlanEva/javascript (ruta absoluta) o javascript (ruta relativa).

Uso dentro de la aplicación

La documentación disponible en (http://www.prototypejs.org) es muy completa e incluye ejemplos bastante descriptivos, por lo que no se presentaron problemas infranqueables con respecto al desarrollo basado en este framework. Entre algunos de los métodos más utilizadas podemos destacar:

- **`$('<id del elemento>')`:** Permite obtener la referencia a un elemento HTML a partir de su identificador.
- **`update()`:** Por medio de este método es posible actualizar el contenido interno de cualquier elemento HTML de la página en forma dinámica. Es necesario utilizar el método descrito anteriormente, `$()`, para obtener la referencia al elemento, y posteriormente actualizarlo, de la siguiente forma:

```
$('<id del elemento>').update().
```

- **`Insertion`:** El objeto `Insertion` permite agregar elementos HTML de forma dinámica. La sintaxis para su uso es:

```
new Insertion
```

  Es necesario especificar el orden en que se agregara el elemento, es decir:

  - **`Insertion.After(`**'**`<id>`**', '**`<HTML>`**'**`):`** Inserta un nuevo elemento después del elemento identificado por '`<id>`'; el nuevo elemento se pasa como segundo parámetro.
  - **`Insertion.Before(`**'**`<id>`**', '**`<HTML>`**'**`):`** Inserta un nuevo elemento antes del elemento identificado por '**`<id>`**'; el nuevo elemento se pasa como segundo parámetro.
  - **`Insertion.Bottom(`**'**`<id>`**', '**`<HTML>`**'**`):`** Inserta un nuevo elemento al final de los elementos que se encuentren al mismo nivel dentro del elemento identificado por '**`<id>`**'. Utilizado cuando se desean agregar elementos a una tabla. El nuevo elemento se pasa como segundo parámetro.
  - **`Insertion.Top(`**'**`<id>`**', '**`<HTML>`**'**`):`** Inserta un nuevo elemento al principio de los elementos que se encuentren al mismo nivel dentro del elemento identificado por '**`<id>`**'. Utilizado cuando se desean agregar elementos a una tabla. El nuevo elemento se pasa como segundo parámetro

- **`hide()`:** Oculta un elemento HTML. Es necesario utilizar el método **`$()`** para obtener la referencia al elemento y posteriormente ocultarlo, de la siguiente manera:

```
$('<id elemento>').hide().
```

- **`show()`:** Hace visible un elemento HTML. Es necesario utilizar el método **`$()`** para obtener la referencia al elemento y posteriormente hacerlo visible, de la siguiente manera:

```
$('<id elemento>').show().
```

3.2.2. Scriptaculous

Esta librería se soporta en Prototype, y permitió incorporar efectos en la interfaz de usuario, como el efecto de aparecer, desaparecer, acordeón, etc. a algunos elementos de la interfaz. La versión utilizada fue la 1.8.1.

Instalación

Esta librería se encuentra disponible en (http://script.aculo.us/). Para incorporarla como parte de la aplicación es necesario escribir dentro de la etiqueta <HEAD>, en las páginas donde se desee utilizar el framework, la siguiente línea:

```
<script type='text/javascript' src='[ruta]/scriptaculous.js'></script>
```

Es importante destacar que esta librería hace uso del framework Prototype, es decir, es necesario incorporar este último según se explicó en la sección anterior.

Ya que esta librería está basada en JavaScript, su incorporación en la aplicación es sencilla. La ruta debe seguir la jerarquía de directorios donde se encuentre almacenado el archivo `scriptaculous.js`, que en nuestro caso corresponde a PlanEva/javascript (ruta absoluta) o javascript (ruta relativa).

Uso dentro de la Aplicacion

La documentación disponible en (http://script.aculo.us/) es muy limitada y los ejemplos que incluye están desarrollados en el lenguaje Ruby; sin embargo, los problemas que se presentaron fueron solventados. El uso de esta librería se presento mas que todo para darle efectos de transición, aparición y desaparición a elementos HTML en la interfaz, entre los más utilizados podemos destacar:

- `Effect.Highlight`: Este efecto hace énfasis en el fondo de un elemento html cambiandole de color repetidamente para así obtener un efecto de brillo, y se utiliza de la siguiente manera:

  ```
  new Effect.Highlight('id_of_element', [options]);
  ```

- `Effect.Appear`: hace que un elemento perteneciente a la interfaz y que previamente haya sido escondido (usando la propiedad `display:none` o `$('<id elemento>').hide()`) aparezca en la interfaz de forma transitoria y se

utiliza de la siguiente forma.

```
Effect.Appear('id_of_element');
 O $('id_of_element').appear();
```

- `Effect.Fade:` desaparece de forma transitoria un elemento html que se le indique en la interfaz y que para el momento de aplicarle el efecto se encuentre visible y se utiliza de la siguiente forma.

```
Effect.Fade('id_of_element');
                O
  $('id_of_element').fade();
```

- `Effect.BlindDown:` Le proporciona un efecto de deslizamiento tipo ventana hacia abajo a cualquier elemento html que se le indique manteniendo todos los elementos internos en su posición, se utiliza de la siguiente forma

```
Effect.BlindDown('id_of_element');
```

- **`Sortable.create:`** Inicializa un elemento html como Sortable, es decir, que el contendido presente en este puede ser ordenado automáticamente. Se utiliza de la siguiente manera

```
Sortable.create('id_of_container',[options]);
```

3.2.3. DWR (Direct Web Remoting)

La incorporación de este framework fue de gran importancia para el desarrollo de la aplicación, ya que por medio de este fue posible realizar peticiones en segundo plano, y además, permite invocar métodos desarrollados en el lenguaje de programación Java de forma transparente, es decir, no es necesario que las clases desarrolladas en Java hereden de alguna clase especial como por ejemplo HTTPServlet, ya que este framework incorpora un Servlet denominado DWRServlet, el cual se encarga de mapear las llamadas a los métodos desarrollados en Java con la clase que implementa este método.

Instalación

Los archivos para la instalación de DWR se encuentra disponible en (http://getahead.org/dwr/), parte de la configuración necesaria para hacer uso de este framework se explica a continuación. En primer lugar se debe agregar el archivo dwr.jar al directorio WEB-INF/lib de la aplicación. Posteriormente es necesario editar el archivo Web.xml como se muestra a continuación:

```
<servlet>
 <servlet-name>dwr</servlet-name>
<servlet-class>
 org.directwebremoting.spring.DwrSpringServlet
</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>dwr</servlet-name>
 <url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

El extracto de código anterior muestra la configuración necesaria para activar el ciclo de vida del servlet DwrSpringServlet, el cual se encargara de recibir las peticiones realizadas por el cliente e invocar el método correspondiente para dar respuesta a dicha petición.

Posteriormente es necesario indicar que clase o cuales clases podrán ser invocadas de forma remota, es decir, cuales clases desarrolladas en el lenguaje de programación Java serán transformadas a código JavaScript para que sean incluidas en las paginas de la aplicación para realizar las peticiones de forma asíncrona. A continuación se muestra como se deben incluir los archivos JavaScript generados por DWR y dos archivos JavaScript propios de DWR que contienen algunos métodos que fueron utilizados para el desarrollo de la aplicación dentro de la etiqueta <HEAD> de las páginas que harán uso del framework:

```
    <script type='text/javascript' src='dwr/engine.js'></script>
     <script type='text/javascript' src='dwr/util.js'></script>
  <script type='text/javascript' src='dwr/interface/[nombre].js'></script>
```

La tercera línea del extracto de código anterior muestra la forma en que deben ser incluidos los archivos JavaScript generados por DWR, donde [nombre] corresponde al nombre de la clase que ha sido configurada como remote.

Más adelante en la sección 4.4 se explica como configurar las clases desarrolladas en Java para que estas sean remote y se genere el código JavaScript correspondiente a estas.

Uso en la Aplicación

La documentación disponible en (http://getahead.org/dwr/) es muy completa e incluye ejemplos, no se presentaron problemas con respecto al desarrollo basado en este framework, entre algunos de los métodos más utilizadas podemos destacar:

- **DWRUtil.setValue**: Permite asignar un valor a un elemento HTML especificando su **id**, su sintaxis es la siguiente:

```
DWRUtil.setValue('<id del elemento HTML>', '<valor>')
```

- **DWRUtil.setValues**: Permite asignar distintos valores a distintos elementos HTML a partir de un objeto, su sintaxis es la siguiente:

```
DWRUtil.setValues(objeto)
```

  El nombre de cada atributo del objeto que se desee mostrar debe corresponderse con el **id** del elemento que se desee que muestre el valor de dicho atributo.

- **DWRUtil.addOptions**: Permite agregar opciones a un elemento HTML **<SELECT>, <OL> o <UL>** a partir de un arreglo, su sintaxis es la siguiente:

```
DWRUtil.addOptions('id del elemento HTML', arreglo)
```

- **DWRUtil.removeAllOptions**: Permite eliminar todas las opciones de un elemento HTML **<SELECT>, <OL> o <UL>** especificando su **id**, su sintaxis es la siguiente:

```
DWRUtil. removeAllOptions('<id del elemento HTML>')
```

- **DWRUtil.cloneNode**: Permite clonar o crear una copia de un elemento HTML especificando su **id** y un prefijo que será utilizado para asignar un id al nuevo elemento HTML creado, su sintaxis es la siguiente:

```
DWRUtil.cloneNode('<id del elemento HTML>',{idSuffix:sufijo})
```

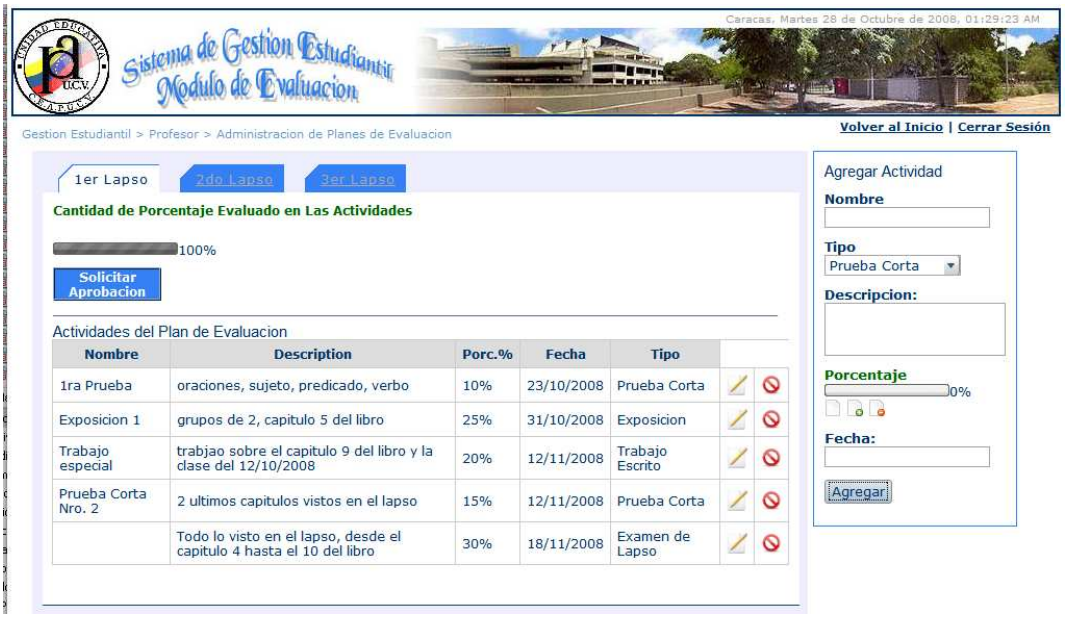
A continuación en el siguiente punto se muestran los artefactos creados en cada iteración del desarrollo de la aplicación.

3.2.4. Páginas HTML

Gracias al enfoque de desarrollo AJAX una página puede manejar diferentes vistas, dada su capacidad para modificarse dinámicamente y hacer peticiones asíncronas. Este caso se presenta en la solución implementada.

A continuación se describen las páginas HTML desarrolladas en cada una de las iteraciones y que presentan las diferentes secciones de la aplicación.

Artefacto: proplaneva.htm 1ra. Iteración, Caso de Uso 1
En esta página se desarrollaron todos los elementos de interfaz de la sección que le permite al usuario de tipo profesor crear los planes de evaluación de cada lapso así como agregar, modificar o eliminar las actividades correspondientes a cada plan de evaluación. Además de código HTML este artefacto tiene asociado un archivo con código JavaScript, utilizado para realizar las peticiones asíncronas y para modificar la vista cuando se ejecute la funcionalidad de gestión de planes de evaluación. Como se mencionó al comienzo de la sección gracias al uso del enfoque AJAX esta misma página puede manejar distintos modos de vista. En cualquiera de las vistas de crear hoja el usuario puede guardar las actividades, solicitar la aprobación del plan de evaluación o volver a la página principal.



Artefacto: pronotas.htm 2da. Iteración, Caso de Uso 2

La página `pronotas` presenta la vista para la carga y modificación de notas de cada actividad realizada en un lapso, la pagina muestra en el panel del lado izquierdo una estructura en forma de árbol en la que debe escoger el lapso y la actividad que desea cargar las notas, en caso de que el plan de evaluación no haya sido aprobado todavía o no exista, no le mostrara ninguna actividad al usuario.

Artefacto: conplaneva.htm 3ra. Iteración, Caso de Uso 3

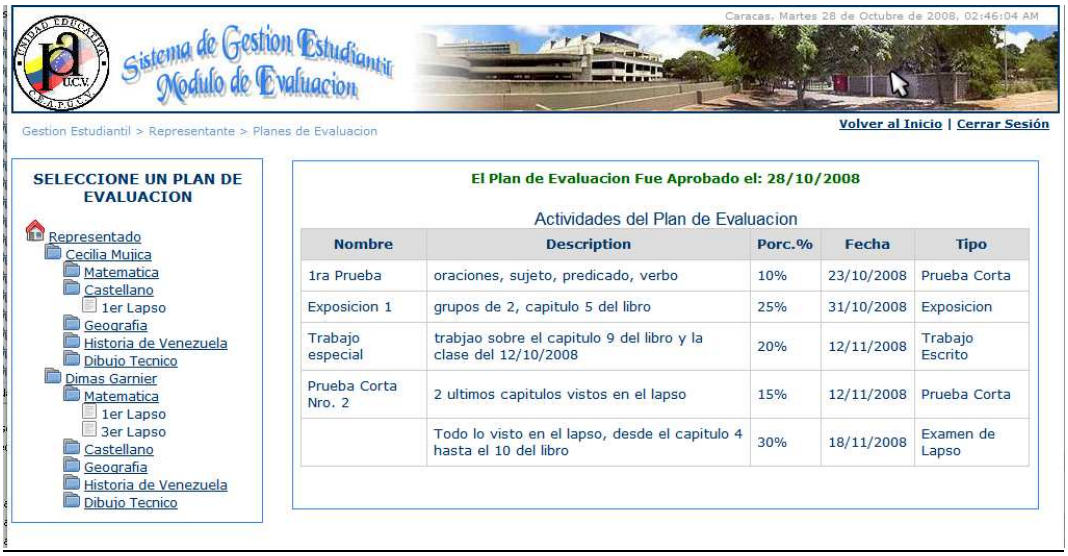
En la página `conplaneva` los usuarios del tipo Oficina de Control de evaluación, pueden consultar, aprobar o rechazar cualquier plan de evaluación que haya sido previamente enviado a su aprobación, por parte de algún usuario de tipo profesor. En el panel de la izquierda, el usuario puede navegar por una estructura de tipo árbol para acceder a todos los planes de evaluación, de todas las materias, de todos los cursos de la institución.

Artefacto: connotas.htm 4ta. Iteración

En la página conplaneva los usuarios del tipo Oficina de Control de evaluación, pueden consultar, todas las notas que han sido guardadas por los usuarios de tipo profesor. Así mismo pueden navegar por la estructura de árbol para acceder a cualquier actividad que haya sido evaluada y/o que pertenezca a un plan de evaluación que ya haya sido aprobado por la Oficina de Control y Evaluación.

Artefacto: repplaneva.htm, 5ta. Iteración, Caso de Uso 4

En la página `repplaneva` los usuarios del tipo representante, pueden consultar, todas las actividades correspondientes a cada una de las materias de los alumnos representados por ellos, con el fin de que los representantes estén al tanto de cuando se realizan cada una de las actividades.



Artefacto: repnotas.htm 6ta. Iteración, Caso de Uso 4

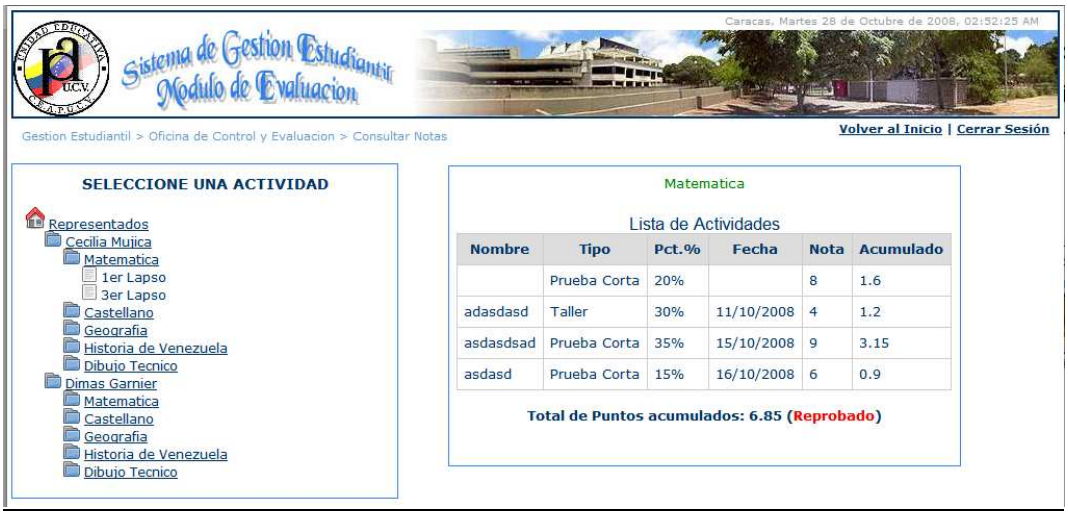
En la página `repnotas` los usuarios del tipo representante, pueden consultar, todas las notas correspondientes a cada una de las actividades, de los planes de evaluación de las materias vistas por sus representados. Al igual que en las otras paginas el usuario debe navegar por la estructura de árbol del panel de la izquierda para escoger el plan de cual representado y materia desea consultar la nota, así como el acumulado de puntos que lleva hasta el momento. En caso de que ya haya sido evaluado el 100% de la puntuación correspondiente al lapso el sistema le indica si el alumno fue aprobado o reprobado en ese periodo.

Artefacto: repnotas.htm 7ma. Iteración

En la página `repnotas` los usuarios del tipo representante, pueden consultar, todas las notas correspondientes a cada una de las actividades, de los planes de evaluación de las materias vistas por sus representados. Al igual que en las otras paginas el usuario debe navegar por la estructura de árbol del panel de la izquierda para escoger el plan de cual representado y materia desea consultar la nota, así como el acumulado de puntos que lleva hasta el momento. En caso de que ya haya sido evaluado el 100% de la puntuación correspondiente al lapso el sistema le indica si el alumno fue aprobado o reprobado en ese periodo.

Artefacto: adminis.htm 8va. Iteración, Caso de Uso 5

En la página `adminis` el usuario de tipo administrador, puede agregar, eliminar o modificar, los 3 componentes principales de la aplicación que son los usuarios (ya sea representante, alumno, profesor, Oficina de control y evaluación, etc.). los cursos que van a ser dictados durante el periodo escolar y las materias



3.3. Análisis Interactivo de la aplicación

En este punto se le realizara un estudio a la interfaz de usuario de la paliación desarrollada con el fin de destacar los puntos que hacen de esta aplicación a nivel interactivo una RIA, así como también hacer énfasis en que punto de la aplicación se hicieron las llamadas asíncronas por medio del motor AJAX y el uso de elementos enriquecedores de la interfaz

Estructura de la aplicación

Esta aplicación fue desarrollada con una profundidad de tres páginas distribuidas de la siguiente forma, la pagina de inicio (figura 12), una pagina menú principal la cual se despliega dependiendo del tipo de usuario y las páginas donde se ejecutan cada una de las actividades que desean realizar los distintos usuarios y son las que contienen el mayor uso de elementos AJAX para la comunicación asíncrona con el servidor.

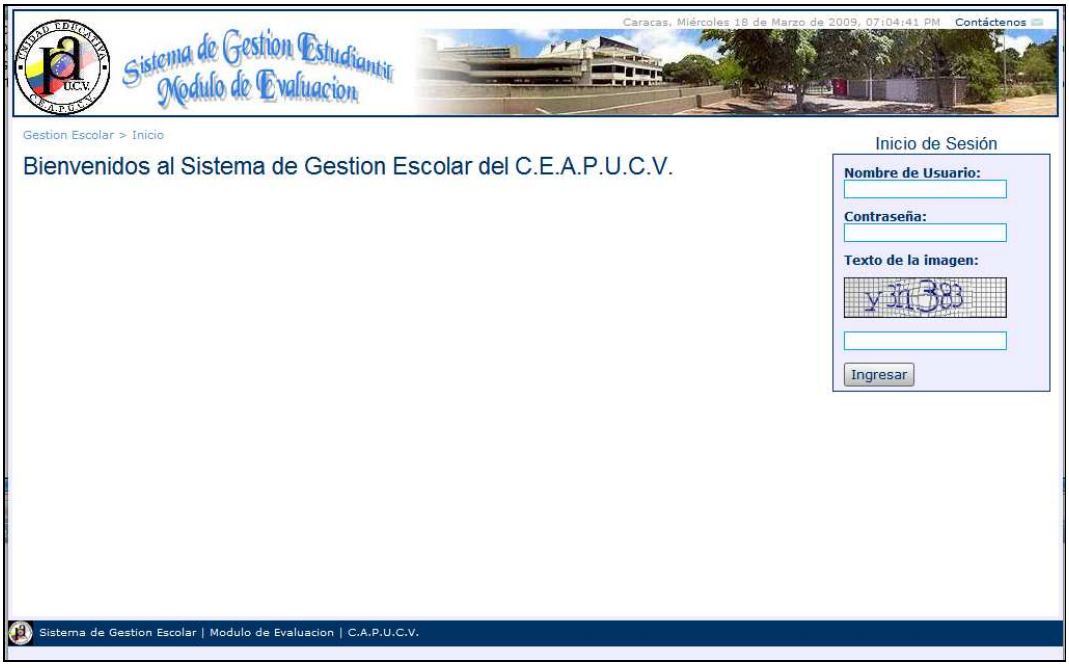


Figura 12 Página de Inicio (inicio.htm)

Patrones de Interacción Utilizados

Básicamente se utilizaron dos patrones de interacción los cuales cumplen con la finalidad de mantener informado al usuario lo que se puede hacer en cada parte de la aplicación (figura 13) y un menú despleglable en las funciones donde hace falta hacer búsquedas y/o consultas de información con el fin de facilitarle la obtención de esta al usuario y que información esta disponible al momento (figura 14).

Figura 13 Menú principal

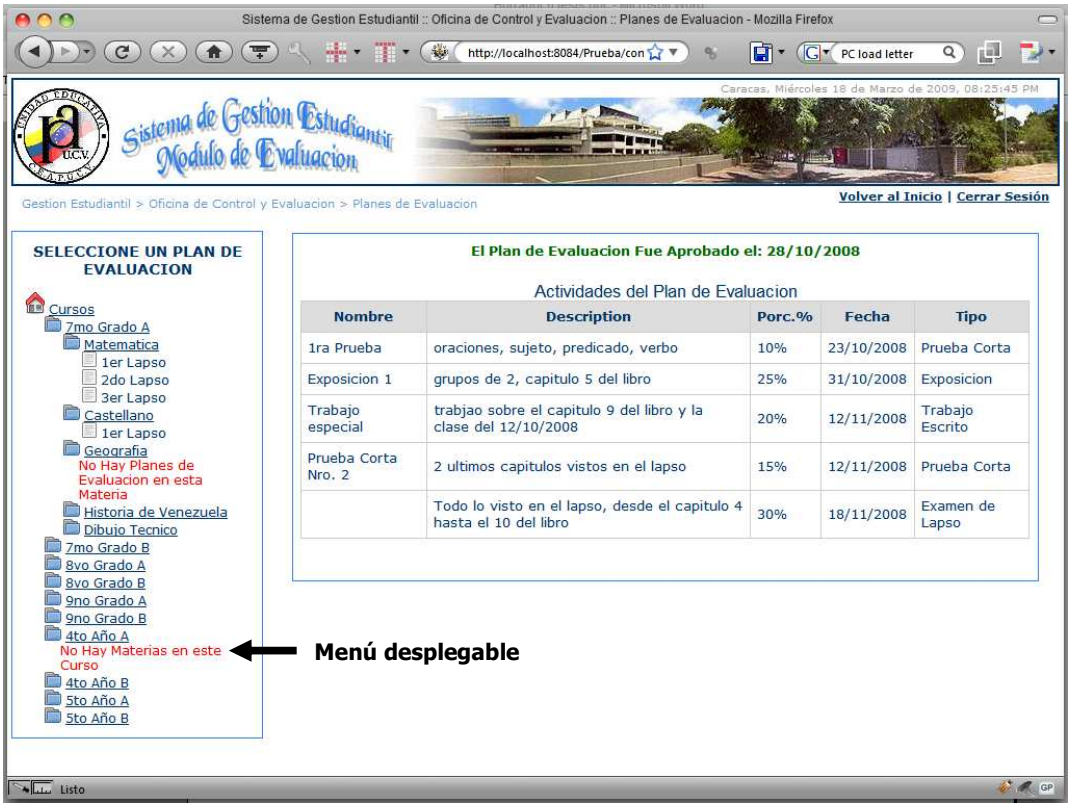


Figura 14 Menú desplegable de contenidos

Otro elemento de interacción destacable fue el uso de pestañas (figura 15) en la función de creación y revisión de planes de evaluación por parte de los usuarios de tipo profesor, el cual ayudo a mantener en una misma pagina toda la información de los planes de evaluación referentes a los tres lapsos correspondientes al año escolar, logrando así que la interacción con la aplicación sea fluida. Este menú de pestañas se logro mediante la combinación de funciones Javascript y Hojas de Estilos en Cascada (CSS) elementos fundamentales para el desarrollo AJAX.

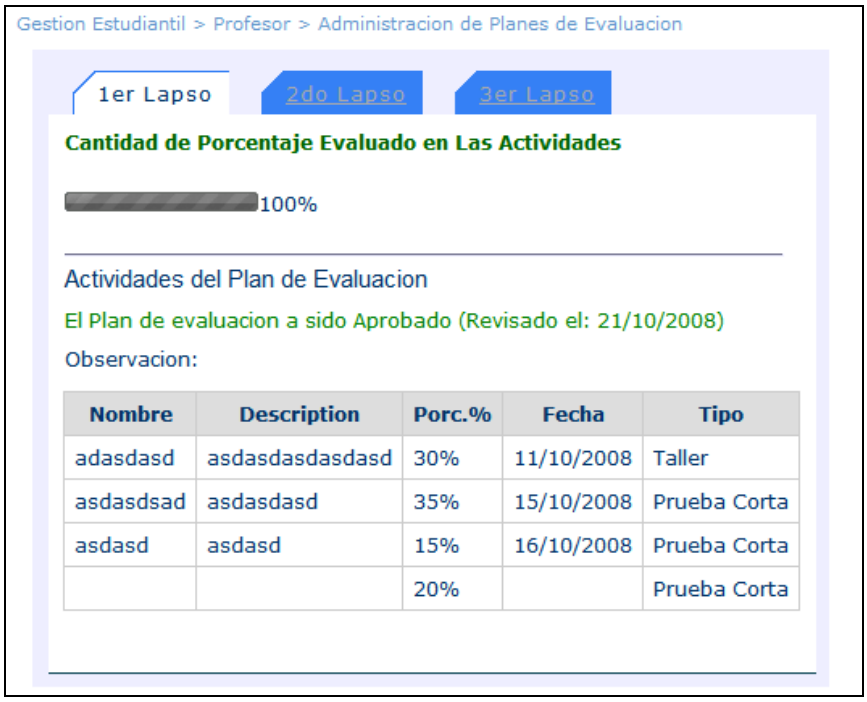


Figura 15 Pestañas

Formularios

El comportamiento de los formularios y campos para ingresar información corresponden uno de los contrastes mas significativos de la aplicación ya que al estos tener un comportamiento basado en AJAX, es decir, enviando información de forma asíncrona al servidor, ya que todos los cambios ocurren en la misma pagina que se encuentra el usuario en el momento de ingresar la información. Otra característica importante presente en los formularios de esta aplicación es el uso de herramientas Javascript y CSS para hacer las notificaciones al usuario en caso de que no se cumpla algún criterio durante las validaciones de campos, estas notificaciones se realizan directo en la interfaz de la pagina y se indica campo por campo en que fallo, como se muestra en la figura 16.

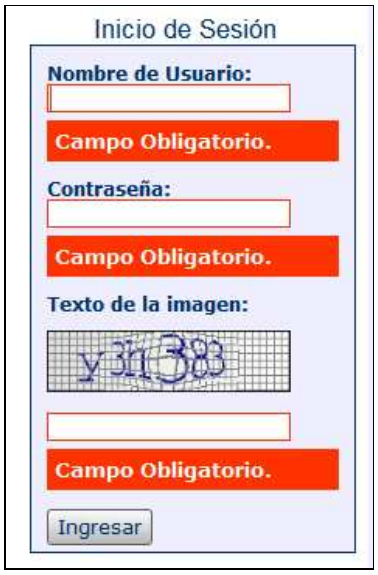


Figura 16 Notificación de errores durante la validación

Además en cada formulario se agregaron elementos que facilitaran a su ves el ingreso de información por parte del usuario, y así hacer que sea mas rápida y robusta la información contenida en los campos a ser enviados, como se muestra por ejemplo en la figura 17 donde se puede observar una barra de progreso para llevar el porcentaje de cada actividad y un calendario desplegable para ingresar la fecha en que va a ocurrir una actividad.

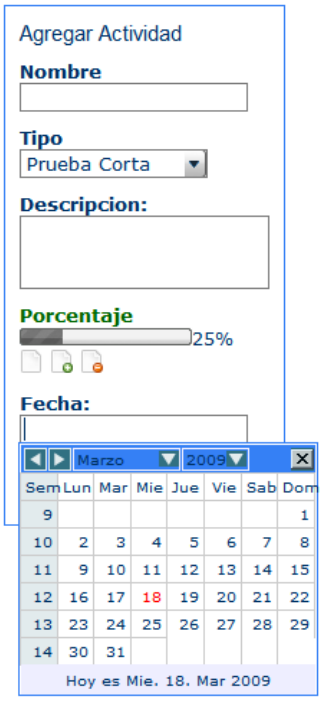
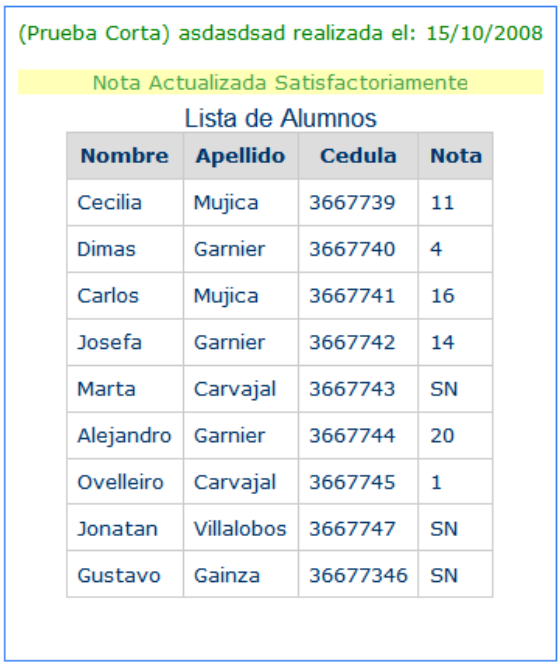


Figura 17 Elementos para ingresar datos

Notificación de cambios o actualizaciones

Al cambiar el esquema de interacción de una aplicación Web tradicional por el esquema AJAX se debe tomar en cuenta que probablemente el usuario no se de cuenta, por ejemplo, de que la información fue ingresada correctamente, ocurrió un cambio en la aplicación, o por el contrario un error mientras se enviaba algún dato o se realizaba una consulta. Por lo que se debe siempre mantener alertas en la interfaz (ej: cambios en esta o mensajes de alerta) como se ve en la figura 18, elementos que le indiquen al usuario que se puede realizar cierta actividad como el uso de ToolTips como se muestra en la figura 19, o indicadores de espera como se muestra en la figura 20


(Prueba Corta) asdasdsad realizada el: 15/10/2008

Nota Actualizada Satisfactoriamente

Lista de Alumnos

Nombre	Apellido	Cedula	Nota
Cecilia	Mujica	3667739	11
Dimas	Garnier	3667740	4
Carlos	Mujica	3667741	16
Josefa	Garnier	3667742	14
Marta	Carvajal	3667743	SN
Alejandro	Garnier	3667744	20
Ovelleiro	Carvajal	3667745	1
Jonatan	Villalobos	3667747	SN
Gustavo	Gainza	36677346	SN

Figura 18 Cambios en la Interfaz

En la figura 18 se muestra como aparece en la interfaz un mensaje indicando que se actualizaron satisfactoriamente las notas al hacer un cambio sobre estas.

Figura 19 ToolTip de notificación

En la figura 19 se muestra como aparece en la interfaz un ToolTip al colocar encima el puntero del ratón indicándole al usuario que se puede hacer cierta acción al hacer clic sobre ese elemento de la interfaz.

Figura 20 Indicadores de Espera

En la figura 20 se muestra un elemento de la interfaz que se encuentra bloqueado y que a su ves muestra un icono que le indica al usuario que debe esperar para continuar.

Comunicación Asíncrona con el Servidor y actualización de la interfaz

Este punto es el que hace que la interacción con la aplicación desarrollada sea mas fluida que con una aplicación Web tradicional ya que toda información que se solicite se muestra en una misma pagina haciendo cambios en la interfaz de la misma por medio del motor AJAX explicado en los capítulos anteriores. Por ejemplo en la figura 21 se muestra un menú desplegable a la izquierda en el cual al hacer clic sobre alguno de los indicadores de lapsos se hace una solicitud al servidor sobre la información del plan de evaluación del lapso solicitado.

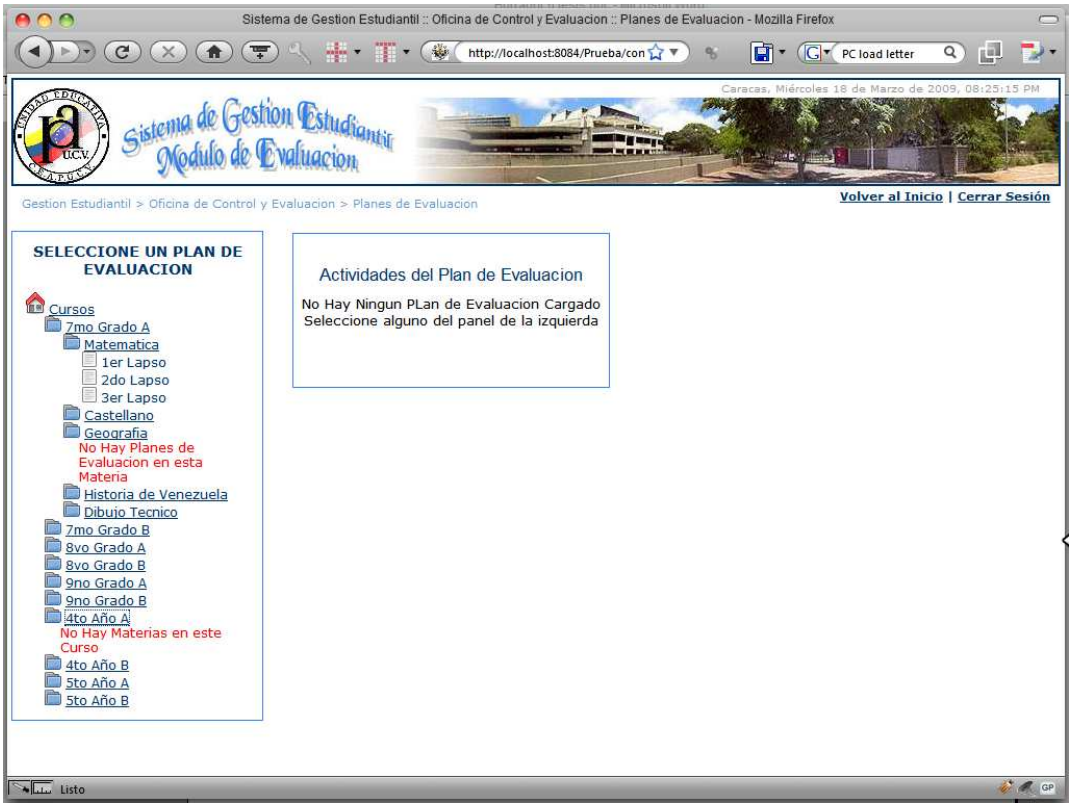


Figura 21

Cuando se obtienen esos datos se hace un cambio en la interfaz y se muestran en la tabla presente en el medio de la pagina utilizando un efecto de deslizamiento para llamar la atención del usuario y así note que se esta mostrando la información solicitada por el como se muestra en la figura 22.

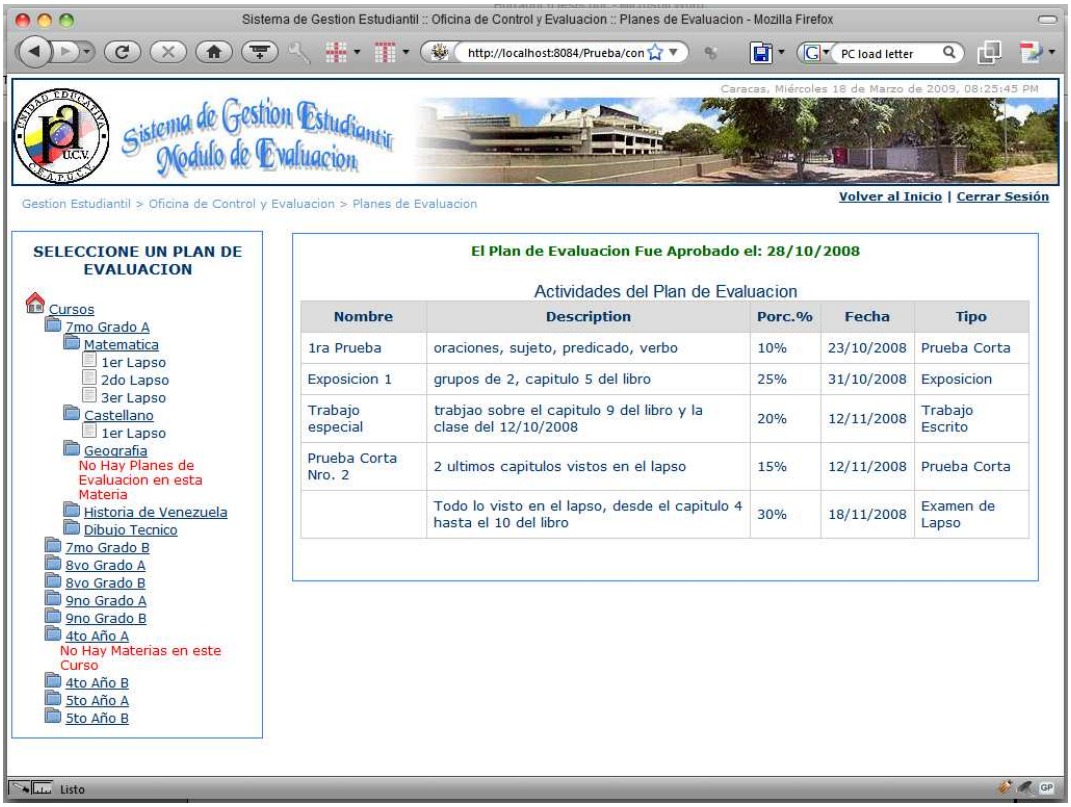


Figura 22

En la figura 23 y 24 se muestra otro ejemplo de este comportamiento al solicitar las notas de las actividades de una lapso se cambian en la misma pagina por las del nuevo lapso solicitado manteniendo así toda la información en una misma pagina lo que en una aplicación Web tradicional hubiese requerido de muchas recargas de pagina para mostrar la información obtenida en cada una de las peticiones realizadas.

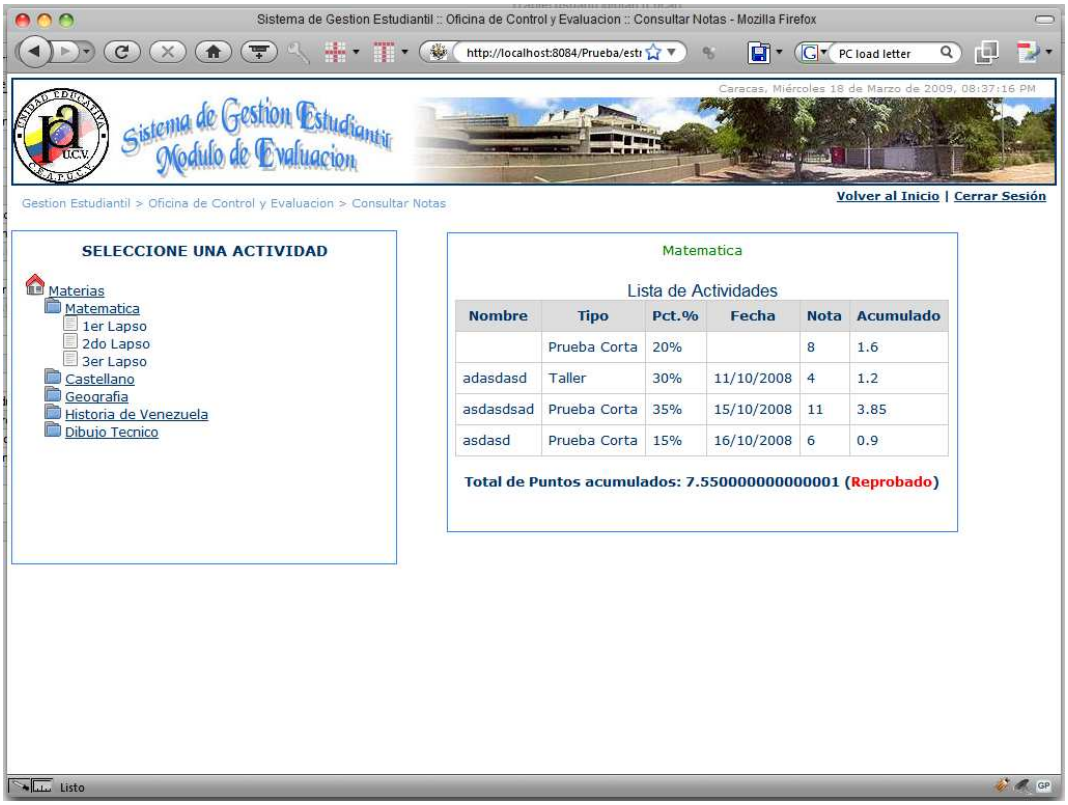


Figura 23 Cambios en la Interfaz

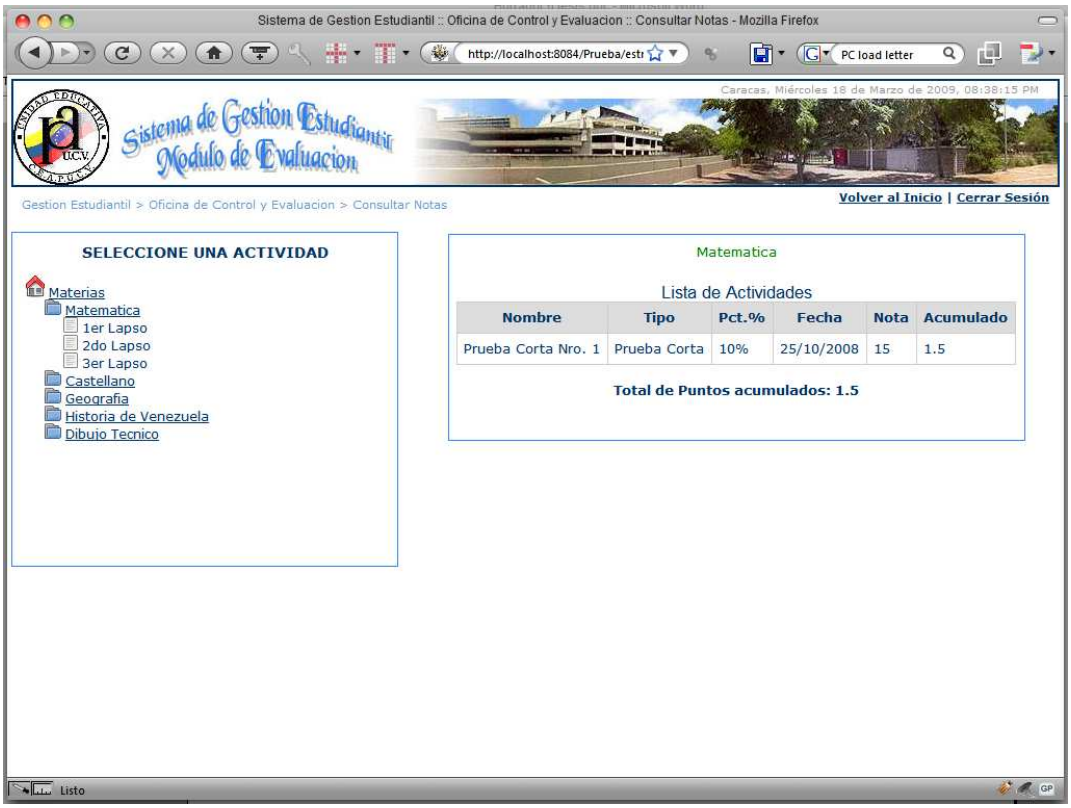


Figura 24 Cambios en la Interfaz

Pruebas

Debido a la gran variedad de navegadores existentes en la actualidad durante cada una de las etapas del desarrollo se realizaron pruebas sobre los navegadores mas utilizados en la actualidad, como son Opera (figura 25), Safari (figura 26), Internet Explorer (figura 27) y Mozilla Firefox (figura 28), realizando los ajustes necesarios en el código HTML y las hojas de estilo para mantener un aspecto consistente en la interfaz en los distintos navegadores, obteniendo resultados un poco diferentes en los distintos navegadores pero que no afectan en la interacción con la aplicación.

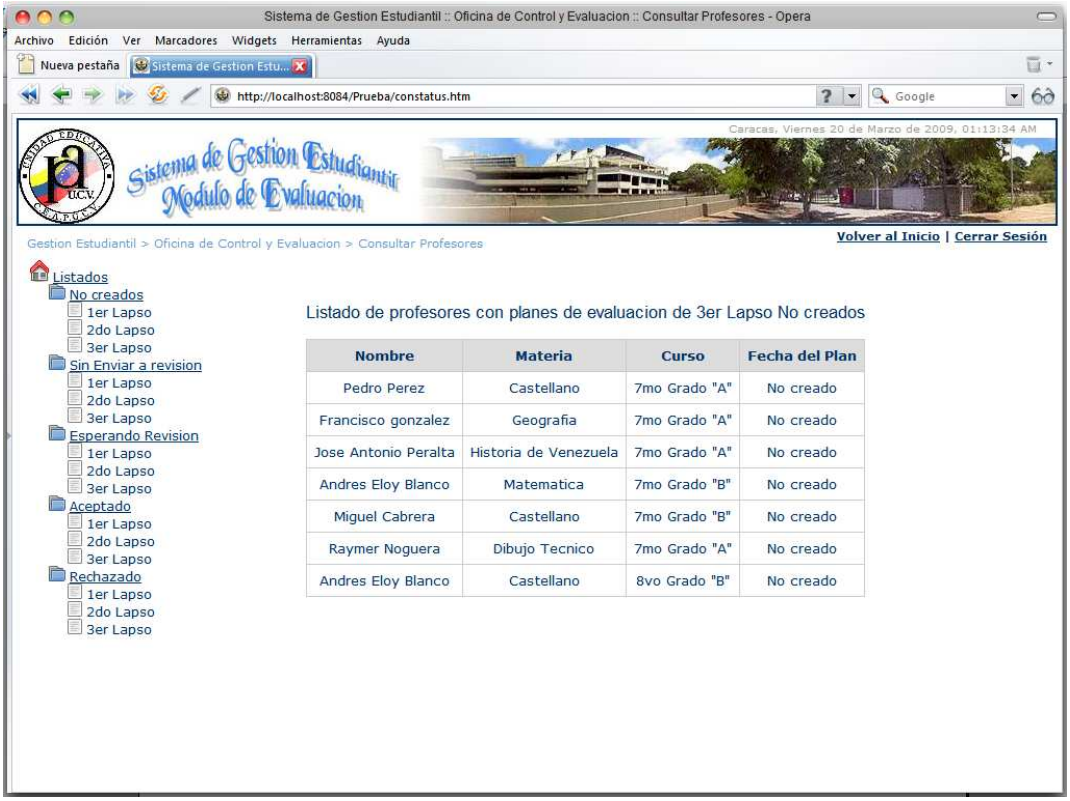


Figura 25 Opera

Figura 26 Safari

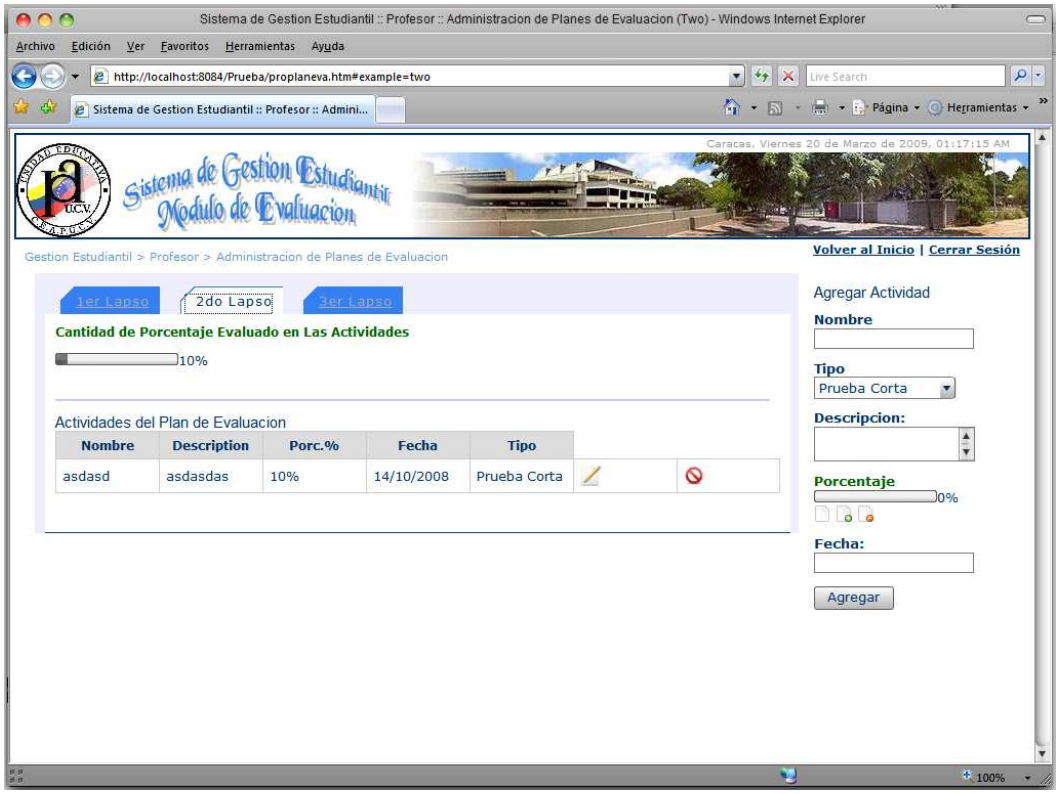


Figura 27 Internet Explorer

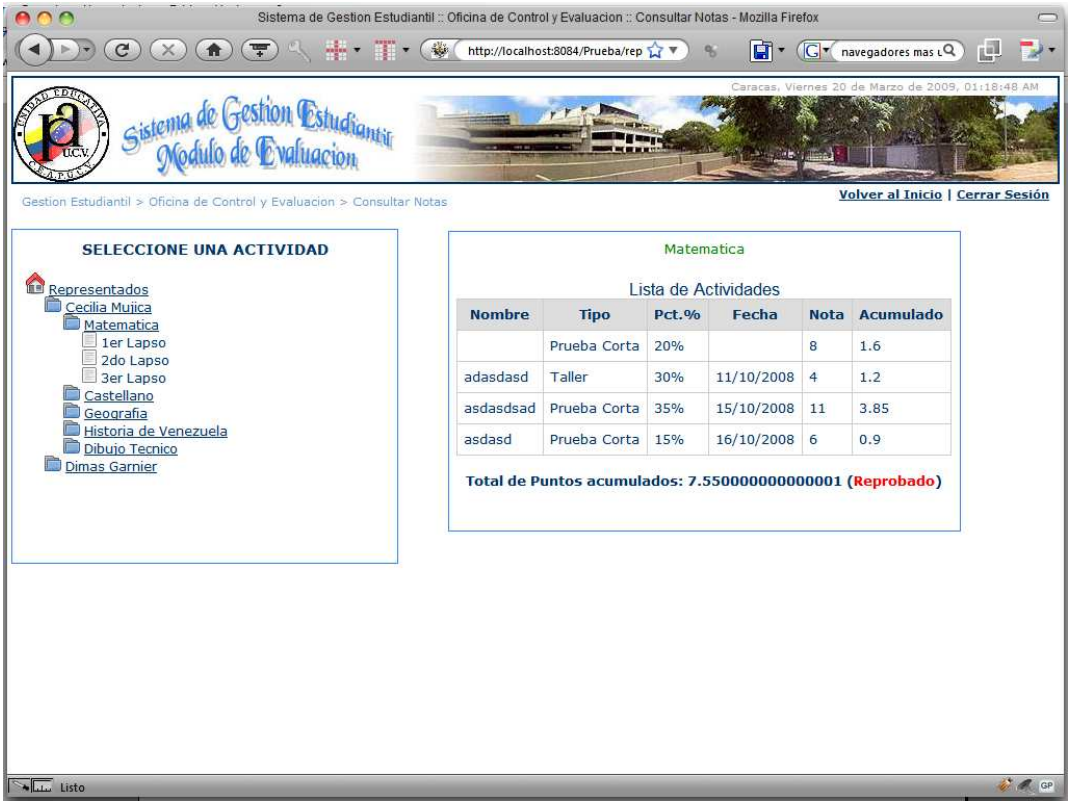


Figura 28 Mozilla Firefox

3.4. Integración de DWR

Para permitir la invocación asíncrona de métodos remotos con DWR es necesario realizar una configuración a través de archivos XML (utilizando el archivo `dwr.xml`) para especificar los métodos remotos y los objetos que deben ser convertidos de Java a JavaScript; de esta forma DWR se encargaría de instanciar los objetos remotos que serán accedidos asíncronamente desde el cliente.

```
<!DOCTYPE dwr PUBLIC "-//GetAhead Limited//DTD Direct Web Remoting
1.0//EN" "http://www.getahead.ltd.uk/dwr/dwr10.dtd">
<dwr>
 <allow>
 <create creator="new" scope="session" javascript="Facade">
 <param name="class" value="com.ges.facade.Facade"/>
 <include method="autenticar"/>
 <include method="obtenerUsuario"/>
 <include method="cerrarSesion"/>
 <include method="agregarActividad"/>
 <include method="borrarActividad"/>
 <include method="revisarActividades"/>
 <include method="obtenerActividad"/>
 <include method="solicitarAprobacion"/>
                                    .
                                    .
                                    .
                                    .
                                    .
 </create>
 <convert converter="bean" match="com.ges.persistence.Plan"/>
 <convert converter="bean" match="com.ges.persistence.Alumno"/>
 <convert converter="bean" match="com.ges.persistence.Nota"/>
                                    .
                                    .
                                    .
 </allow>
</dwr>
```

Como se muestra en el diagrama de componentes de la Figura 10 la aplicación se implementó utilizando el patrón Facade donde una clase sirve de fachada para el acceso a los métodos de la lógica del negocio: de esta manera, los únicos métodos remotos que se configuraron para DWR son los de la clase `Facade`. Esta configuración se realizó a través de la etiqueta `<include method>`, en la cual se indica por un parámetro anterior donde se encuentra la clase que contiene al método a ser llamado remotamente y en la etiqueta se coloca el nombre de cada uno de los métodos a ser invocados desde el archivo JavaScript. Dentro de la etiqueta `<convert>` se especifican todos los tipos de datos que deben ser transformados por dwr para poder ser utilizados desde JavaScript. A continuación se muestra una función escrita en el

lenguaje JavaScript que hace una invocación de un método remoto declarado en el archivo `dwr.xml`.

```
function loginFormSubmitHandler() {
 var user = {
 login:$F('login'),
 pass:$F('pass')
 };
 if(user.login==""||user.pass=="")
 return false;
 $('formulario_inicio').startWaiting('bigWaiting');
 $('loginform').reset();
 Facade.autenticar(user,recibir);
 return false;
}
```

Como se puede apreciar en el codigo anterior, se observa el objeto `Facade` y el llamado a la función `autenticar` la cual reside en el servidor en el paquete `com.ges.facade.Facade` en el archivo `Facade.java` y cuyo código, el cual esta escrito en Java, es el siguiente:

```
package com.ges.facade;
public List autenticar(Usuario u){
 UsuarioManager um = new UsuarioManager();
 ScriptSession session = null;
 HttpSession httpSession = null;
 contexto = WebContextFactory.get();
 session = contexto.getScriptSession();
```

En el código JavaScript se puede notar que se le pasan dos argumentos como parámetros a la función autenticar, mientras la función autenticar que se encuentra en el archivo `Facade.java` recibe un solo parámetro, esto se debe a que el segundo parámetro que se le agrega a la función llamada desde JavaScript es el nombre de la función que va a ser utilizada como callback, es decir, la función que va a recibir y procesar los datos devueltos por el servidor, hacia el cliente cuando se hizo la llamada remota a la función `autenticar`. De esta forma se realiza la comunicación asíncrona de AJAX usando el framework DWR.

3.5. Implementación del lado del servidor

La aplicación maneja del lado del servidor varias capas que fueron divididas por paquetes, según su nivel. La ruta principal de los paquetes es com.ges. Así, según su nivel en la arquitectura multicapa y el papel que juegan sus clases, los paquetes definidos son los siguientes:

facade: Contiene la clase que sirve como fachada y que contiene los métodos que son invocados desde el cliente.

manager: Contiene las clases que manejan todos los accesos a los objetos persistentes.

util: Contiene todas las utilidades externas a la lógica de programación del sistema, como el SessionFactory de Hibernate, funciones para cambiar de formato fechas, etc.

persistence: Contiene las clases persistentes de la aplicación que son mapeadas a la base de datos utilizando los archivos de mapeo del framework de persistencia Hibernate ubicados en el mismo paquete.

A continuación se describen las clases de mayor importancia contenidas en cada paquete.

3.5.1. Paquete facade

Dentro del paquete `facade` se encuentra una única clase con el mismo nombre, la cual provee todos los métodos que pueden ser invocados desde el cliente de forma asíncrona a través del framework DWR.

Los métodos de mayor importancia se listan en la siguiente tabla.

Método	Descripción
`void autenticar(Usuario u)`	Recibe un objeto usuario de la vista y hace una llamada al objeto de negocio correspondiente para que verifique su existencia en la base de datos.
`Usuario obtenerUsuario()`	Verifica si el usuario se ha autenticado e invoca a las funciones JavaScript correspondientes, para mostrar sus datos o redireccionar a la página de autenticación.
`void cerrarSesion()`	Invalida la sesión del usuario actual y

	redirecciona la pagina actual a la pagina de inicio
void revisarActividades(String lapso)	Dado un periodo (lapso) busca las actividades asociadas a ese periodo de tiempo e invoca a la función JavaScript que muestra los datos.
void cargarAlumnos(int idact)	Crea una lista de alumnos que tengan notas asociadas a una actividad que se desee conocer las notas, en caso de que no tengan ninguna nota cargada agrega a los restantes alumnos del curso.
String crearArbol()	Esta Función crea la estructura de árbol en código HTML que va a ser desplegada en el panel izquierdo de las vistas para la navegación de los usuarios.
void solicitarAprobacion(String lapso)	Envía un plan de evaluación dependiendo del periodo que reciba por parámetros a que sea evaluado por la Oficina de Control y Evaluación.
void cargarNotas(int idusuario, int idplan)	Carga todas las notas correspondientes a las actividades de un determinado alumno y de un determinado plan de lapso.
void notificarPlan(int opt, int id, String obs)	Le cambia el status a algún plan de evaluación en caso de que sea rechazado o aprobado por la Oficina de control de evaluación, además se guarda una posible observación que se haya hecho sobre ese plan.
void actualizarNota(Nota n)	Recibe un objeto Nota de la vista y llama a al objeto de negocio correspondiente para que actualice la correspondiente en la base de datos
void agregarActividad(Actividad a, String lapso)	Recibe un objeto de tipo Actividad y se lo agrega al plan de evaluación correspondiente al periodo que se le indique.
Plan planLapso(String lapso)	Devuelve el plan de evaluación de el periodo indicado por parámetro

3.5.2. Paquete manager

El paquete `manager` contiene todas las clases que implementan la lógica del negocio por cada clase del paquete `persistence` se crea una clase manager que define los métodos que manipulan directamente estos Objetos, de tal manera que la clase `Facade` para acceder a los Objetos del paquete `persistence` y hacer operaciones sobre estos debe invocar al objeto correspondiente que se encuentra en este paquete `manager` con el objeto de separar la lógica del negocio de los objetos de control logrando un mayor desacoplamiento de las capas.

Las Clases contenidas en el paquete `manager` son las siguientes:

- `MateriaManager`: Define los métodos relacionados a la obtención de materias ya sea por profesor, alumno o plan de evaluación.
- `UsuarioManager`: Define los métodos correspondientes al control de usuarios y consulta de la existencia de estos.
- `PlanManager`: Presenta los métodos relacionados con la creación y consulta de los planes de evaluación lo cual es el foco principal de este Trabajo Especial de Grado.
- `NotaManager`: Define los métodos correspondientes a la consulta en inserción de notas en el sistema.
- `ActividadManager`: Presenta los métodos relacionados con la creación y consulta de las actividades correspondientes a cada plan de evaluación.

A continuación se presentan los métodos más importantes de cada una de las clases presentes en el paquete `manager` a fin de entender más a fondo su importancia dentro del sistema

`MateriaManager`

Método	Descripción
`Materia` `porProfesor(int id)`	Devuelve el objeto materia correspondiente al id del profesor, es decir, la materia que dicta ese profesor
`List` `porCurso(int id)`	Retorna la lista de las materias asociadas al curso que se le indica
`Materia` `porPlan(int id)`	Devuelve el objeto materia correspondiente a un id del plan de evaluación indicado, es decir, la materia a la cual corresponde ese plan de evaluación

UsuarioManager

Método	Descripción
Usuario autenticar(Usuario u)	Consulta el usuario, según los datos provistos, en la base de datos, y retorna el objeto con todos los datos del usuario
List porCurso(int id)	Retorna la lista de los usuarios (Alumnos) asociadas al curso que se le indica.
List porRepresentante(int id)	Retorna la lista de los usuarios (Alumnos) asociadas al representante que se le indica.

PlanManager

Método	Descripción
Plan planPeriodoProfesor(int idprofesor, String lapso)	Consulta el usuario, según los datos provistos, en la base de datos, y retorna el objeto con todos los datos del usuario
List porMateria(int id)	Retorna la lista de los planes de evaluacion asociados al id de la materia que se le indica

NotaManager

Método	Descripción
Nota porActividadAlumno(int idalumno, int idactividad)	Devuelve el Objeto Nota asociado al alumno y la actividad indicada

ActividadManager

Método	Descripción
List buscarActividades(String lapso, int idprofesor)	Retorna una lista con todas las actividades correspondientes al lapso y a la materia dictada por el profesor indicado
List porPlan(int id)	Retorna la lista de las actividades correspondientes a un plan de evaluación dado.
List conNotas(int idusuario, int idplan)	Retorna una lista de todas las actividades y la nota de esta correspondiente a un usuario (Alumno) determinado.

Todas las clases contenidas en el paquete manager también implementan tres funciones adicionales cada uno, una de inserción o guardado, una consulta y una de eliminación de objetos, llamadas `save`, `findById` y `delete` respectivamente, a las cuales no se le hicieron referencia en los cuadros anteriores por ser estas unas funciones generales y comunes para cada clase.

El paquete `manager` a su vez realiza todas las consultas, inserciones y actualizaciones en la base de datos a través del framework de persistencia Hibernate. Para esto hace uso de las entidades que se encuentran en el paquete `persistence` sus archivos de mapeo y del Objeto `HibernateUtil` contenido en el paquete `util`.

En el siguiente fragmento de código se muestra cómo el método autenticar() hace uso del Objeto `HibernateUtil` para crear la sesión de persistencia y una consulta para obtener una entidad de tipo `Usuario` que se encuentra en el paquete `persistence`.

```
public Usuario autenticar(Usuario u) {
 try{
 SessionFactory factory = HibernateUtil.getSessionFactory();
 Session session = factory.getCurrentSession();
 session.beginTransaction();
 Query query = session.createQuery("from Usuario where login = ? and
pass = ?");
 query.setString(0, u.getLogin());
 query.setString(1, u.getPass());
 return (Usuario) query.uniqueResult();
 }catch(Exception e){e.printStackTrace();}
 return null;
 }
```

Como se muestra en el fragmento de código, la referencia al Objeto `HibernateUtil` devuelve una sesión para crear un objeto de tipo Query. Seguidamente a este objeto se le asignan los parámetros para posteriormente ejecutar la consulta, la cual retorna una entidad del tipo `Usuario`.

Es importante destacar que gracias al uso de Hibernate el desarrollador no tuvo que preocuparse por detalles del modelo de datos luego de la primera configuración; inclusive los querys son basados en el modelo orientado a objetos y no es necesario hacer recorrido a los resultados para asignarle los valores a las entidades, éstas ya vienen con la información requerida.

3.5.3. Paquete Persistence

El paquete persistence contiene todas las entidades persistentes de la aplicación y sus correspondientes archivos de mapeo hbm utilizados por Hibernate para manejar la persistencia. Cada una de las clases presentes en este paquete esta asociada a cada una de las tablas presentes en el modelo de datos representado en la figura del capitulo 3 de analisis y diseño.

A continuación se muestran las figuras correspondientes a la tabla, el código de la clase y el contenido del archivo hbm a fin de ilustrar como se encuentrar relacionados estos entre si.

actividad	
PK	idactividad
	nombre descripcion **porcentaje** fecha
FK1,I1 **FK2,I2**	**idplan** **tipo**

Tabla actividad en el Modelo de datos

```
<hibernate-mapping>
  <class catalog="plan" name="com.ges.persistence.Actividad"
table="actividad">
    <id name="idactividad" type="java.lang.Integer">
      <column name="idactividad"/>
      <generator class="identity"/>
    </id>
    <property name="nombre" type="string">
      <column length="100" name="nombre"/>
    </property>
    <property name="descripcion" type="string">
      <column length="250" name="descripcion"/>
    </property>
    <property name="porcentaje" type="int">
      <column name="porcentaje" not-null="true"/>
    </property>
    <property name="fecha" type="string">
      <column length="10" name="fecha"/>
  </property>
    <property name="idplan" type="int">
      <column name="idplan" not-null="true"/>
    </property>
    <property name="tipo" type="int">
      <column name="tipo" not-null="true"/>
    </property>
  </class>
</hibernate-mapping>
```

Contenido de Archivo Actividad.hbm.xml

```
public class Actividad{


     private Integer idactividad;
     private String nombre;
     private String descripcion;
     private int porcentaje;
     private String fecha;
     private int idplan;
     private int tipo;

//getters y setters de cada atributo

}
```

Contenido de Archivo Actividad.java

Se puede observar que las clases contenidas en el paquete persistente son clases normales con una serie de atributos con la única peculiaridad que el tipo de datos corresponde con el de su homónimo en la base de datos, es en el archivo de mapeo (hbm) donde se le indica a `Hibernate` que esta clase esta relacionada con la tabla actividad de la base de datos y se le indica cual cuales campos van a ser mapeados en la base de datos, como buena practica de programación y para mantener el orden y la legibilidad se le suelen colocar los mismos nombres pero esto no tiene por que corresponder, mientras se le indique de forma correcta en el archivo de mapeo que campo mapea a cual y que clase mapea a que tabla

De esta forma quedan integradas todas las piezas que conforman la estructura del sistema desarrollado para este trabajo especial de grado

CONCLUSIONES

El desarrollo de software es una actividad que implica análisis y preparación en el que se debe tomar en cuenta aspectos como la tecnología a utilizar, su costo y la metodología. El enfoque de desarrollo AJAX proporciona una nueva forma de interactuar con las aplicaciones Web, la cual hace que el usuario se comunique de una forma mas fluida con el sistema.

Después de realizar esta investigación queda evidenciado que un software basado en el enfoque de desarrollo AJAX puede ser implementado sin ningún inconveniente en centros educativos con el fin de optimizar sus procesos y comunicaciones, tanto internas como externas ya que se lograron cumplir todos los requerimientos solicitados por la comunidad estudiantil.

La limitante de este tipo de sistemas será la capacidad de soporte que se le puede dar a estas aplicaciones al estar basadas en tecnologías especializadas. Por lo que se requeriría de personal especializado a la hora de dar un soporte a este sistema.

Contribuciones

Los principales aportes que se generaron al culminar el Sistema de Gestión Escolar fueron los siguientes:

- Se desarrolló un sistema que permita automatizar actividades como:
  - La creación por parte de los profesores de los planes de evaluación correspondientes a cada lapso y el envió de los mismos a la Oficina de Control y Evaluación para su evaluación.
  - Evaluación y notificación de aprobación o rechazo de los planes de evaluación por parte de la Oficina de Control y Evaluación
  - Consultar todas las notas, actividades y planes de evaluación de cada materia, de los alumnos de la institución.
- Usando herramientas de desarrollo de Aplicaciones de Internet Enriquecidas se pudo obtener un sistema con una interacción mucho más fluida y rápida que con una aplicación Web tradicional.

Recomendaciones y trabajos futuros

El presente trabajo de grado se enfocó en la implementación del módulo principal de la aplicación: el control de hojas de tiempo. Es así como la investigación que se realizó en este trabajo pretende servir de base para trabajos futuros que puedan implementar y/o enriquecer los siguientes módulos sobre la arquitectura planteada.

Tomando en cuenta lo anterior, se proponen los siguientes trabajos futuros y recomendaciones:

- Integrar este modulo del sistema con el sistema ya existente en la institución de comunicación y agenda de miembros de la comunidad
- Utilizar mecanismos para que la información perdure en el tiempo y se tenga una base de datos con planes de evaluación antiguos para poder re-usarlos y notas de otros años escolares.
- Utilizar http sobre ssl (https) para proteger la información sensible y evitar los accesos indeseados.
- Parametrizar de forma mas avanzada las opciones correspondientes a la configuración del sistema, como numero de lapsos, cantidad de materias dictadas por un profesor, etc.
- Implementación de un módulo de reportes sobre los datos manejados por el sistema.
- Implementación de un modulo que notifique los cambios en el en el sistema por correo electrónico.

Referencias Bibliograficas

[1] Asleson, R. & Schutta, R. T. (2005). *Foundations of Ajax*. Apress.

[2] Gadge, V. V. (2006). Technology options for Rich Internet Applications. http://www-128.ibm.com/developerworks/web/library/wa-richiapp. (Visitado 12/2007).

[3] Crane, D., Pascarello, E. & James, D. (2005). AJAX in action. Manning Publications Co.

[4] Tutorial de JavaScript de la w3 http://www.w3schools.com/js/default.asp (Visitado 12/2007).

[5] Referencia para el Manejo del DOM de la w3 http://www.w3schools.com/htmldom/dom_reference.asp (Visitado 12/2007).

[6] Robin Alberto Castro Gil, Universidad Icesi (2004). Estructura básica del proceso unificado de desarrollo de software

[7] El Proceso Unificado de Desarrollo de Software. (Visitado 09/2008) http://yaqui.mxl.uabc.mx/~molguin/as/RUP.htm

[8] John Hunt, (2003) Guide to the Unified Process featuring UML, Java and Design Patterns

[9] Philippe Kruchten, (2000) The Rational Unified Process: An Introduction (2nd Edition)

[10] Hamilton, K. & Miles, R. (2006). Learning UML 2.0. O'Reilly

[11] Direct Web Remoting (2008). Documentation.

http://directwebremoting.org/dwr/documentation. (Visitado 06/2008).

[12] Gross C. (2006). Ajax Patterns and Best Practices. Apress.

[13] Prototype JavaScript Framework (2008). Prototype.

http://www.prototypejs.org. (Visitado 06/2008).

[14] Scriptaculous (2008). Scriptaculous.

http://script.aculo.us. (Visitado 06/2008).

[15] Wikipedia (2008). Ritech Internet Application.

http://en.wikipedia.org/wiki/Rich_Internet_application. (Visitado 08/2008).

[16] Wikipedia (2008). AJAX.

http://es.wikipedia.org/wiki/AJAX. (Visitado 07/2008).

[17] Zakas, N. C., McPeakand, J. & Fawcett, J. (2006). Professional Ajax. Wrox Press.

[18] Wells, C.(2007). *Securing Ajax Applications*. O'Reilly Media, Inc.

[19] Lauriat, Shawn M.(2008) *Advanced Ajax : architecture and best practices* Pearson Education, Inc.

[20] Craig Larman. Agile and iterative development: a manager's guide. (2004). Addison Wesley.