Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Aplicaciones con Tecnología Internet

Trabajo Especial de Grado
Sistema de Recolección de Estadísticas y Generación de Reportes Web como una extensión del navegador Mozilla Firefox

Trabajo Especial de Grado
presentado ante la ilustre
Universidad Central de Venezuela
por los Bachilleres
José J. Sanchez P., C.I: 16.179.270
y
Angel D. Gil R., C.I: 17.977.347
para optar por el título de
Licenciado en Computación

Tutor
Prof. Sergio Rivas

Caracas, Mayo del 2011

Acta

Quienes suscriben miembros del Jurado, designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por los **Bachilleres José Javier Sanchez Ponte, C.I. 16.179.270 y Angel David Gil Ramirez, C.I. 17.977.347**, con el título **Sistema de Recolección de Estadisticas y Generación de Reportes Web como una extensión del navegador Mozilla Firefox**, 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 17 de Mayo de 2011, para que sus autores lo defendieran en forma pública, se hizo en la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, 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 diecisiete días del mes de mayo del año dos mil once, dejándose también constancia de que actuó como Coordinador del Jurado el profesor Sergio Rivas.

Prof. Sergio Rivas
Tutor

Prof. Andrés Sanoja
Jurado

Prof. Jossie Zambrano
Jurado

Agradecimientos y Dedicatoria

- A la Universidad Central de Venezuela, alma mater y fuente de conocimientos "La Casa que vence las sombras", por darme la oportunidad de estudiar en tan prestigiosa institución.
- A mi tutor académico, el profesor Sergio Rivas por confiar en nosotros, por su gran ayuda, colaboración, orientación y flexibilidad a lo largo del desarrollo de este proyecto.
- A todos mis amigos con lo que compartí alegrías y penas; y me ayudaron a crecer académica, mental y espiritualmente a lo largo de la carrera.
- Quisiera agradecer a toda mi familia por el apoyo brindado durante el desarrollo de mis estudios universitarios, a mis compañeros y profesores por la ayuda recibida y finalmente a la Universidad Central de Venezuela, como institución, que con sus grandes virtudes y defectos me brindó la oportunidad de culminar exitosamente mis estudios y poder obtener el título de Licenciado en Computación.

José J. Sanchez P

- A mi padre Angel Gil, por haberme orientado y ayudado durante mi formación profesional.
- A mi madre Elisaira Ramirez, quien ha sido un pilar de fortaleza y estimulo para seguir hacia adelante.
- A la Universidad Central de Venezuela, por haberme permitido estudiar en sus prestigiosas aulas.
- A mi tutor académico, el profesor Sergio Rivas por confiar en nosotros, por su gran ayuda, colaboración, orientación y flexibilidad a lo largo del desarrollo de este proyecto.

Angel D. Gil R.

Dedicatorias

La realización de este proyecto está dedicado especialmente:

- A mis padres, por su ayuda incondicional en los buenos y malos momentos, y apoyo en todo momento, quienes con sus sabios consejos y todos sus sacrificios hicieron posible este proyecto.
- A mi abuelo Manuel Salvador Ponte Betancourt, y a mi abuela Maria Luisa Ponte García, que ya partieron de esta tierra, y que desde el cielo me bendicen con su amor, sabiduria y protección.
- A dios quien me dió la fuerza, paciencia, constancia y espíritu de lucha para superar las adversidades a lo largo de la carrera.

José J. Sanchez P

- A mi padre Angel Gil y a mi madre Elisaira de Gil por siempre estar a mi lado y dandome las fuerzas para continuar con este objetivo y lograr alcanzarlo.
- A mi tía Margarita Bastardo por su ayuda durante toda la carrera e inclusive en el desarrollo de este trabajo.
- A los compañeros de trabajo en especial Esteban Soto, y Jean Carlos por icentivarme y apoyarme en los momentos dificiles del desarrollo de este trabajo.

Angel D. Gil R.

Resumen

Hoy en día, muchos sitios Web carecen de mecanismos que les permitan obtener información acerca del comportamiento de los usuarios que navegan por la red. La mayoría de los servicios de estadísticas ofrecidos, realizan un análisis del comportamiento de los usuarios que acceden a un sitio Web, de manera global, ya que tienen la necesidad de abarcar la mayor cantidad de sitios Web posibles a quienes puedan ofrecer sus servicios. Sin embargo, dichos sistemas no proporcionan información de un usuario en específico, ni el comportamiento que estos presentan en aspectos relacionados a la navegabilidad, funciones utilizadas, tiempo de respuesta del usuario para completar una tarea específica en un sitio Web, tipos de interacción, etc.

Dada esta situación, en el presente trabajo se desarrolla un sistema de estadísticas basado en la arquitectura Cliente-Servidor encargado de recolectar estadísticas y mostrar la información acerca del comportamiento del usuario mientras navega por la Web, brindando reportes que pueden ser analizados por un ente interesado y lograr mejoras relacionadas a la usabilidad del sitio Web.

Palabras Clave: XUL, XPCOM, JavaScript, XPConnect, JSF,extensión, estadísticas.

Contactos

José Javier Sanchez Ponte **nirvana01@gmail.com**

Angel David Gil Ramirez **angelgil87@gmail.com**

Sergio Jose Rivas Atanacio (Tutor) **sergiorivas@gmail.com**

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción y Problema

En el presente capítulo se proporciona una explicación acerca de la situación actual de los servicios de estadísticas. Posteriormente se hace referencia a los diversos Analizadores de información tales como los Analizadores de logs de servidores Web y los Servicios de Estadísticas, explicando a groso modo su propósito y las desventajas de su utilización; y la solución propuesta en el presente trabajo especial de grado.

1.1. Situación actual

Cada vez más Internet es el canal de comunicación global por excelencia. El intercambio de cualquier tipo de información por vía electrónica dejó de ser ficción para convertirse en realidad.

En muchos países del mundo se ha denotado una fuerte tendencia hacia el aumento de usuarios en la red y gracias a esto, existen una gran variedad de sistemas que ofrecen sus servicios para la recolección de datos en la Web cuyo diseño, interacción y procesamiento de información parecen no estar dirigidos en pro de las necesidades y/o expectativas de los usuarios, debido a que la información recolectada por los mismos tratan de englobar la mayor cantidad de mercados posibles; en base a un conjunto de estudios realizados por dichos servicios para determinar que productos ofrecer, y así tratar de generar ventajas competitivas para sus clientes, de forma gratuita o mediante alguna forma de pago; pero generalmente no suelen estar centrados en la interacción del usuario.

Entre los servicios más comunes ofrecidos podemos mencionar el análisis de tráfico existente en la Web, sistema operativo empleado por el usuario, número de visitas en una página Web determinada, y lugar de procedencia del usuario. Respecto a estos dos últimos aspectos, se contemplan una serie de limitantes que serán descritas a lo largo del presente trabajo de grado.

Además suelen utilizar una gran variedad de analizadores de información, entre los cuales cabe mencionar: los analizadores de logs de servidores web y los servicios de estadísticas.

A continuación, una explicación acerca de los analizadores mencionados:

1.1.1. Analizadores de logs de servidores web

Dichos analizadores de logs generan información acerca del uso de un sitio web. Estos a su vez, constituyen una de las herramientas más confiables, pero tienen como inconveniente, que dichos logs muestran la información de manera engorrosa, poco legible, y de forma genérica.

Otra forma de poder realizar un análisis de la información de un sitio web son los proxys. Estos presentan información básica sobre el acceso a determinados sitios web, pero los mismos no incluyen aspectos relacionados a la interacción del usuario con la página a través de un navegador.

1.1.2. Servicios de estadísticas

Como fuente de información adicional existen los llamados servicios de estadísticas en la web, que generan una gran variedad de reportes estadísticos para los sitios web que incluyen sus servicios, y los cuales ofrecen indicadores generales que pueden ser visualizados, pero no muestran detalles específicos como por ejemplo, la navegabilidad de un usuario en un sitio web, además de otros indicadores relacionados a la interacción del usuario y la página.

Además muchos de estos servicios de estadísticas generan código JavaScript, que debe ser incorporado a cada una de las páginas de los sitios web que son monitorizadas utilizando las funcionalidades creadas por estos, lo cual puede resultar engorroso si el sitio web, no presenta un diseño bien estructurado, e inclusive el código JavaScript puede no ser compatible con algunos navegadores.

Cabe agregar que, el código JavaScript disminuye el rendimiento de las páginas, ya que debe ser ejecutado internamente en la página para que pueda ser monitorizado.

Por lo expuesto anteriormente podemos deducir que existen grandes brechas en los servicios ofrecidos por los generadores de estadísticas existentes en la Web, además de los bajos medidores que se obtienen de los analizadores de logs.

En general, los servicios y analizadores de información expuestos permiten realizar un análisis del comportamiento de los usuarios que acceden a un sitio web, de manera global. Esto es debido a que la mayoría de los sistemas que utilizan los servicios

y/o analizadores mencionados, tienen la necesidad de abarcar la mayor cantidad de sitios web posibles a quienes puedan ofrecer sus servicios, y por ello la información presentada por los mismos trata de dar a conocer servicios que sean de interés para la mayoría de los usuarios; sin embargo muchas veces no satisfacen las expectativas o necesidades de un usuario en particular que necesite recolectar otro tipo de información; a diferencia de un sistema de estadísticas configurable que muestre información de un usuario en específico, que permita clasificar o categorizar a los usuarios según sus necesidades, obtener información acerca del comportamiento que estos presentan en aspectos relacionados a la navegabilidad, funcionalidades utilizadas, tiempo de respuesta del usuario para completar una tarea específica en el sitio web, tipos de interacción, etc.

Lo cual constituye una especie de guía de diseño, que le muestre al ente interesado, ya sea el administrador, o un ente externo del sitio web, de que forma puede lograr mejoras relacionadas a la usabilidad del mencionado.

De acuerdo a lo expuesto anteriormente, se propone la implementación de una herramienta de estadísticas que permita obtener información de interés de uno o varios sitios web, conformada por una aplicación cliente (recolector de datos) y una aplicación servidor (almacenador y encargado de mostrar la información obtenida de manera gráfica), quienes establecerán comunicación entre sí mediante servicios web.

La aplicación cliente reside como una extensión del navegador Mozilla Firefox, y cuya función principal es la recolección configurable de datos, mediante la interacción del usuario con el navegador, a través de una página web.

Dichos datos posteriormente son analizados por la aplicación servidor quien se encarga de capturar la información proporcionada por la aplicación cliente, y almacenarla utilizando un sistema de manejador de bases de datos. Además de poder analizar y generar información estadística, que pueda ser visualizada en un sitio web residente en el servidor.

La recolección de información por parte de la aplicación cliente, y la visualización de la información de forma gráfica obtenida del mismo, por parte de la aplicación servidor, permite conocer más a fondo al usuario, y a su vez contribuye a una mejor toma de decisiones en aspectos relacionados al análisis, diseño, desarrollo, mantenimiento e innovación de sitios web.

Además de resolver los inconvenientes de generar código JavaScript, que debe ser incorporado a cada una de las páginas de los sitios web que van a ser monitorizados, como es el caso de los servicios de estadísticas, debido a que este funciona como una extensión del navegador Mozilla Firefox, con lo cual no es necesario agregar código a las páginas que serán estudiadas y a su vez se evita una disminución del rendimiento de las mismas, ya que dicho codigo debe ser ejecutado internamente en la página para

que pueda ser monitorizado, en contraposición al sistema anterior, donde el código Javascript se ejecuta como extensión del navegador y no internamente en la página.

1.2. Propuesta tecnológica

A continuación se muestra un gráfico de la tecnología propuesta para la realización del sistema de estadísticas anteriormente mencionado:

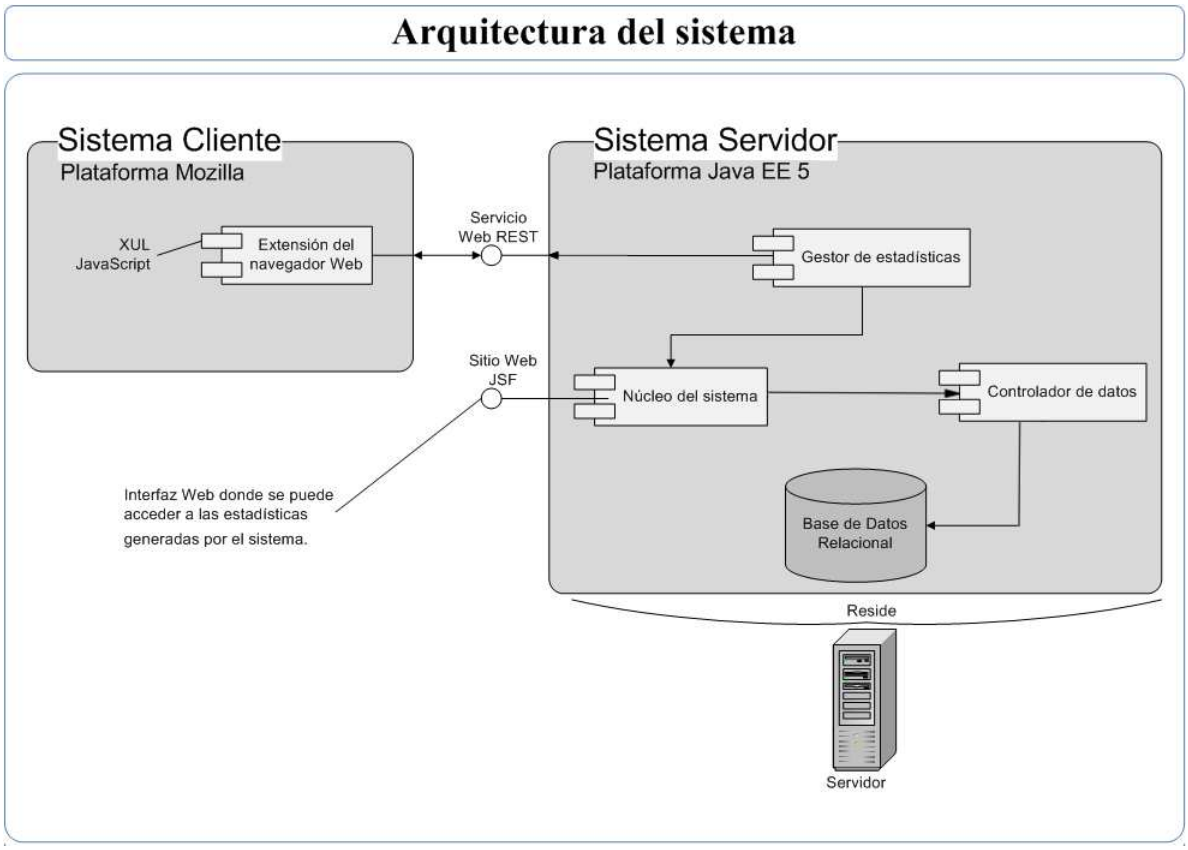


Figura 1.1: Arquitectura propuesta para el TEG

1.2.1. Explicación de la arquitectura propuesta

En la presente sección se proporciona una síntesis de la arquitectura propuesta.Dicha arquitectura, se encuentra dividida en dos componentes: Sistema Cliente y Sistema Servidor.

1.2.2. Sistema Cliente

Es el encargado de recolectar información acerca del comportamiento del usuario mientras navega por la Web. Dicho sistema está conformado por 3 componentes:

- **Extensión del navegador Web:** Representa la capa de presentación de la aplicación, que fue creada utilizando el lenguaje de interfaces de usuario XUL (XML User Interface).
- **Comunicación con el núcleo:** Encargado de capturar información acerca de los eventos realizados por el usuario mientras navega, ya sea sobre una página Web determinada o sobre la capa de presentación de la extensión, tales como, el invocar una interfaz de usuario, haciendo click en algunas sus opciones. El componente *Comunicación con el núcleo*, dependiendo de los eventos acaecidos en los escenarios ya descritos, establece o no una comunicación con el componente denominado *Componentes de la arquitectura de Mozilla Firefox*, para obtener información acerca de un determinado evento.
- **Componentes de la arquitectura del navegador Mozilla Firefox:** Conformada por un componente multiplataforma que posee un modelo de objetos similar al Microsoft COM, denominado XPCOM, que permite que los componentes u objetos XPCOM, puedan ser utilizados o implementados por lenguajes de programación tales como: JavaScript, Java, Python, o C++; y la tecnología XPConnect, que permite la interoperabilidad entre componentes XPCOM y procedimientos implementados en JavaScript.

1.2.3. Sistema Servidor

Se encarga de recibir la información recolectada por la aplicación cliente, y la cual es enviada por el mencionado, utilizando servicios Web REST implementados en JavaScript. El sistema servidor está conformado por los siguientes componentes:

- **Manejador de la interfaz de servicios Web:** Es el conjunto de componentes implementados por medio de JAX-WS para el despliegue de servicios web REST, a objeto de que el cliente pueda enviar las estadísticas y obtener la configuración de los proyectos definidos en el servidor.
- **Gestor de estadísticas:** Constituyen los componentes invocados por la interfaz de servicios web para cargar las estadísticas de un documento XML en la base de datos, o generar el XML de los proyectos configurados en el sistema.
- **Núcleo del sistema:** Conformado por los componentes que permiten la creación y generación de los reportes en el sistema Web y ofrecen la comunicación de la capa web con la capa de persistencia.

- **Controlador de datos:** Tiene como función ofrecer una comunicación transparente entre el núcleo y la interfaz web con la base de datos del sistema. Está constituida por el conjunto de entidades de persistencia.

1.2.4. Funcionamiento

Inicialmente el usuario debe crear un proyecto de estadísticas en el Sistema Servidor, a través de un sitio Web implementado en JSF (Java Server Faces) 1.2. Una vez creado el proyecto, se instala la aplicación cliente en el navegador Mozilla Firefox y se realizan las configuraciones necesarias para iniciar el proceso de recolección.

Dicho proceso se realiza de la siguiente manera:

El Sistema Cliente solicita un id de visita al Sistema Servidor a través de un servicio Web REST implementado en JavaScript. Si el Sistema Servidor no se encuentra activo, o no esta disponible, no se inicia la recolección de estadísticas en el Sistema Cliente. En caso contrario, el Sistema Servidor, envía como respuesta un id de visita a la aplicación cliente, utilizando un servicio Web implementado en JAX-RS (Java API for RESTful Web Services).

Posteriormente cuando un usuario ingresa a un dominio en la web; y dicho dominio pertenece al proyecto sobre el que se desea realizar la recolección de estadísticas; sino existe una visita activa, se inicia una nueva visita en la aplicación cliente. La aplicación cliente inicia la recolección de información. Si el timeout de visita seleccionado por el usuario (el del Sistema Cliente o el del Sistema Servidor) expira, o el usuario decide cerrar el navegador, el Sistema Cliente envía la información recolectada hasta los momentos al Sistema Servidor en forma de un archivo XML, utilizando un servicio Web REST implementado en JavaScript. Luego de enviar las estadísticas, el ciclo se inicia nuevamente.

Una vez finalizada la recolección de datos estadísticos, el Sistema Servidor obtiene la información recolectada por el cliente, mediante un archivo XML enviado por el Sistema Cliente. Dicha información es almacenada en la base de datos del Sistema Servidor, y a partir de su contenido se procede a la construcción de diversos reportes, en base a las preferencias de configuración del proyecto seleccionadas por el usuario, en el presente sistema, al momento de crear el proyecto de su preferencia.

1.3. Objetivo general

Diseñar e implementar un Sistema de Recolección de estadísticas Web que obtenga información acerca del comportamiento del usuario mientras navega por la red, para que posteriormente, dicha información sea almacenada y visualizada de manera gráfica.

1.4. Objetivos específicos

- Diseñar e implementar una aplicación del lado del cliente encargada de realizar el proceso de extracción de los datos, como una extensión para el navegador Mozilla Firefox.
- Diseñar las interfaces de usuario correspondientes a la aplicación cliente para la recolección de datos del Sistema de Visualización y generación de estadísticas web configurables.
- Desarrollar los módulos encargados de medir la interacción del usuario con la página web.
- Implementar el núcleo de la aplicación cliente encargado de la invocación de los servicios web residentes en la aplicación servidor.
- Implementar el núcleo de la aplicación servidor, que permita la creación de proyectos de estadísticas.
- Diseñar e implementar la base de datos residente de la aplicación servidor que permita almacenar la información relacionada a los parámetros de interacción del usuario y la pagina, e integrarla al modelo de datos existente del Sistema de Visualización y generación de estadísticas Web.
- Desarrollar los módulos de la aplicación servidor encargados de generar los reportes estadísticos que serán mostrados.
- Integrar la aplicación cliente con la aplicación servidor para formar el sistema de visualización y generación de estadísticas web.
- Aplicar pruebas a las funcionalidades desarrolladas en la aplicación cliente y en la aplicación servidor, para verificar su correcto funcionamiento y comunicación, una vez integradas con el Sistema de Visualización y generación de estadísticas Web.
- Utilizar un proceso de desarrollo ligero, enfocando el desarrollo de la aplicación en función de los requerimientos comunicados por el cliente.

1.5. Proceso de desarrollo

La metodología a utilizar será eXtremme Programming (XP) o Programación Extrema, la cual constituye un proceso de desarrollo ligero, que intenta reducir la complejidad del software por medio del trabajo en equipo; generalmente en parejas; a objeto de desarrollar la mayor cantidad de código en el menor tiempo posible, la transmisión de conocimientos entre los diversos integrantes de un proyecto de desarrollo; a objeto de que los programadores incrementen sus habilidades y por ende; el desarrollo de las

aplicaciones sea agilizado; y el desarrollar código de calidad, con la menor cantidad de documentación posible.

Parte I

Marco Teórico

Capítulo 2

Firefox

En el presente capítulo se explican los conceptos manejados en la arquitectura del navegador Web Mozilla Firefox, de igual manera se exponen el conjunto de tecnologías y lenguajes que hacen posible el funcionamiento del mismo. Adicionalmente se define el término *extensión* en el contexto del navegador Web y se presentan las ventajas y desventajas de una extensión.

2.1. Definición

Es un navegador Web de código abierto desarrollado en el lenguaje de programación C++ que tiene la capacidad de incorporar cualquier número de aplicaciones complementarias que pueden ejecutarse en máquinas que utilicen la mayoría de los sistemas operativos de escritorios existentes, tales como: Microsoft Windows, OS X, Linux, MacOS, entre otros [Feldt, 2007].

2.2. Arquitectura multiplataforma de Mozilla Firefox

Mozilla Firefox no es solo una aplicación convencional, su arquitectura está relacionada con aplicaciones Web que emplean HTML dinámico. Está constituída por plataformas basadas en estándares que permiten el manejo de eventos codificados en lenguaje JavaScript, y que residen en la interfaz XUL, que será explicada en el capítulo 4.

Por otra parte, el funcionamiento de la arquitectura del navegador Web radica en combinar lo que el usuario visualiza (a través del contenido Web) y las tecnologías fundamentales usadas para accesar a la información, decodificar, desplegar y estilizar el contenido. Las tecnologías que hacen esto posible son quizás las menos comprendidas del conglomerado de innovaciones de Mozilla [Firefox, 2009b].

En si la arquitectura del navegador web está basada en las siguientes tecnologías y lenguajes [Firefox, 2009d]:

- CSS
- JavaScript
- XUL
- XPIDL
- XPConnect
- XPCOM
- Gecko
- C++

Las tecnologías y lenguajes anteriormente listados están asociados a los siguientes componentes que conforman la arquitectura:

- **Interfaz Gráfica de Usuario (GUI):** corresponde a toda la capa de presentación de la aplicación al usuario final y está estructurada y basada en el lenguaje XUL. Adicionalmente la presentación del GUI es definida a través de archivos CSS [Firefox, 2009d].
- **Lógica de control:** conforma todos los procesos de manejo de eventos entre la interfaz de usuario y el núcleo del sistema. Esta lógica es manejada por medio del lenguaje JavaScript [Firefox, 2009d].
- **Modelo de componentes:** se encarga de establecer la comunicación entre la capa de control y el núcleo a través de un sistema de objetos multiplataforma. Este modelo es manejado por medio de las tecnologías XPConnect y XPCOM [Firefox, 2009d].
- **Núcleo del sistema:** contiene toda la lógica de funcionamiento básica de la aplicación y el análisis del contenido Web para que pueda ser desplegado. El núcleo se encuentra desarrollado en C++ y utiliza como plataforma de renderizado a Gecko [Firefox, 2009d] .

La figura 2.1 muestra la arquitectura del navegador Mozilla Firefox [Firefox, 2009b].

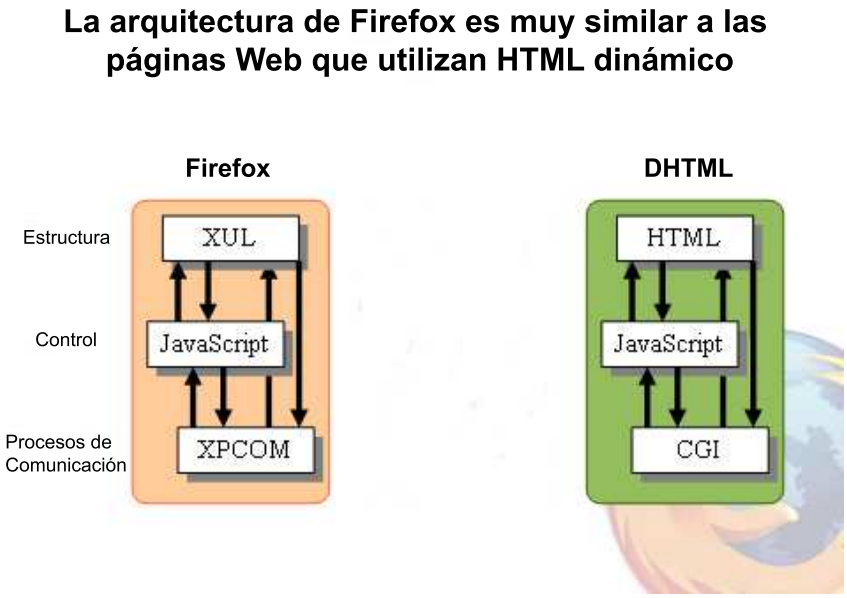


Figura 2.1: Arquitectura de Firefox

CSS (Cascading Style Sheets): es el lenguaje utilizado para describir la presentación de documentos HTML o XML. Esto incluye varios lenguajes basados en XML como son XHTML o SVG. Además, es usado para modificar la interfaz de usuario de aplicaciones y programas, este es el caso de los productos basados en XUL como son Firefox, Seamonkey o Thunderbird.[Firefox, 2009c] [Firefox, 2009b]

JavaScript: es un lenguaje de scripts, interpretado, multiplataforma y parcialmente orientado a objetos. Fue creado por Netscape específicamente para su uso en el desarrollo de sitios Web. Actualmente es un estándar mantenido por el ECMA. Para mayor información consultar el capítulo 3 contenido en el presente documento. [Firefox, 2009a] [Firefox, 2009b]

XUL (XML User Interface Language): es el lenguaje XML para interfaces de usuario de Mozilla. Permite crear potentes aplicaciones multiplataforma que pueden ejecutarse con conexión a Internet o sin ella; así como permitir que las aplicaciones puedan ser personalizadas con facilidad con texto alternativo, gráficos, y diseños; por lo que pueden ser fácilmente instaladas o traducidas. [Firefox, 2009f] [Firefox, 2009b]

XPIDL (Cross Platform Interface Description Language): es un lenguaje de descripción de interfaces usado para especificar clases en la interfaz XPCOM.

El lenguaje de descripción de interfaces (IDL, por Interface Description Language) es utilizado para describir interfaces independientemente del lenguaje y de la máquina.

Los IDLs permiten definir interfaces que luego puedan ser procesadas por herramientas para autogenerar especificaciones de interface dependientes del lenguaje. [Firefox, 2009e]

XPConnect: es la tecnología que funciona como puente entre JavaScript y XPCOM. Dicha tecnología permite el uso de componentes XPCOM dentro de código JavaScript, así como la interacción con objetos JavaScript dentro de componentes XPCOM. [Firefox, 2007a]

XPCOM(Cross Platform Component Object Model): XPCOM no es un lenguaje de programación. Es un modelo multi-plataforma sencillo, basado en COM que proporciona un lenguaje e interfaces independientes de la plataforma, que otros objetos pueden utilizar para acceder a sus servicios sin conocer su implementación. [Firefox, 2008b]

Gecko: es el nombre del motor de presentación desarrollado por Mozilla. En un principio se llamó NGLayout. La función de Gecko es leer el contenido de la Web, en HTML, CSS o JavaScript, y presentarlo en pantalla o imprimirlo. [Firefox, 2007c]

2.3. Extensión de Firefox

Es un pequeño add-on (complemento agregado) que añade o modifica funcionalidades en una aplicación principal (aplicación host) como Firefox [Firefox, 2008a].

La gama de funcionalidades que abarca una extensión están principalmente dirigidas a la personalización del sistema host. Ejemplos de extensión en Firefox son:

1. Agregar una nueva barra de herramientas con nuevas funcionalidades.
2. Cambiar completamente una característica del navegador.

2.3.1. Ventajas

- No requiere realizar cambios en el código de la aplicación principal con lo cual no es necesaria la compilación de la misma.
- No hace más complejo el código principal de la aplicación.
- Facilidad de integrarse como un módulo adicional en la aplicación.
- Fácil de remover.
- Permite realizar una personalización de la interfaz de usuario.
- Se proporciona informaciósn adicional al usuario.

2.3.2. Desventajas

- Se requiere de una API para establecer una comunicación con la aplicación host.
- Pueden influir negativamente en el rendimiento del sistema.
- Necesita de la reinicialización de la aplicación para completar la instalación.

Capítulo 3

JavaScript

El presente capítulo describe los conceptos involucrados con el lenguaje de programación interpretado JavaScript, iniciando con la definición, especificaciones oficiales y versiones del lenguaje.

3.1. Definición

Es un lenguaje de programación interpretado basado en guiones (scripts) utilizado principalmente en la creación de páginas Web dinámicas. Una página Web dinámica es aquella que incorpora efectos tales como animaciones, acciones que se activan al presionar elementos de interfaz gráfica de una página Web como botones, despliegue de ventanas con mensajes de aviso al usuario, entre otros.

En constraste con un lenguaje de programación no interpretado (como Java), el código fuente JavaScript no requiere de procesos intermedios como la compilación para ser ejecutado, por lo tanto, puede ser probado directamente (código JavaScript embebido en una página Web) en cualquier navegador Web que sea capaz de interpretar el lenguaje.

Por otra parte, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Java es un lenguaje de programación compilado de alto nivel.

Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems. [Producciones, 2008] [librosweb.es, 2009]

3.2. Especificaciones oficiales

ECMA ha publicado varios estándares relacionados con ECMAScript. En Junio de 1997 se publicó la primera edición del estándar ECMA-262. Un año después, en Junio de

1998 se realizaron pequeñas modificaciones para adaptarlo al estándar ISO/IEC-16262 y se creó la segunda edición.

La tercera edición del estándar ECMA-262 (publicada en Diciembre de 1999) es la versión que utilizan los navegadores actuales y se puede consultar gratuitamente en `http://www.ecma-international.org/publications/standards/Ecma-262.htm`.

Actualmente se encuentra en desarrollo la cuarta versión de ECMA-262, que podría incluir novedades como paquetes, espacios de nombres, definición explícita de clases, etc. ECMA también ha definido varios estándares relacionados con ECMAScript, como el estándar ECMA-357, que define una extensión conocida como E4X y que permite la integración de JavaScript y XML. [librosweb.es, 2009]

3.3. Motores de JavaScript disponibles

Mozilla.org ofrece dos motores de JavaScript. El primero JavaScript fue creado por Brendan Eich en Netscape, y ha sido actualizado (la versión 1.5 de JavaScript) para ajustarse a la tercera edición de ECMA - 262 desde entonces. Este motor, cuyo nombre en código es SpiderMonkey, es implementado en C.

Por otro lado, el segundo motor denominado Rhino, y creado principalmente por Norris Boyd (también en Netscape) es un interpretador de JavaScript en Java. De la misma manera que SpiderMonkey, Rhino es compatible con la tercera edición de ECMA - 262. [librosweb.es, 2010]

3.4. Versiones de navegadores y de JavaScript

El lenguaje ha ido avanzando durante sus años de vida e incrementando sus capacidades. En un principio se podían realizar muchas funcionalidades en la página Web, pero se contaba con pocas instrucciones para poder crear efectos especiales. Con el tiempo también el HTML ha avanzado, y se han añadido nuevas características tales como las capas, que permiten estructurar y manejar los documentos de distintas formas [Producciones, 2008].

JavaScript también ha avanzado y para manejar todas estas nuevas características se han creado nuevas instrucciones y recursos.

Para resumir vamos a comentar las distintas versiones de JavaScript:

- **JavaScript 1.0:** nació con el Netscape 2.0 y soportaba una gran cantidad de instrucciones y funciones, casi todas las que existen ahora ya se introdujeron en

el primer estándar [Firefox, 2009a].

- **JavaScript 1.1:** es la versión de JavaScript que se diseñó con la llegada de los navegadores 3.0. Implementaba poco más que su anterior versión, ejemplo de ello es el tratamiento de imágenes dinámicamente y la creación de arreglos [Firefox, 2009a].

- **JavaScript 1.2:** que viene junto con la creación de los navegadores 4.0. Tiene como desventaja, que es algo distinta en plataformas Microsoft y Netscape, ya que ambos navegadores crecieron de distinto modo y estaban en competencia por el mercado [Firefox, 2009a].

- **JavaScript 1.3:** versión que implementan los navegadores 5.0. En esta versión se han limado algunas diferencias y asperezas entre los dos navegadores [Firefox, 2009a].

- **JavaScript 1.5:** versión actual que implementa Netscape 6 [Firefox, 2009a].

- **JavaScript 1.6:** presenta algunas características nuevas, tales como las especificadas a continuación: E4X(ECMAScript for XML), inclusión de nuevas operaciones sobre arreglos como el *indexOf()* y *lastOf()*, y *strings* genéricos [Firefox, 2009a].

- **JavaScript 1.7:** es una actualización de la versión 1.6 de JavaScript, donde se incluyen generadores e iteradores sobre estructuras tales como las listas, sentencias *let*. El soporte para JavaScript esta disponible a partir de Firefox 2, Beta 1 [Firefox, 2009a].

- **JavaScript 1.8:** está planificado para integrarse como parte de Gecko 1.9 (incorporado en Firefox 3). Es una actualización menos sustancial que la descrita en JavaScript 1.7, pero tiene algunas actualizaciones para comprobar el progreso hacia ECMAScript 4/JavaScript 2 e incluirá todas las nuevas características especificadas en JavaScript 1.6 y JavaScript 1.7 [Firefox, 2009a].

  Por su parte, Microsoft también ha evolucionado hasta presentar su versión 5.5 de JScript.

Capítulo 4

XUL

XUL son las siglas de Lenguaje XML de Interfaz de Usuario, (del inglés: XML User Interface Language), y es un lenguaje desarrollado para las interfaces de usuario (IU) de Netscape y Mozilla. Es parte del navegador de Internet Mozilla Firefox y otras aplicaciones relacionadas, así como de Gecko, que es el motor de renderizado del navegador. De hecho XUL es tan poderoso que la interfaz completa del navegador Mozilla está implementada en este lenguaje.
De forma similar al HTML, en XUL es posible crear una interfaz usando un lenguaje de marcado, definir la apariencia de esta interfaz con hojas de estilo CSS y usar JavaScript para manipular su comportamiento; a diferencia del HTML, XUL tiene un conjunto extenso de componentes gráficos utilizados para crear menús, barras de herramientas, cajas de texto, entre otros.
En otras palabras, XUL puede usarse para crear interfaces multiplataformas, multidispositivos y de forma ligera; que pueden ejecutarse con conexión a Internet o sin ella. Estas aplicaciones son fácilmente personalizables con texto alternativo, gráficos, y diseñ os, por lo que pueden ser fácilmente instaladas o traducidas para diversos mercados. [Firefox, 2009f] [WikiBooks, 2009a]

Algunos elementos que pueden ser creados son:

- Controles de entrada como los cuadros de texto y cajas de chequeo o verificación.
- Barra de herramientas con botones u otros contenidos.
- Menú en barras de menú o menú emergente.
- Pestañas de diálogos.
- árbol de información jerárquica o tabulada.
- Teclas de accesos directos.
- Barras de progreso. [Scribd, 2008]

Las siguientes figuras ilustran ejemplos de algunos componentes creados en XUL. Consultar las figuras 4.1, 4.2, 4.3 y 4.4.

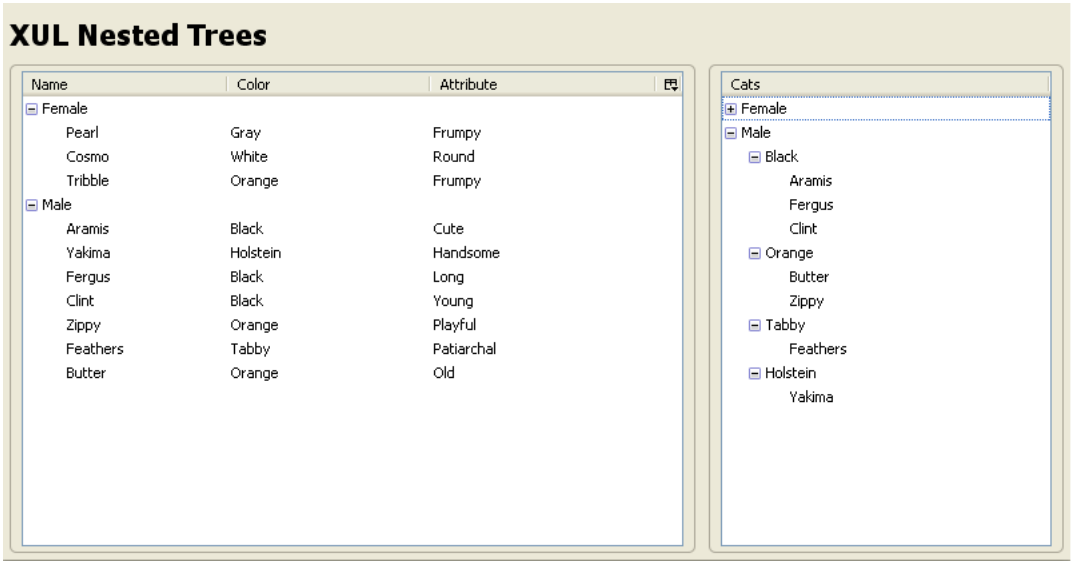


Figura 4.1: árbol de información jerárquica en XUL

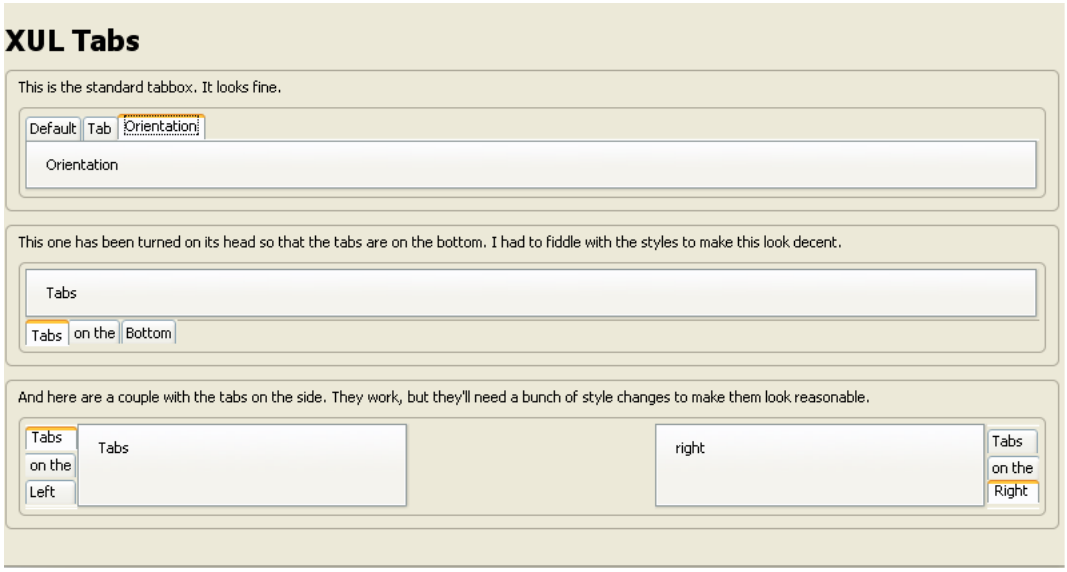


Figura 4.2: Pestañas de diálogos en XUL

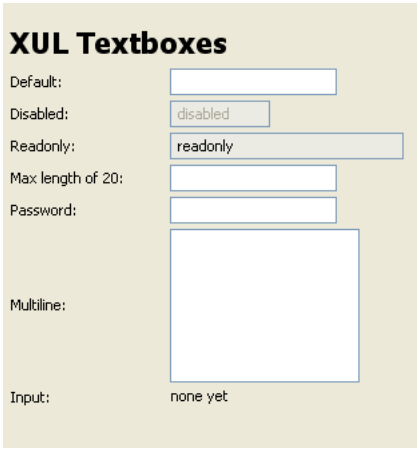


Figura 4.3: Cuadros de texto en XUL

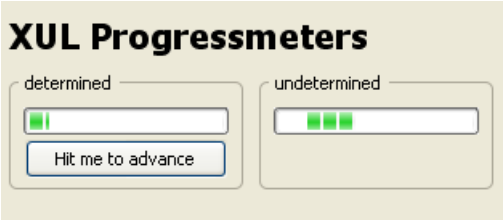


Figura 4.4: Barras de progreso en XUL

4.1. Ventajas de usar XUL

- La mayoría de las aplicaciones necesitan ser creadas usando características de una plataforma específica, lo que hace que su conversión a otras plataformas sea costosa en términos monetarios y de tiempo.

- Algunos usuarios querrían usar una aplicación en herramientas diferentes a una computadora tradicional, por ejemplo, dispositivos de mano. El lenguaje Java fue creado con ese propósito: ser multiplataforma y multidispositivo, pero la creación de interfaces de usuario en Java es una tarea difícil.
  XUL fue diseñado para crear interfaces de forma fácil y rápida, además está disponible en todas las versiones de sistemas operativos como Windows, Macintosh, Linux y Unix, pero el mayor inconveniente hasta el momento es que no es compatible con Internet Explorer.
  Para ilustrar su potencial, en este capítulo se desarrollarán algunos ejemplos. Se resalta la palabra potencial, debido a que las completas capacidades del XUL sobrepasan el alcance de este capítulo.

4.2. Características fundamentales

1. **Poderoso lenguaje de marcado basado en componentes:** el fin de XUL es construir aplicaciones multiplataforma, en constraste con DHTML el cual está dirigido a la construcción de páginas Web. Por esta razón, XUL es orientado hacia el desarrollo de artefactos de aplicación tales como ventanas, etiquetas, botones y campos de texto en lugar de páginas, niveles de encabezado e hipervínculos. De hecho, muchos desarrolladores invierten una gran cantidad de tiempo y esfuerzo en lograr buenos resultados en sus páginas web DHTML, pero a costa de la complejidad y el rendimiento, sin el soporte de ningún estándar [Firefox, 2007b].

2. **Basado en estándares existentes:** XUL es un lenguaje basado en el estándar de la especificación 1.0 de XML de la W3C. Las aplicaciones escritas en XUL están basadas en estándares tecnológicos adicionales de la W3C, en las cuales se resaltan: HTML 4.0, versiones 1 y 2 de Cascading Style Sheets (CSS), nivel 1 y 2 del Document Object Model (DOM), JavaScript 1.5, incluyendo la tercera edición de ECMA-262 (ECMAScript) y XML 1.0 [Firefox, 2007b].

3. **Plataforma portable:** igual que en HTML, XUL está diseñado para ser una plataforma neutral, haciendo que las aplicaciones sean fácilmente portables a todos los sistemas operativos en los cuales Mozilla se ejecuta. Desde que XUL provee una abstracción de componentes de interfaz de usuario, este se basa en la premisa de çodifícalo y ejecútalo donde desees". La interfaz de usuario para todas las aplicaciones con núcleo Mozilla (en las que se incluye Mozilla Firefox) están escritas en XUL con un único código base soportado en todas las plataformas Mozilla [Firefox, 2007b].

4. **Separación de la presentación de la lógica de negocios:** el mayor talón de aquiles de la mayoría de las aplicaciones web es el fuerte acoplamiento entre los elementos de interfaz de usuario y la lógica de la aplicación cliente. XUL provee una clara separación entre la definición de la aplicación cliente y la lógica de programación (contenido consistente de XUL, XBL, y JavaScript), la presentación (consiste de CSS e imágenes), y las etiquetas con textos específicos del idioma (consiste en DTDs y cadenas internacionalizadas definidas dentro de archivos .properties). El despliegue y la apariencia de las aplicaciones XUL puede ser alterada sin tomar en cuenta la definición y la lógica de la aplicación. Adicionalmente, la aplicación puede ser internacionalizada en diversos idiomas y regiones independientemente de su lógica o presentación. Este grado de separación conlleva a la creación de aplicaciones que son más fáciles de mantener por los desarrolladores. El flujo de trabajo resulta más fácil de coordinar que aplicaciones web basadas en HTML, con un menor impacto en la calidad y estabilidad del sistema.[Firefox, 2007b]

5. **Fácil personalización y localización:** Además de la separación que XUL ofrece entre la lógica de la aplicación, presentación e idiomas, es fácil de personalizar para diferentes vendedores o grupos de usuarios. Un desarrollador puede mantener una

base primaria de códigos, a la vez de personalizar el logo de la aplicación y enlazarlo con diferentes vistas para proporcionar diferentes interfaces a cada uno de los vendedores.
Una aplicación que es escrita y desplegada con una interfaz de usuario en Inglés puede ser traducida al Francés por el mismo vendedor. Mientras que la mayoría de los cambios son extensivos y afectan a la mayoria y no a toda la aplicación, estos están separados entre si permitiendo que el núcleo y la lógica de la aplicación puedan ser compartidas. [Firefox, 2007b]

4.3. Desarrollo en XUL

A continuación se proporciona una breve explicación acerca de la estructura de un archivo XUL y el manejo de eventos en el mencionado.

4.3.1. Estructura del archivo

XUL es XML, y como todos los archivos XML, debe empezar con la declaración estándar de XML. Actualmente XUL utiliza la versión 1.0.

Un archivo XUL debe tener una extensión *.xul*. El navegador Mozilla Firefox lo reconocerá automáticamente y sabrá qué hacer cuando el usuario lo ejecute.

Como archivos XML, las interfaces XUL, deben estar bien formadas, es decir cada elemento debe tener su etiqueta de apertura y cierre.

En el caso de ventanas en XUL, el elemento raíz para su construcción debe ser una etiqueta window.$< window > < /window >$.
Los documentos XML, son codificados para ser extensibles, o ser capaces de incorporar diferentes tipos de elementos, dependiendo de que tipo de información va a ser codificada en el documento.
Los documentos XHTML poseen elementos que representan elementos de documentos HTML tales como body, head, div, entre otros.

No existen hojas de estilo implícitas para XUL, debido a esto, siempre debe existir una declaración de hoja de estilo asociada.

Mozilla incluye una hoja de estilo estándar llamada "xul.css".
Aún cuando sea posible cargar las hojas de estilo que se deseen, la mejor práctica es usar inicialmente "xul.css", ver figura 4.5.

Nótese la referencia a *chrome*. El *chrome* es la parte de la aplicación que está fuera del área de contenido de una ventana. Barras de herramientas, barras de menú, barras de progreso y títulos de ventanas son ejemplos de elementos que son parte típica del *chrome*. *Chrome* es el término descriptivo para nombrar todos los elementos de una aplicación XUL.

Todos los documentos XML deben tener una declaración de espacio (único) de nombres (namespace). Los desarrolladores de XUL crearon un espacio de nombres que muestra el origen del nombre XUL.
La figura 4.5 muestra un ejemplo del archivo **"xul.css"**.

xul.css

```
1  <?xml version="1.0"?>
2  <?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
3  <window
4  id="theWindow"
5  title="The Window"
6  orient="horizontal"
7  width = "400"
8  height = "300"
9  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
10 </window>
```

Figura 4.5: Estructura del archivo xul.css

El siguiente elemento notorio es la etiqueta $<$ *window* $>$ (ventana).

Esta etiqueta es análoga a la etiqueta $<$ *body* $>$ (cuerpo) del HTML.
Todos los elementos estarán dentro de la etiqueta window. En la figura anterior, la etiqueta window tiene tres atributos, que son muy importantes.

- **id:** el atributo id es importante debido a que es la manera de identificar la ventana, para que el código (JavaScript) de la aplicación pueda hacer referencia a ella.
- **title:** aún cuando el atributo no es necesario, es una buena práctica dar un nombre descriptivo, el título se mostrará en la barra de título de la ventana.
- **orient:** le indica al navegador la dirección en la cual mostrar los elementos descritos en el archivo XUL (horizontal o vertical). Vertical es el valor por omisión, así por lo tanto no se especifica el atributo "horizontal", lo que indican que los componentes aparecerán uno sobre otro.

Existen otras reglas de sintaxis que es necesario tomar en cuenta:

1. Todos los eventos y atributos deben escribirse en minúsculas.

2. Todas las cadenas de texto deben estar entre comillas dobles (").
3. Todos los componentes interactivos deben tener etiquetas de apertura y cierre $< etiqueta > < /etiqueta >$ para estar bien formados.
4. Todos los atributos deben tener un valor. [WikiBooks, 2009a]

Ejemplos de componentes XUL

XUL posee una gran librería de componentes gráficos, entre los cuales se incluyen:

- Botones.
- Cajas de texto.
- Barras de progreso.
- Etiquetas.
- Listas desplegables.
- Menús contextuales.
- Barra de desplazamiento, entre otros.

A continuación se explicara la construcción de algunos de los componentes anteriormente mencionados:

Ejemplo básico

La figura 4.6 muestra el código fuente de una aplicación que imprime la frase Hola Mundo 3 veces. [WikiBooks, 2009b]

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window
 id="Hola"
 title="Ejemplo: hola mundo"
 orient="vertical"
 persist="screenX screenY width height"
 xmlns= "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
   <description style='font-size:24pt'>Hola mundo</description>
   <description value='Hola mundo' style='font-size:24pt'/>
   <label value = 'Hola mundo'  style='font-size:24pt'/>
</window>
```

Figura 4.6: Código fuente para imprimir Hola Mundo 3 veces

En la figura 4.6 se indicaron 3 maneras distintas de desplegar la frase Hola mundo, en el navegador y el resultado final puede ser visualizado en la figura 4.7

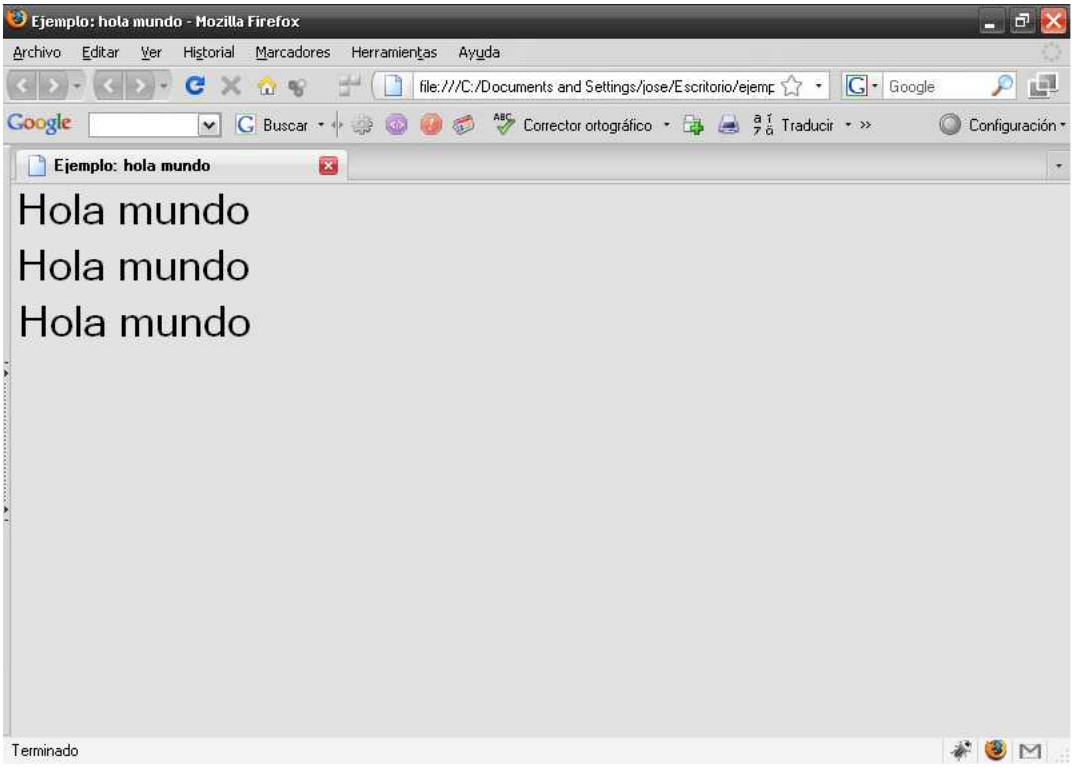


Figura 4.7: Hola mundo en el navegador

Cabe destacar las diferentes formas en las que la frase es mostrada: dos veces utilizando la etiqueta *description* y una vez utilizando la etiqueta *label*.

Ejemplo de contenedores

Para definir un componente contenedor XUL, se define una etiqueta similar a la etiqueta $< body >$ de HTML, para insertar componentes gráficos. La etiqueta $< box >$ sirve para encapsular otros elementos. Existen varios tipos de etiquetas box.
En el siguiente ejemplo se utilizarán las etiquetas $< hbox >$ (caja horizontal), $< vbox >$ (caja vertical), $< toolbox >$ (caja de herramientas) y $< tabbox >$ (caja con pestañas de selección).
$< hbox >$ y $< vbox >$ son iguales a la etiqueta $< window >$ pero con los atributos "orient" = "horizontal" y "orient" = "vertical" respectivamente. Estos dos elementos pueden tener otros componentes y a su vez pueden estar anidados. $< toolbox >$ puede usarse para crear barras de herramientas arriba o abajo de la ventana. $< tabbox >$, puede establecer hojas con lenguetas en las ventanas.

En la figura 4.8 se muestra el uso de los diversos tipos de etiquetas box.

Figura 4.8: Usos de etiquetas box

Es indispensable brindar una explicación acerca del funcionamiento del atributo flex (contracción de flexible), el cual constituye una forma de posicionar y asignar tamaño a los componentes gráficos de manera dinámica.

Este atributo puede tener valores entre 0 y 1 (por ejemplo 0.5). A mayor valor, el componente ocupará un mayor porcentaje de área cuando el contenedor donde se encuentra cambie de tamaño. Cabe mencionar que también existen los atributos "width" y "heigth", que definen la preferencia de tamañ o del componente.

Ejemplo de menús

Para el despliegue de componentes interactivos al estilo de tipo menú se utilizan las etiquetas:

- $< menubar > < /menubar >$
- $< menu > < /menu >$
- $< menupopup > < /menupopup >$
- $< menuitem > < /menuitem >$

En la figura 4.9 se despliega una barra con los menús Archivo, Editar y Ver.

Figura 4.9: Barra de menú en XUL

Al hacer click en cada una de las opciones anteriormente mencionadas a continuación:

En la figura 4.10 se despliega la opción Archivo.

Figura 4.10: Menú 1 desplegado en XUL

En la figura 4.11 se despliega la opción Editar.

Figura 4.11: Menú 2 desplegado en XUL

En la figura 4.12 se despliega la opción Ver.

Figura 4.12: Menú 3 desplegado en XUL

Ejemplo de Pestañas

En la figura 4.13 se hace referencia a la creación de 3 pestañas con diversos componentes interactivos.

Figura 4.13: Pestañas desplegadas en el navegador utilizando XUL

Las pestañas se definen con la etiqueta $< tab >$ y se les da un identificador (id) y un texto. A continuación se crea un conjunto de paneles, cada uno con contenido diferente. La primera pestaña (visualizada en la siguiente figura) muestra la aplicación de estilos dentro del código como en HTML.

Las siguientes pestañas tienen componentes con funcionalidad como lo son la barra de desplazamiento y el medidor de progreso funcionan.

En la figura 4.14 se muestra la barra de desplazamiento contenida en **Hoja02**

Figura 4.14: Barra de desplazamiento en XUL

En la figura 4.15 se muestra la barra de desplazamiento contenida en **Hoja03**

Figura 4.15: Medidor de progreso en XUL

En la siguiente sección se explica como enlazar la interfaz de usuario con la lógica de funcionamiento de la aplicación mediante el uso de manejadores de eventos.

En la figura 4.16 se muestran algunas de las instrucciones de como enlazar la interfaz de usuario con la lógica de funcionamiento de la aplicación.

```
<box width="200px">
  <button id="show" label="Show" default="true" oncommand="show();"/>
  <button id="hide" label="Hide" default="true" oncommand="hide();"/>
</box>
```

Figura 4.16: Manejo de eventos en XUL

4.3.2. Manejadores de eventos

XUL utiliza un esquema similar al de HTML para el manejo de eventos, a objeto de poder establecer una relación entre la interfaz de usuario y el núcleo de la aplicación.

En HTML, un manejador de evento se asocia con un elemento y se realiza alguna acción cuando se activa el manejador. La mayoría de los manejadores de HTML también están en XUL y hay algunos más que solo se encuentran en XUL.

A continuación se muestran algunos manejadores de eventos que son válidos por la mayoría de los elementos XUL

- **Blur:** Un elemento pierde el foco. **Manejador:** onblur.
- **Focus:** Un elemento pierde el foco. **Manejador:** onfocus.
- **KeyDown:** En un elemento que tiene el foco se presiona una tecla. **Manejador:** onkeydown.
- **KeyUp:** Una tecla deja de ser presionada sobre un elemento que posee el foco. **Manejador:** onkeyup.
- **KeyPress:** Se presiona y suelta una tecla o se mantiene presionada. **Manejador:** onkeypress.
- **MouseDown:** Se pulsa con un botón del mouse sobre un elemento. **Manejador:** onmousedown.
- **MouseUp:** Se deja de pulsar un botón del mouse sobre un elemento. **Manejador:** onmouseup.

- **MouseMove:** Se mueve el cursor del mouse sobre un elemento. **Manejador:** onmousemove.
- **MouseOut:**Se mueve el cursor fuera de un elemento. **Manejador:** onmouseout.
- **MouseOver:** Se mueve el cursor dentro de un elemento. **Manejador:** onmouseover.
- **Click:**Se presiona y suelta un botón del mouse sobre un elemento. **Manejador:** onclick.
- **DblClick:** Se realiza doble click con un botón del mouse sobre un elemento. **Manejador:** ondblclick.
- **Command:** Un elemento es activado. Existen diversas maneras en los elementos son activados. Por ejemplo, un botón es activado cuando se hace click sobre él o se presiona la tecla enter cuando tiene el foco; un menú es activado cuando se selecciona con el mouse o presionando la tecla de acceso directo. **Manejador:** oncommand.

En la figura 4.17 se visualiza un código XUL donde se muestra la utilización de diversos manejadores de eventos como: oncommand, onclick y ondblclick.

```
<vbox flex="1" style="overflow: auto">
 <groupbox>
  <caption label="These buttons tab oddly." />
  <hbox>
    <button flex="1" label="6" tabindex="6" oncommand="setText('tab-text','6');" />
    <button flex="1" label="3" tabindex="3" oncommand="setText('tab-text','3');" />
    <button flex="1" label="4" tabindex="4" oncommand="setText('tab-text','4');" />
    <button flex="1" label="2" tabindex="2" oncommand="setText('tab-text','2');" />
    <button flex="1" label="5" tabindex="5" oncommand="setText('tab-text','5');" />
    <button flex="1" label="1" tabindex="1" oncommand="setText('tab-text','1');" />
  </hbox>
  <hbox pack="center">
    <description  id="tab-text" value="(no input yet)"
    ondblclick="setText('tab-text', 'Doble clic')" />
  </hbox>
 </groupbox>
</vbox>
```

Figura 4.17: Código fuente para manejo de eventos básicos en XUL

El código anterior permite que al hacer clic o se presione la tecla enter sobre un botón se muestre una etiqueta con el nombre del botón que se presionó. Además, al hacer doble clic sobre el mensaje mostrado por el botón aparece el texto "Doble clic". Este comportamiento puede ser visualizado en las figuras mostradas a coninuación:

En la figura 4.18 se muestra la ventana inicial, sobre la cual no se ha realizado ningún tipo de evento.

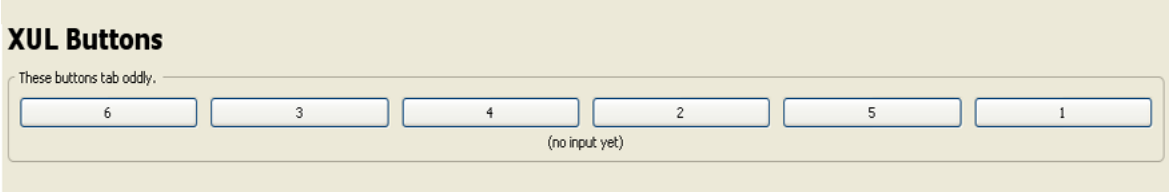


Figura 4.18: Botones XUL

En la figura 4.19 se muestra una etiqueta con el nombre del botón que se presionó.

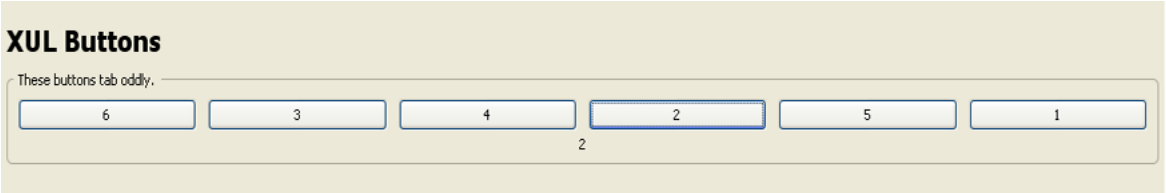


Figura 4.19: Visualización de la etiqueta con el nombre del botón presionado

En la figura 4.20 se muestra el texto "Doble clic" cuando se hace doble clic sobre el componente "label" ubicado debajo de los botones.

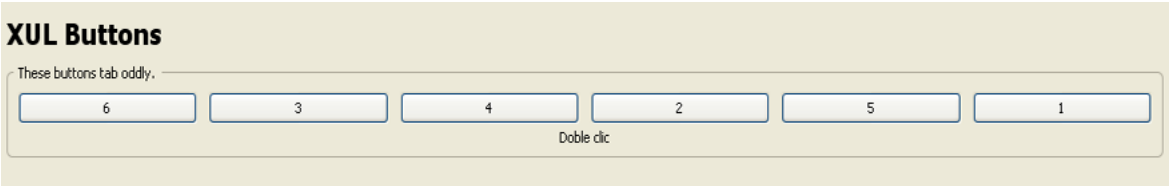


Figura 4.20: Texto "Doble clic" mostrado al hacer doble clic

Capítulo 5

Servicios Web

Son un conjunto de aplicaciones o tecnologías con capacidad para interoperar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Los servicios Web son usados para establecer la comunicación entre dos aplicaciones, ya que estos son una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones, las cuales pueden estar desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma.

Para el desarrollo de los servicios web existen diversas arquitecturas cuyas características pueden ser visualizadas en el cuadro 5.1.

Arquitectura simple	**Arquitectura compleja**
Interacción simple	Interacción compleja
Orientada al consumidor	Orientada a negocios
Cortos procesos de ejecución	Largos procesos de ejecución
No soporta procesos de negocios colaborativos	Soporte para procesos de negocios colaborativos
No garantiza seguridad y confiabilidad	Soporta seguridad y confiabilidad

Cuadro 5.1: Comparación de arquitecturas de servicios web

En particular se utilizará una arquitectura de servicios web *simple*.

Hoy en día existen diversas tecnologías para el desarrollo de servicios web, en este caso se hará énfasis en la tecnología REST, que será explicada con mayor detalle en la siguiente sección.

5.1. REST

Es una técnica de arquitectura de software para sistemas hipermedia distribuidos como la World Wide Web. REST es un término empleado por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP, durante la escritura de una tesis doctoral sobre la Web, en el año 2000.

La primera versión de REST fue desarrollada entre Octubre de 1994 y Agosto de 1995, principalmente como un medio para comunicar los conceptos de Web cuando se escribía la especificación de HTTP/1.0 y la propuesta de HTTP/1.1 inicial.

REST define un conjunto de principios arquitectónicos para diseñar servicios Web que están enfocados en la administración de recursos de un sistema, incluyendo la forma en que los recursos son direccionados y transferidos sobre HTTP por medio de una gran variedad de clientes escritos en diferentes lenguajes. REST ha emergido en los últimos años y se ha convertido en un modelo de diseño de servicios web predominante. De hecho, REST ha tenido un gran impacto en la Web hasta el punto de ir desplazando a los servicios web basados en la tecnología SOAP e interfaz WSDL, por ser más fácil de utilizar.

Principios de diseño básico

Una implementación concreta de un servicio web en REST sigue cuatro principios básicos de diseño:

- Emplea explícitamente métodos HTTP.
- Es sin estado.
- Expone la estructura de directorios como URIs.
- Transfiere archivos XML, JSON (JavaScript Object Notation), o ambos.

A continuación se procederá a explicar los principios ya mencionados.

Uso explícito de métodos HTTP

Una de las características claves de los servicios web REST es el uso explícito de métodos HTTP de manera que se siga el protocolo definido por el RFC 2616. HTTP GET, por ejemplo, es definido como un método para la obtención de recursos por parte de una aplicación cliente, o para ejecutar una consulta en el servidor y esperar una respuesta que incluya los recursos que concuerdan con la consulta realizada.

El principio básico de diseño de REST establece una relación uno a uno entre las operaciones crear, recuperar, modificar y eliminar (CRUD) y los métodos HTTP. De acuerdo a estas operaciones las relaciones serían las siguientes:

- Para crear un recurso en el servidor se emplea el método POST.
- Para recuperar un recurso se emplea el método GET.
- Para cambiar el estado de un recurso o modificarlo se utiliza el método PUT.
- Para quitar o eliminar un recurso en el servidor se emplea el método DELETE.

Esquema de trabajo sin estado

Los servicios web en REST necesitan ser escalables con la finalidad de mantener un buen rendimiento al aumentar los niveles de demanda del mismo. Por ejemplo, los servidores de clústeres con capacidades de balanceo de carga son normalmente organizados siguiendo una topología de servicios, lo cual permite que una solicitud pueda ser redireccionada de un servidor a otro cuando se necesite disminuir la sobrecarga y el tiempo de respuesta en la llamada a un servicio web. El uso de servidores intermedios para mejorar la escalabilidad requiere de clientes de servicios web REST que incluyan la información completa en la solicitud que se realizará, es decir, enviar toda la información necesaria con la solicitud de manera que los componentes de los servidores intermedios puedan redireccionar, enrutar y realizar balanceo de carga sin necesidad de almacenar localmente información de estado entre solicitudes.

Un servicio web REST (aplicación o cliente) debe incluir dentro de la cabecera y el cuerpo HTTP todos los parámetros, información de contexto, y los datos requeridos por los componentes del lado del servidor para que puedan generar una respuesta. Al trabajar en un esquema sin estado en los servicios web se mejora el rendimiento de estos, adicionalmente se simplifica el diseño y la implementación de los componentes del lado del servidor debido a que la ausencia de estado en el mismo, elimina la necesidad de sincronizar datos de sesión con una aplicación externa.

En la figura 5.1 se ilustra el diseño de un servicio con estado, donde la aplicación puede solicitar la información de la próxima página a mostrar, de un conjunto de resultados que han sido paginados. Asumiendo que el servicio mantiene almacenada la información de la página actual que está siendo mostrada, en este diseño con estado el servicio incrementa y almacena una variable 'siguientePag' con la cual puede responder la solicitud del cliente.

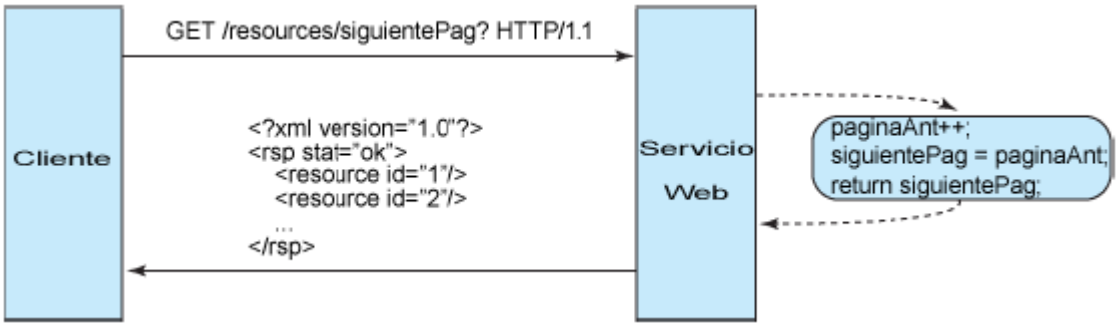


Figura 5.1: Ejemplo de un servicio web con estado

El esquema de trabajo visualizado en la figura anterior es complicado debido a la necesidad que tiene el servidor de almacenar y habilitar la sincronización de datos de la sesión. Además, la sincronización de la sesión añade sobrecarga al servidor, lo cual influye negativamente en su rendimiento.

Por otra parte, el diseño y desarrollo de componentes del lado del servidor que sean sin estado es menos complicado. En un servicio Web REST, el servidor es el responsable de generar la respuesta y de proveer una interfaz al cliente, de manera que le permita mantener el estado de la aplicación por si mismo. Por ejemplo, en una solicitud de página para un conjunto de resultados paginados, el cliente pudiera incluir el número de página actual como parámetro a la solicitud de la siguiente página.

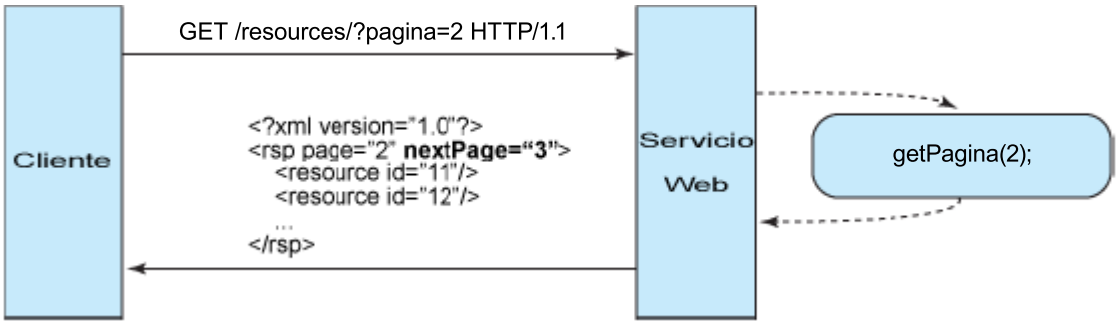


Figura 5.2: Ejemplo de un servicio web sin estado

En la figura 5.2 un servicio web sin estado genera una respuesta con el enlace a la siguiente página de resultados y le brinda al cliente la información del número de la página actual con la finalidad de que pueda almacenarla.

Exposición de la estructura de directorios como URIs

Desde el punto de vista de que las aplicaciones cliente solicitan recursos, el URI determina la ubicación del mismo. El URI de un servicio web REST debe ser intuitivo, de manera que pueda ser fácil de adivinar. En fin, la estructura de un URI debería ser sencilla, predecible y entendible.

Transferencia de archivos XML, JSON, o ambos

Típicamente la representación de un recurso refleja el estado actual del recurso y sus atributos, al momento que una aplicación cliente lo solicite. En adición, los objetos en el modelo de datos de una aplicación están relacionados de alguna manera, y las relaciones entre los objetos del modelo de datos (recursos) estaría reflejada en la manera en que ellos son representados a la hora de ser transferidos a la aplicación cliente.

En la figura 5.3 se ilustra un ejemplo del formato de respuesta XML de un servicio web REST representando un objeto "discusion".

```
<?xml version="1.0"?>
<discussion date="{date}" topic="{topic}">
 <comment>{comment}</comment>
 <replies>
  <reply from="joe@mail.com" href="/discussion/topics/{topic}/joe" />
  <reply from="bob@mail.com" href="/discussion/topics/{topic}/bob" />
 </replies>
</discussion>
```

Figura 5.3: Ejemplo de una respuesta XML de un servicio web REST

No obstante, para que el cliente puede reconocer el tipo de formato devuelto por el servicio web, este último debe añadir esta información MIME (Multiporpouse Internet Mail Extension) en la cabecera HTTP de la respuesta. Los tipos MIME soportados por REST se muestran en el cuadro 5.2. [Alex, 2007] [Costello, 2002] [Shailesh K. Mishra, 2007]

Tipo MIME	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xml+xhtml

Cuadro 5.2: Tipos MIME soportados por REST

Capítulo 6

Estadísticas en la Web

Hoy en día, la utilización de la Internet como medio de comunicación global entre los usuarios que circulan por la red es cada vez es mayor.
No obstante, muchos sitios web carecen de mecanismos que les permitan obtener información acerca de los usuarios que se encuentran conectados. Es por ello que herramientas como los visores de estadisticas web, constituyen una solución para entes públicos, privados y particulares, que les permita obtener información de interés, ya sea para generar ventajas competitivas, incursionar en mercados internacionales, rediseñar un sitio web desde el punto de vista estático, funcional, e incluso su arquitectura, de forma automatizada.

Entre los visores de estadísticas web más conocidos en la red actualmente se pueden mencionar:

Servicios gratuitos

- Google Analytics [maestros del web, 2007]
- Clicky [maestros del web, 2007]
- StatCounter [maestros del web, 2007]
- ShinyStat [maestros del web, 2007]
- phpMyVisites [maestros del web, 2007]
- BBClone [maestros del web, 2007]
- EstadisticasGratis [maestros del web, 2007]
- Weboscope [maestros del web, 2007]

A continuación se explicarán algunos de dichos servicios:

Google Analytics: Es un servicio de estadísticas para sitios web. Surge a raíz de la compra de la empresa Urchin (hasta entonces la mayor compañía de análisis estadístico de páginas web) de Google, y ofrece una gran variedad de funciones y ventajas para cualquier tipo de usuarios, ya sean ejecutivos experimentados, profesionales del marketing, publicidad, propietarios de sitios web, desarrolladores de contenidos, y webmasters; de las cuales podemos mencionar: [Analytics, 2009]

Implementación rápida: Con sólo copiar el código de seguimiento de Google Analytics en cada una de las páginas de su sitio web, el mismo se inicia de forma inmediata (ver figura 6.1) [Analytics, 2009].

Figura 6.1: Implementación rápida

Comparativa de campañas y palabras clave: Compare y supervise todos sus anuncios, boletines informativos por correo electrónico, campañas de afiliados, referencias, vínculos de pago y palabras claves en Google y otros motores de búsqueda (ver figura 6.2) [Analytics, 2009].

Figura 6.2: Comparativa de campañas y palabras claves

Paneles personalizados: No busque más en los informes. Coloque toda la información que necesita en un panel personalizado que podrá enviar a otros usuarios por correo electrónico (ver figura 6.3) [Analytics, 2009].

Figura 6.3: Paneles personalizados

Integración con AdWords: Compre palabras clave de AdWords de Google y utilice Google Analytics para averiguar cuáles son las más rentables para usted (ver figura 6.4) [Analytics, 2009].

Figura 6.4: Integración con AdWords

Búsqueda interna en sitios: Descubra de que forma los usuarios realizan búsquedas en su sitio, qué buscan y dónde culminan dichas búsquedas (ver figura 6.5) [Analytics, 2009].

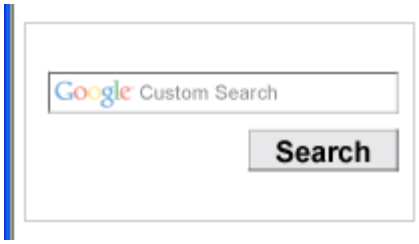


Figura 6.5: Búsqueda interna en sitios

Comparativas: Averigue si los indicadores de uso de su sitio son superiores o inferiores a los de su sector. Las comparativas permiten equiparar sus indicadores claves con indicadores de rendimiento global; además de respetar la confidencialidad de sus datos (ver figura 6.6) [Analytics, 2009].

Figura 6.6: Comparativas

Barra deslizante de tendencia y fecha: Compare diversas tendencias de su sitio web por períodos de tiempo, y seleccione intervalos de fechas sin perder de vista las tendencias a largo plazo (ver figura 6.7) [Analytics, 2009].

Figura 6.7: Barra deslizante de tendencia y fecha

Seguimiento de comercio electrónico: Permite realizar un seguimiento de sus transacciones mediante el uso campañas y palabras claves; obtener cifras de latencia y fidelización, así como el identitificar sus fuentes de ingresos (ver figura 6.8) [Analytics, 2009].

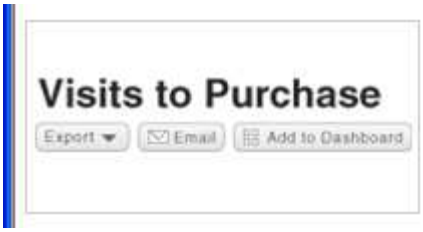


Figura 6.8: Seguimiento de comercio electrónico

Visualización de redireccionamiento: Averigue hacia dónde se dirigen sus clientes potenciales (ver figura 6.9) [Analytics, 2009].

Figura 6.9: Visualización de redireccionamiento

Superposición del sitio: Acceda a información sobre el tráfico y sobre las conversiones de cada vínculo al visualizar su sitio (sin descarga previa) (ver figura 6.10) [Analytics, 2009].

Figura 6.10: Superposición del sitio

Informes por correo electrónico: Programe o envíe informes personalizados por correo electrónico que contengan exactamente la información que desee compartir (ver figura 6.11) [Analytics, 2009].

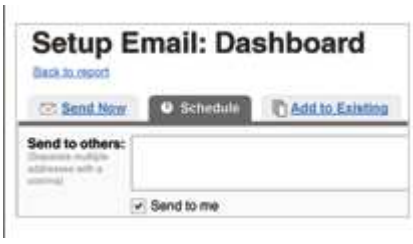


Figura 6.11: Informes por correo electrónico

Orientación geográfica: Averigue la procedencia de los usuarios que visitan su sitio web e identifique los mercados geográficos más lucrativos (ver figura 6.12) [Analytics, 2009].

Figura 6.12: Orientación geográfica

Tecnologías utilizadas: Código Javascript, que puede ser incluido en las páginas que se deseen analizar, y el cual permite cargar algunos archivos desde los servidores de Google para monitorizar y enviar información al servidor Google, que posteriormente es almacenada en la cuenta del usuario correspondiente [Analytics, 2009].

Posee una interfaz de reportes gráficos desarrollados en Adobe Flash.

EstadisticasGratis: Es un servicio de estadísticas que le permite analizar en detalle las estadísticas de su página web (Análisis de tráfico, marketing, perfil de usuarios) al instante y desde cualquier sitio [EstadisticasGratis, 2004b] .

Entre las funcionalidades que posee podemos hacer énfasis en las siguientes:

Figura 6.13: EstadisticasGratis - Tipos de Reportes

Dentro de las funcionalidades asociadas al *Tráfico Horario* [EstadisticasGratis, 2004d] podemos observar lo mostrado a continuación (ver figura 6.14 y 6.15):

Figura 6.14: EstadisticasGratis - Páginas vistas por hora

Figura 6.15: EstadisticasGratis - Visitas por hora

Ahora visualizaremos lo que la opción *Tráfico Diario* [EstadisticasGratis, 2004c] ofrece (ver figura 6.16 y 6.17):

Figura 6.16: EstadisticasGratis - Páginas vistas por día

Figura 6.17: EstadisticasGratis - Páginas visitadas por día

Entre las funcionalidades contempladas en la opción *Tráfico Mensual* [EstadisticasGratis, 2004c] podemos visualizar las mostradas en los gráficos posteriores (ver figura 6.18 y 6.19):

Figura 6.18: EstadisticasGratis - Páginas visitadas por mes

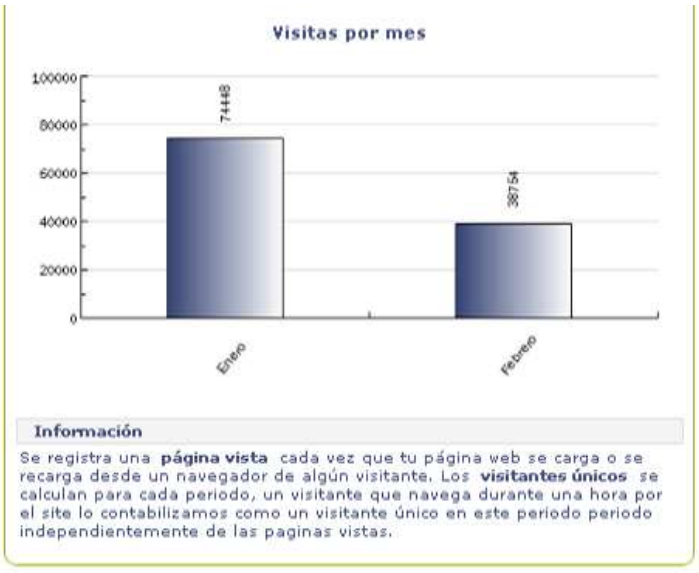


Figura 6.19: EstadisticasGratis - Páginas vistas por mes

Haciendo click en la opción *Calidad del color* de Perfiles de Usuario, [EstadisticasGratis, 2004a] se puede observar lo siguiente (ver figura 6.20):

Figura 6.20: EstadisticasGratis - Calidad del Color

Servicios pagos

- XiTi Web Analytics
- Web-stat
- 3DStats

A continuación se proporciona una breve explicación acerca del servicio 3DStats.

3DStats es un portal web que ofrece un servicio para analizar patrones de tráfico y perfiles de usuario en su sitio web, y visualizar información en tiempo real sin la necesidad de instalar ningún software. [3DStats, 2004a]
Permite realizar un seguimiento del número de visitantes, solicitudes y órdenes para cualquier campaña publicitaria en linea que usted dirija.

Cada vez que un usuario accede a su sitio web, se recolecta información que será utilizada para proveerle detalles estadísticos en tiempo real del mismo. Dicho código es fácil de añadir a la página web existente, sólo se trata de copiar y pegar código HTML, y el cual no afecta el rendimiento de la misma. 3DStats soporta SSL.

Las estadísticas de su sitio web están disponibles en cualquier momento y son protegidas por una contraseña (la clave que 3DStats le proporciona).
Si no tiene tiempo para instalar el código de 3DStats, en su sitio web, un experto en 3DStats puede ayudarlo. El precio inicial por esto es de 69 dólares. [3DStats, 2004b]

A continuación un ejemplo de lo que la herramienta ya mencionada ofrece (ver figuras 6.21, 6.22, 6.23, 6.24, 6.25, 6.26 y 6.27) [3DStats, 2004c]:

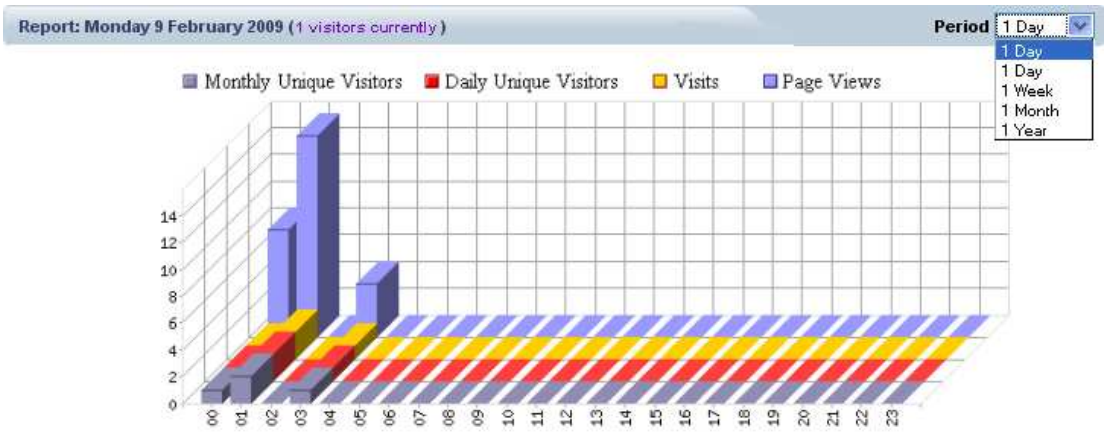


Figura 6.21: 3dStats - Reporte general

Figura 6.22: 3dStats - Búsqueda personalizada

Visits - Page Views - Unique Visitors

	Per Hour	Estimate	Today	Yesterday	Week	Month	Month -1
Visits:	1.00	29.3	**4**	44	4	325	1421
Page Views :	6.72	129.4	**27**	321	27	1756	6689
Daily Unique Visitors :	1.00	28.0	**4**	42	4	314	1355
Monthly Unique Visitors :	1.00	28.0	**4**	42	4	301	1276

Figura 6.23: 3dStats - Reporte por visitas, número de páginas y visitantes

Best Day

	Date	Best Day	Today Estimate	Comparison
Visits:	Wednesday 12 September 2001	2390	29.3	-98.77%
Page Views:	Monday 23 February 2004	12638	129.4	-98.98%
Daily Unique Visitors:	Wednesday 12 September 2001	2381	28.0	-98.82%
Monthly Unique Visitors:	Monday 23 February 2004	2305	28.0	-98.79%

Figura 6.24: 3dStats - Reporte por Mejor día

Quick Evolution

	Visits	Today	Page Views	Today	Daily Unique	Today	Monthly Unique	Today
Sunday 8 February 2009	44	-33.33%	321	-59.70%	42	-33.33%	42	-33.33%
Monday 2 February 2009	39	-24.79%	173	-25.23%	37	-24.32%	35	-20.00%
Monday 26 January 2009	48	-38.89%	246	-47.42%	44	-36.36%	43	-34.88%
Monday 19 January 2009	54	-45.68%	257	-49.67%	52	-46.15%	50	-44.00%
Monday 12 January 2009	44	-33.33%	187	-30.82%	43	-34.88%	40	-30.00%

Figura 6.25: 3dStats - Evolución de visitas en el tiempo

Last 40 Visits

	Hostname	PV	Last	Os	More
Mon 09 Feb 2009,03:53:59	ppp-117-156.33-151.iol.it MILANO http://www.google.it/search?hl=1C1CHMY_itIT306IT306&sourceid=chrom...	4	New	Win XP	
Mon 09 Feb 2009,01:35:08	p5b27d2bc.dip.t-dialin.net ESSEN http://www.google.de/search?hl=de&q=hopkins+fbi%2Bxp&btnG=Google-Su...	10	New	Win XP	
Mon 09 Feb 2009,01:19:09	cpe-24-243-115-212.stx.res.rr.com DOTHAN http://www.happypenguin.org/show?Hopkins%20FBI	5	New	Linux	
Mon 09 Feb 2009,00:12:32	cuscon56657.tstt.net.tt http://www.google.com/search?hl=en&q=f.b.i.+website&aq=0&oq=f.b.i.	8	New	Win XP	
Sun 08 Feb 2009,23:46:47	d54c49f06.access.telenet.be MECHELEN http://www.libsdl.org/games.php?start=200&order=name&category=-1&co...	4	New	Linux	
Sun 08 Feb 2009,23:31:24	168-226-11-240.speedy.com.ar BUENOS AIRES http://www.uruguaysoft.com/Hopkins+FBI-14163.html	3	New	Win XP	
Sun 08 Feb 2009,23:20:45	86.104.138.20 BUCHAREST http://www.google.ro/search?client=firefox-a&rls=org.mozilla%3Aen-U...	6	New	Win XP	
Sun 08 Feb 2009,23:14:14	bhx154.neoplus.adsl.tpnet.pl POZNAN http://ourworld.compuserve.com/homepages/jchristophe/hopkins.htm	6	New	Win XP	
Sun 08 Feb 2009,22:43:05	dynamic-ip-19015810614.cable.net.co SANTAFé DE BOGOTá http://www.uruguaysoft.com/Hopkins+FBI-14163.html	6	New	Win XP	
Sun 08 Feb 2009,22:36:50	208-85.agr.adsl.eastport.ru ZELENOGRAD No referring Url	3	New	MacOS	

Figura 6.26: 3dStats - últimas 40 visitas

Today (Yesterday results are in grey)

	Visits		Page Views		D.Unique		M.Unique		Hours %
00:00 - 00:59	1	2	8	23	1	2	1	2	25.00%
01:00 - 01:59	2	1	15	9	2	1	2	1	50.00%
02:00 - 02:59	0	2	0	30	0	2	0	2	0.00%
03:00 - 03:59	1	1	4	5	1	1	1	1	25.00%
04:00 - 04:59	0	1	0	3	0	1	0	1	0.00%
05:00 - 05:59	0	0	0	0	0	0	0	0	0.00%
06:00 - 06:59	0	1	0	12	0	1	0	1	0.00%
07:00 - 07:59	0	1	0	2	0	1	0	1	0.00%
08:00 - 08:59	0	1	0	3	0	1	0	1	0.00%
09:00 - 09:59	0	0	0	0	0	0	0	0	0.00%
10:00 - 10:59	0	2	0	5	0	2	0	2	0.00%
11:00 - 11:59	0	3	0	25	0	3	0	3	0.00%
12:00 - 12:59	0	1	0	1	0	1	0	1	0.00%
13:00 - 13:59	0	5	0	53	0	5	0	5	0.00%
14:00 - 14:59	0	3	0	14	0	3	0	3	0.00%

Figura 6.27: 3dStats - Reporte del día

6.1. Medidores estadísticos

En la presente sección se hará énfasis en algunos de los medidores empleados para la recolección de estadísticas en la Web, los cuales son enumerados a continuación:

1. **Tráfico web:** corresponde a la recolección de la cantidad de visitas realizadas en una página durante un período de tiempo específico, el cual puede ser clasificado en meses, días, horas, etc [3DStats, 2004c].

2. **Tecnología de usuarios:** se refiere a la recolección de datos tales como: información del sistema operativo del computador desde el cual una página es visitada, software del navegador, calidad de color, resolución de pantalla, etc [3DStats, 2004c].

3. **Ubicación geográfica:** hace referencia a la recolección de datos correspondientes al país de procedencia del visitante de una página, así como el idioma del mismo [3DStats, 2004c].

4. **Lugar de procedencia:** constituye la recolección de datos haciendo referencia a la URL desde donde llegan las visitas a una página [3DStats, 2004c].

6.2. Análisis de datos estadísticos en la Web

En esta sección se analizarán algunas causas del por qué muchos visores de estadísticas web arrojan resultados erróneos, así como el proporcionar elementos que permitan determinar la fiabilidad de un servicio de estadísticas web.

La gran mayoría de servicios gratuitos de estadísticas de acceso web muestran una visión distorsionada acerca de lo que realmente sucede en un sitio web. Sin embargo existe la tecnología necesaria para realizar análisis de tráfico.

No obstante, sólo suelen ser utilizadas por los servicios de estadísticas más costosos, por lo que muchos webmasters optan por los planes gratuitos, y quienes suelen contentarse con reportes y gráficas que sólo reflejan una parte de lo que en realidad está ocurriendo en su sitio web (en el mejor de los casos) ya que muchos servicios no reportan cifras reales.

6.2.1. Análisis de visitas de un sitio web

La mayoría de los sitios web están compuestos de varias páginas, y todas ellas tienen la posibilidad de recibir una visita sin necesidad de que el visitante ingrese en la página de inicio. El tipo de visita directo (cuando el visitante ingresa en la página principal de la aplicación), tiende a incrementarse cuando las páginas han sido indexadas por buscadores (que suelen mostrar subpáginas de diferentes sitios en los resultados de sus búsquedas). También es posible que desde otros sitios web existan enlaces hacia subpáginas específicas de su sitio web, y ésta constituye otra fuente de visitas que no pasa por la página de inicio.

Los sistemas de estadísticas que se basan en la inclusión de un botón en nuestra página de inicio sólo cuentan las visitas realizadas en la página principal, y por lo tanto no proporcionan información de las actividades que se realizan en el resto de las páginas del sitio web.

6.2.2. Análisis de accesos o visitas a un sitio web

El ***acceso***, es una apertura de página, no importa en qué condiciones: si se ingresa en un sitio web y se hace clic 9 veces en el botón "recargar" del navegador, esto genera 10 accesos a la página (un acceso inicial al ingresar en la página, más 9 accesos que genera recargándola). Posiblemente el webmaster vea el reporte y concluya: "Acaban de ingresar diez personas".

La ***visita***, se define como la entrada de un usuario de forma individualizada a una página, independientemente de cuantas veces se abrió o recargó en el navegador.

Es muy común que una persona que visita un sitio web lo recorra abriendo varias veces determinadas páginas (para volver a acceder a un menú, o una lista de enlaces, por ejemplo).

Cuando se maneja el concepto de "visita", también debe manejarse el concepto de *timeout de visita.*

El timeout de visita, es el tiempo de inactividad que debe transcurrir para que se considere que una visita ha concluído. Dicho timeout puede ser establecido por el webmaster. Una vez transcurrido este tiempo de inactividad, si el visitante vuelve a abrir la página, esto se considera como una nueva visita. En fin, es posible que un usuario nos visite varias veces al día.

Sólo se debe tener la precaución de determinar mediante el timeout si una nueva apertura de página es parte de una visita en curso, o en cambio el usuario abandona la página, y después regresa generando una nueva visita. Para terminar de ilustrar el concepto: imaginemos la situación que se generaría en una máquina instalada en un cibercafé, desde donde un usuario visitó nuestra página. Si al cabo de un rato esa misma máquina es ocupada por un nuevo usuario que también abre nuestra página, no hay ninguna razón para dejar de contabilizarlo como visita.

6.2.3. Clientes detrás de proxys y routers NAT

Un servidor proxy es un dispositivo que permite acelerar la conexión a Internet de sus clientes (las computadoras que están configuradas para navegar haciendo uso de sus servicios). El servidor proxy guarda en la memoria caché las páginas Web a las que acceden los sistemas clientes de la red durante un cierto tiempo.

Cuando un sistema solicita la misma página web, el servidor proxy utiliza la información guardada en la memoria caché en lugar de recuperarla del proveedor de contenidos. De esta forma, se accede con más rapidez a las páginas Web. Adicionalmente constituyen un mecanismo de seguridad implementado por los administradores de la red en un entorno de Intranet para desactivar el acceso o filtrar las solicitudes de contenido para ciertas sedes Web consideradas ofensivas o dañinas para la red y los usuarios.
Por ejemplo: en una escuela un proxy permite que el administrador bloquee el acceso a páginas para adultos, logrando al mismo tiempo una gran calidad de navegación a pesar de poseer una línea de baja velocidad para atender decenas de computadores en el aula de informática.
El problema radica en que todas las peticiones a Internet parecen salir de una única máquina (el proxy), que esconde la actividad individual de las máquinas que se encuentran detrás.

A su vez existen dos tipos de proxy:

- **Los normales:** este tipo de proxy añade en la cabecera HTTP información de su condición de proxy y qué máquina está llevando a cabo el acceso.
- **Los anónimos:** este tipo de proxy esconde a Internet su condición de proxy.

Por otra parte, el NAT (Network Address Translation) es implementado mediante routers (complejos dispositivos encaminadores, que constituyen el soporte de las comunicaciones en Internet) y es una técnica que permite a un proveedor de acceso a Internet lograr que una gran cantidad de clientes naveguen usando una misma dirección IP (Internet Protocol). Para las empresas que cuentan con pocas direcciones IP es una solución ideal: las direcciones IP son un recurso cada vez más escaso, por lo que la técnica NAT se usa cada vez más. Existen pequeñas ciudades y poblaciones enteras que se conectan a Internet mediante un NAT configurado por su compañía de telecomunicaciones, usando unas pocas direcciones IP para la conexión de miles de clientes. Desde el punto de vista de un sistema de estadísticas, todas esas máquinas son en realidad vistas como si se tratara de un solo cliente (lo que le lleva a tener reportes de tráfico completamente alejados de la realidad).

Existe una tecnología capaz de individualizar los clientes que nos visitan desde atrás de un NAT o un proxy anónimo: el "client footprint", que consiste en el análisis de un paquete de características particulares de su máquina (la "huella" de la máquina), y que le permite saber qué máquinas distintas están generando actividad en su sitio web a pesar de venir desde una misma IP. Esta técnica es usada por muy pocos sistemas de estadísticas web.

6.2.4. Visitas desde .COM .NET .EDU y .ORG

En los reportes sobre el origen geográfico de las visitas puede observar cuántos accesos ha tenido desde España, México, Argentina, etc. Pero posiblemente verá entre los países, que lo han visitado desde "EEUU Comercial (.com)", o desde ".net y .org".

Dificilmente las visitas que dicen ser de "EEUU Comercial(.com)" realmente provengan de Estados Unidos, ya que el dominio COM puede estar asociado a una máquina en cualquier lugar del mundo. Lo mismo para NET, EDU y ORG.
Entonces ¿por qué el sistema de estadísticas no le proporciona el lugar geográfico real de la visita, en lugar de decirle que es desde una red COM?. Porque se está utilizando una tecnología fácil pero inapropiada: la resolución DNS reversa.

En la resolución DNS reversa, cuando llega una visita a un sitio web, se obtendrá la IP de la máquina que realizó la petición (ejemplo: 200.96.85.14). Entonces el sistema de estadísticas hace una búsqueda DNS reversa para esa IP, con el fin de obtener el

nombre de la máquina.

Un método serio para determinar la situación geográfica de una dirección IP es mediante una base de datos de direcciones IP repartidas por países. Es el método que usan tecnologías tales como GeoIP (`http://www.maxmind.com`), o directorios de internet como ip-to-country (`http://ip-to-country.webhosting.info/`) entre otros.

6.2.5. Análisis de logs

Otra herramienta para obtener datos estadísticos en los sitios web, es el análisis de logs del servidor web.

Hoy en día, existen programas que analizan los archivos de registro de actividad generados por el propio servidor. Estos registros son sin duda la fuente de información más fiel acerca de qué es lo que ocurre en un sitio web. Pero su uso resulta engorroso, y la información que se obtiene es incompleta, por ejemplo, no se especifican las capacidades del navegador en cuanto a extensiones, y no pueden identificar clientes detrás de proxys o NATs.

Parte II

Marco Aplicativo

Capítulo 7

Marco Aplicativo

En el siguiente capítulo se describen los aspectos más importantes del proceso de desarrollo Programación Extrema(XP) y la adaptación del proceso de implementación, para el desarrollo de la herramienta que se desea construir.

7.1. Adaptación del proceso de desarrollo XP

eXtremme Programming (XP)[xprogramming, 2001] es un proceso de desarrollo ágil basado en la simplicidad del software, la comunicación directa entre los integrantes involucrados en el proceso, el seguimiento del desarrollo y la reutilización de código, no enfocandose en la documentación, sino en los requerimientos comunicados por el cliente.

El proceso de desarrollo XP está basado en iteraciones que permiten ir desarrollando entregables del sistema hasta obtener el producto final. Inicialmente se realiza un análisis de requerimientos y durante el resto del proceso de desarrollo se mantiene una retroalimentación progresiva entre los actores involucrados con la finalidad de monitorear el progreso del desarrollo de la aplicación, detectar cambios y realizar las correcciones pertinentes; que los desarrolladores involucrados incrementen sus habilidades, y el agilizar los procesos de construcción del software deseado. Durante el desarrollo de la presente herramienta, las iteraciones estan basadas en lapsos de tiempo de cada par de semanas. A fin de no extender el presente documento cada iteración es agrupada en lapsos de 4 semanas.

7.2. Actores y responsabilidades

Existen diferentes actores y responsabilidades en XP para diferentes actividades y propósitos durante el proceso. Para el presente trabajo especial de grado los roles existentes son:

- a)**Programador**: quien es el responsable de las decisiones técnicas y de construir el sistema teniendo como labores: Analizar, diseñar, implementar y realizar las pruebas del sistema ya mencionado.
- b)**Cliente**: quien es parte del equipo, y determina qué construir y cuando, elabora pruebas funcionales para determinar cuando esta completo un requerimiento determinado o no.

En el cuadro 7.1 se muestran las personas encargadas para cada rol.

Roles existentes durante el desarrollo

Rol	Nombre
Programador	Br. Angel Gil
Programador	Br. José Sanchez
Cliente	Prof. Sergio Rivas

Cuadro 7.1: Roles existentes durante el desarrollo

7.3. Metáfora del sistema

Es un lenguaje que se utiliza para describir la arquitectura de una aplicación, con la finalidad de ayudar a facilitar la comunicación entre los desarrolladores y el cliente.

A continuación se describen los módulos principales que conforman el software a desarrollar:

7.3.1. Aplicación Cliente

- a) Módulo **Visor de estadísticas Web**, creado para visualizar las diversas opciones de configuración del visor de estadísticas web, e información acerca de los autores de la aplicación. Dicho módulo puede ser desplegado mediante el uso del click derecho del mouse, o a través de la barra de herramientas del navegador, o mediante el panel de extensiones que provee el navegador Mozilla Firefox.
- b) Módulo **Parseo de datos**, se encarga de obtener información básica acerca de cada uno de los proyectos existentes en el servidor de la aplicación, y que posteriormente son cargados en la interfaz de opciones del visor de estadisticas web.
- c) Módulo **Recolección de estadisticas**, analiza el comportamiento del usuario en la web, mediante diversos parámetros ya establecidos en el servidor de la aplicación.

- d) Módulo **Envío de datos** Encargado de enviar la información obtenida acerca del comportamiento del usuario en la web al servidor de la aplicación.

7.3.2. Aplicación Servidor

- a)Módulo **Capa de acceso a datos:** encargada de permitir un acceso transparente a las capas superiores y a la fuente de datos del sistema, en nuestro caso la base de datos (BD) desplegada en el Sistema Manejador de Base de Datos (SMBD) MySQL. Dicha capa esta conformada de los siguientes submódulos:
  - a.1) Submódulo **Acceso/Gestión de entidades:** es el conjunto de objetos que conforman las entidades de persistencia y permiten realizar las operaciones CRUD en las respectivas entidades de la BD. Lo anterior se implementa empleando el patrón DAO y fáricas de objetos propuesta por Hibernate [Gavin king, 2009].
  - a.2) Submódulo **Interfaz de conexión con BD:** constituye el grupo de objetos que permiten lograr una conexión transparente con la base de datos, esto se logra utilizando las tecnologías JPA y Hibernate.
- b)Módulo **Capa de procesamiento de datos:** encargada de permitir la carga/almacenamiento de estadísticas y de generar los objetos correpondientes para cada uno de los reportes del sistema. Se encuentra conformado por los sub-módulos especificados a continuación:
  - b.1) Submódulo **Carga de estadísticas:** conformado por un conjunto de objetos que permiten leer y validar los documentos XMLs con los datos estadísticos enviados por las extensiones clientes, a través de la tecnología de carga/validación y mapeo de datos XMLs a objetos de JAXB [Oracle, 2010]. Por otra parte, dichos datos son cargados en las entidades de la BD por medio de la capa de acceso a datos.
  - b.2) Submódulo **Generación de reportes:** constituido por clases de utilidades, fábricas y pool de objetos que definen los diversos reportes que pueden ser generados en el sistema. A través de las entidades mencionadas anteriormente, se permite la generación de los datos de los reportes, utilizando el framework de colecciones (listas) de Java, permitiendo de esta manera que la capa web pueda utilizar el presente módulo para generar la presentación de los reportes por medio de ampliaciones a través del patrón Decorator [Erich Gamma, 1994].
- c)Módulo **Capa de presentación y despliegue de servicios web:** conformado por los siguientes sub-módulos:

- c.1) Submódulo **Interfaz Gráfica de Usuario Web:** representa las clases y objetos que definen las páginas visualizadas en el sistema y el conjunto de controladores y modelos de presentación de las diversas secciones para la gestión de los proyectos, sesiones de proyectos y visualización de los reportes estadísticos. Lo anteriormente expuesto es contruido utilizando el framework JavaServer Faces (JSF) [Chris Chalk, 2007].

  Adicionalmente diversos componentes del presente módulo emplean la librería de componentes para JSF denominada PrimeFaces [PrimeFaces, 2009].

- c.2) Submódulo **Despliegue de servicios web REST:** conformado por un conjunto de clases que representan los recursos (por implementación de servicios web REST) [Fielding, 2000] definidos en el sistema y conformar la interfaz de comunicación entre la extensión cliente y el sistema servidor. El conjunto de recursos es implementan a través de la tecnología de despliegue de servicios web REST, JAX-RS [Oracle, 2010].

- c.3) Módulo **Presentación de reportes:** permite la generación y despliegue de los reportes definidos en la capa de procesamiento utilizando ampliaciones a dichos componentes a través del patrón de diseño de software Decorator [Erich Gamma, 1994]. Cabe destacar, que los componentes gráficos de los reportes emplean la librería OpenFlashChart 2 (OFC2) y una implementación ajustada a dicho sistema del API de Java para OFC2 denominado JOFC2. En adición a lo anterior, la visualización de los gráficos del lado del navegador se logra a través de la implementación de un componente para JSF gracias a su gran capacidad de extensión.

- c.4) Módulo **Generación de bitácora:** debido a la utilización de la metodología XP y la realización de las pruebas al final de cada iteración, se incluye el framework log4j [Apache Software, 2010] para la generación de mensajes tipo log acerca de las actividades en el sistema e inclusive para las unidades de pruebas asociadas a cada componente existente en las capas de procesamiento y acceso a datos. Las unidades de pruebas utilizan la tecnología JUnit para pruebas unitarias de las clases y un conjunto de clases.

A continuación se detalla la interacción de cada uno de los módulos de la aplicación:

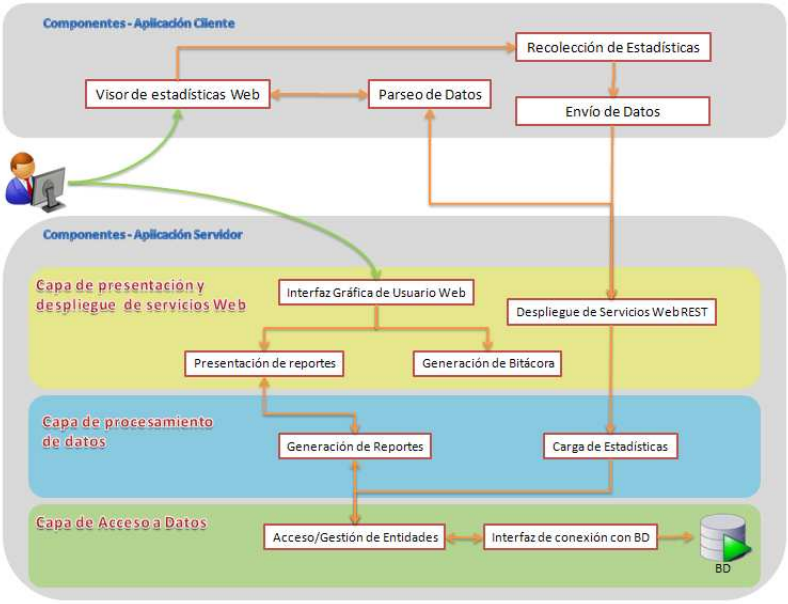


Figura 7.1: Interacción - Módulos de la Aplicación

7.4. Adaptación de las actividades XP

La aplicación de las actividades XP busca lograr el éxito en la comunicación con el cliente y la agilidad en el proceso de desarrollo.

Se describen cuatro actividades fundamentales durante las iteraciones, y las mismas son adaptadas como se describe a continuación:

- a)**Objetivo:** Se define de forma general los requerimientos que se esperan cumplir en una iteración determinada. Dicha actividad no pertenece al método XP pero se considera necesaria para el desarrollo de las iteraciones.
- b)**Planificación:** Se identifican las actividades realizadas para alcanzar el objetivo descrito.
- d)**Desarrollo:** Se identifican y proponen soluciones para llevar a cabo una determinada actividad, mediante el diseño de la solución y la elaboración de diagramas; aunado a esto, se realiza la implementación de las actividades propuestas en la planificación.
- e)**Pruebas:** Constituyen un conjunto de test de aceptación, en donde el usuario o cliente pone a prueba la solución desarrollada, y verifica que los requerimientos planteados han sido desarrollados correctamente.

Existe la posibilidad de que en algunas iteraciones no se contemplen todas las actividades descritas anteriormente. Lo antes expuesto esta condicionado a los requerimientos a desarrollar en cada iteración.

7.4.1. Estrategia de desarrollo

Cada iteración se realiza en un lapso estimado de 2 semanas y por cada 3 iteraciones se realiza una presentación al cliente para su aceptación. Cabe destacar, que a objeto de no hacer muy extenso el presente documento, cada iteración representa 4 semanas de trabajo, y en cada una de estas se resaltan los aspectos más importantes del desarrollo, por el motivo mencionado anteriormente.

El esquema de trabajo se estructura como se plantea a continuación:

7.4.2. Iteración 0

- **Objetivo:** Analizar y diseñar el modelo de datos del sistema en general así como el proceso de interacción entre la aplicación cliente y la aplicación servidor.
- **Planificación:**

  1. Análisis y diseño de la aplicación cliente y servidor respectivamente.
  2. Definición de la lógica de interacción entre la aplicación cliente y la aplicación servidor.
  3. Determinación de las tecnologías a utilizar en el lado del cliente y el lado del servidor, así como de las herramientas de desarrollo necesarias para construir la solución planteada.
  4. Investigación acerca de como crear una extensión para el navegador Mozilla Firefox.
  5. Creación de ejemplos de interfaces de usuario en XUL (XML User Interface).
  6. Creación de una extensión para el navegador Mozilla Firefox.
  7. Creación de la capa de persistencia para el modelo de datos de la aplicación utilizando la tecnología JPA(Java Persistence API).
  8. Creación de fábricas de objetos utilizando el Patron Abstract Factory.

- **Desarrollo:** En la figura 7.2, se muestra la primera versión del Diagrama de clases de la aplicación servidor.

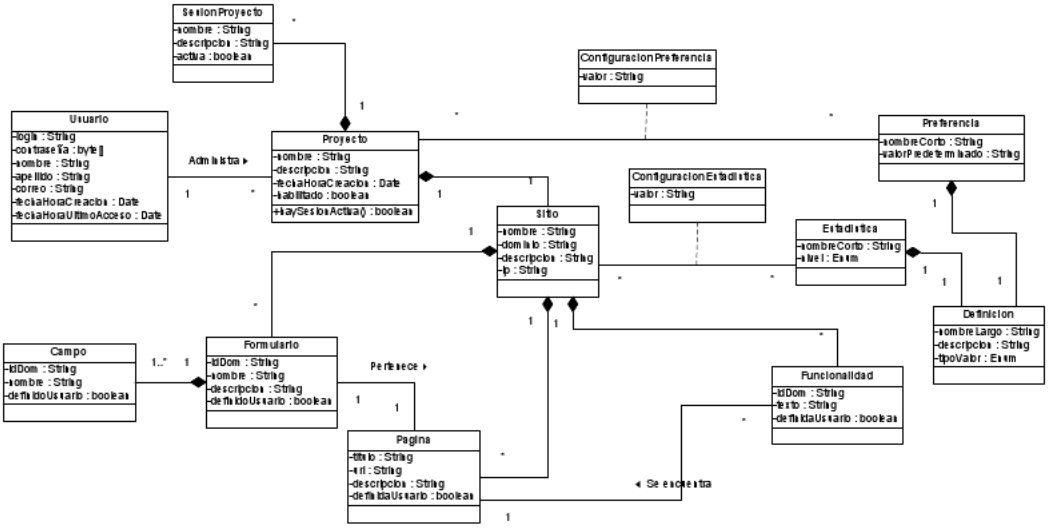


Figura 7.2: Diagrama de Clases de la aplicación Servidor - versión inicial

- **Pruebas:** A los componentes de base de datos construidos utilizando la librería JPA se le definen y ejecutan casos de pruebas unitarios y de interacción contra la base de datos cargada en MySQL, por lo cual se obtiene un error específicamente en la clase DAOEstadisticaVisitaJPA, realizando las correcciones pertinentes y obteniendo luego resultados exitosos en todos los casos de pruebas.

7.4.3. Iteración 1

- **Objetivo:**

  Crear el modelo inicial de las interfaces gráficas de usuario para la extensión de Mozilla Firefox e implementar los manejadores de datos de las entidades de proyectos y estadísticas del modelo de datos.

- **Planificación:**

  1. Creación de la interfaces: opcionesVisor.xul e infoOpcionesVisor.xul.
  2. Creación de las clases encargadas de manejar la información referente a los proyectos.
  3. Creación de una utilidad para el manejo de las preferencias de los proyectos y estadísticas.

4. Creación y actualización de DAOs (Data Access Object) para poder realizar las operaciones CRUD(Create-Read-Update-Delete) sobre las entidades de la base de datos.

- **Desarrollo:** En la figura 7.3 se puede visualizar el método encargado de cargar la lista de proyectos en el sistema.

```
cargarProyectos : function() {
    var ofv = org.fireweb.visorweb;
    var xml;
    var proys;

    // Se carga el XML con los proyectos desde el servidor
    xml = this._getXmlProyectos();
    if (!xml) return false;

    // Se cargan los datos de los proyectos en las estructuras de la
    // extensión
    proys = ofv.Util.getProyectosXML(xml);
    if (proys) {
        this._proyectos = proys;
        return true;
    } else {
        return false;
    }
},
```

Figura 7.3: Método encargado de cargar la lista de proyectos en el sistema

- **Pruebas:** Al igual que en la iteración anterior se ejecutan los casos de pruebas unitarios utilizando JUnit en los DAOs de proyectos y estadísticas, logrando encontrar un error en el identificador de la clase EstadisticaCampo. Adicionalmente se ejecutan pruebas de interacción de usuario con las interfaces gráficas en XUL de la extensión obteniendo resultados exitosos.

7.4.4. Iteración 2

- **Objetivo:**

  Analizar e implementar el primer conjunto de estadísticas recolectadas por la aplicación cliente tales como información general del computador, datos de clics, desplazamiento en las páginas, visualización en pantalla completa, y la creación de los documentos XML que se intercambiarán entre las aplicaciones involucradas.

- **Planificación:**

1. Investigación y desarrollo de la funcionalidad Número de visitas, Cantidad de acceso, Número de clics, Ubicación de los clics, Pantalla Completa, Información Cliente, Desplazamiento horizontal y Desplazamiento vertical, y Submit.

2. Creación de algunas de las clases generadoras del documento XML y de la configuración de todos los proyectos habilitados a existir en la aplicación.
3. Creación de una utilidad general con métodos de ayuda para el documento XML.
4. Actualización de la clase *GeneradorXMLProyecto* para generar el XML con la configuración completa de los proyectos de estadísticas. En este punto la clase se encuentra en una fase de pruebas ya que su desarrollo está por finalizar.

- **Desarrollo:** En la figura 7.4, se puede visualizar la implementación del método pantalla completa. En la figura 7.5, puede apreciarse la implementación de las funcionalidades: Versión del Navegador y Dirección IP.

```
onPantallaCompleta : function(browser, activada) {
    var contexto;
    var pag;

    // Recolectar o no
    if (!this._contexto.isEstPantallaCompletaActiva()) return;

    contexto = this.getContextoDeBrowser(browser);
    if (!contexto) {
        this._logContextoNoObtenido(browser.contentWindow,
            "onPantallaCompleta");
        return;
    }

    pag = this._contexto.getPaginaURL(contexto.docURL,
        contexto.docTitle);
    if (!pag) return;

    contexto.actualizarInfoPantallaCompleta(contexto.window, activada);

    if (this._logger.DEBUG) {
        this._logger.debug("Pantalla completa " +
            "en: " + this._ofv.Util.getUbicacionVentana(contexto.window) +
            ", activada: " + activada,
            "onPantallaCompleta", "ManejadorEvento");
    }
```

Figura 7.4: Funcionalidad Pantalla Completa01

```
getVersionNavegador : function() {
    var comps = String.split(navigator.userAgent, "/");
    var version = comps[comps.length - 1];
    comps = String.split(version, ".");
    version = comps[0];
    if (comps.length > 1) {
        version += "." + comps[1];
    } else {
        version += ".0";
    }
    return version
},

getIP : function() {
    if (!this.ip) {
        var cls = Components.classes['@mozilla.org/network/dns-service;1'];
        var dns = cls.getService(Components.interfaces.nsIDNSService);
        var nsrecord = dns.resolve(dns.myHostName, true);

        this.ip = nsrecord ? nsrecord.getNextAddrAsString() : null;
    }

    return this.ip;
},
```

Figura 7.5: Funcionalidades - Versión del Navegador , Dirección IP

- **Pruebas:** Los casos de pruebas en la presente iteración se realizan comparando los datos estadísticos generados por la aplicación cliente con respecto a la información del computador, contra los datos reales del computador donde se encuentra instalada la extensión. Cabe destacar que se obtiene una falla relacionada con la función encargada de obtener la versión del navegador. Dicha falla es corregida para dar fin a las pruebas de esta iteración.

7.4.5. Iteración 3

- **Objetivo:** Analizar e implementar el segundo conjunto de estadísticas capturadas por la extensión cliente asociadas con los tiempos de cargas de las páginas web, los tiempos de interacción del usuario y el desarrollar los componentes encargados de almacenar los datos enviados en formato XML por el cliente al servidor.

- **Planificación:**

  1. Investigar y desarrollar la funcionalidad Tiempo de Carga de una página.
  2. Investigar y desarrollar la funcionalidad Tiempo de Uso de una página; es decir el tiempo de actividad e inactividad de un usuario en una página Web.
  3. Actualización de la clase de utilidad para el manejo de documentos XML: *XMLUtil*.
  4. Desarrollo del método encargado de la generación del archivo XML que contiene la configuración de todos los proyectos habilitados en el sistema; en la clase *ManejadorXML*.
  5. Desarrollo del método para la carga de los datos estadísticos contenidos en un documento XML en las entidades de la base de datos, en la clase *ManejadorXML*.
  6. Verificación y validación de datos contenidos en el documento XML; y carga exitosa de los mismos a la base de datos. Aunado a esto, se realizan validaciones acerca del ingreso de datos no correctos objetivamente.
  7. Finalización de la validación objetiva de datos estadísticos contenidos en el documento XML; y finalización de la carga de datos a las diversas entidades de la base de datos.
  8. Creación de la clase ManejadorFabricasDAO; encargada del manejo e instanciación de las fábricas de DAOs, asociándolas a diversos hilos de ejecución.
  9. Actualización de las fábricas de DAOs, FabricaDAO y FabricaDAOJPA, a objeto de añadir métodos para el manejo de transacciones; y de cierre de la fábrica.
  10. Ajustes sobre el archivo de configuración de persistencia *persistence.xml* para trabajar bajo un ambiente JTA.
  11. Ajustes sobre la fábrica general *FabricaDAO* para la inclusión de la transacción JTA (manejada por el servidor web).

- **Desarrollo:** En la figura 7.6, se visualiza la manera en que es calculado el tiempo que tarda una página web en cargarse. Cabe destacar que el valor retornado es en milisegundos.

```
getTiempoCarga : function() {
    if (this._url && this._inicioCarga && this._finCarga) {
        return this._finCarga - this._inicioCarga;
    } else {
        return 0;
    }
},
```

Figura 7.6: Funcionalidad - Tiempo de Carga de una página

- **Pruebas:** El conjunto de pruebas de esta iteración se divide en dos secciones: cliente y servidor.
  En el lado del cliente, las pruebas se realizan con ayuda de la extensión HttpFox, midiendo los tiempos de cargas generados por la extensión cliente y comparándolos con los tiempos de cargas generados por la extensión HttpFox, los cuales fueron iguales con un margen de error de menos del 0.5 %.
  Respecto a la aplicación servidor las pruebas se ejecutan enviando datos de un archivo XML local bien definido (en total se arman 21 XMLs de prueba) y comparando estos datos con los almacenados por el sistema en la base de datos.
  Posteriormente se obtienen problemas de validación de data del XML al momento de ser almacenado, procediendo a su corrección y finalizando las pruebas de la iteración.

7.4.6. Iteración 4

- **Objetivo:**

  Terminar la implementación de la captura de estadísticas de información del usuario por parte de la extensión del navegador Mozilla Firefox así como diseñar y desarrollar las interfaces gráficas de usuario de inicio de sesión en la aplicación Web.

- **Planificación:**

  1. Culminación del desarrollo de la funcionalidad País de procedencia del usuario.
  2. Implementación de la funcionalidad Cantidad de Colores de la Pantalla de Computador.
  3. Implementación de la funcionalidad Dirección IP del usuario.
  4. Creación de la páginas de inicio de sesión, registro de usuarios, Administrar Proyectos y Creación de proyectos en Java Server Faces 1.2.

- **Desarrollo:**: En la figura 7.7 se pueden apreciar los componentes utilizados para obtener la dirección IP del usuario, tales como: Resolución DNS, y la utilización del servicio dns-service de Mozilla.

```
getIP : function() {
    if (!this.ip) {
        var cls = Components.classes['@mozilla.org/network/dns-service;1'];
        var dns = cls.getService(Components.interfaces.nsIDNSService);
        var nsrecord = dns.resolve(dns.myHostName, true);

        this.ip = nsrecord ? nsrecord.getNextAddrAsString() : null;
    }

    return this.ip;
},
```

Figura 7.7: Funcionalidad - Dirección IP del Usuario

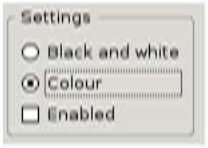
- **Pruebas:** Los casos de pruebas se ejecutan construyendo primero las clases que realizan las verificaciones utilizando la librería JUnit, que validan la correcta información de autenticación del usuario por el componente de inicio de sesión. Al finalizar el proceso de pruebas se obtienen resultados satisfactorios.

7.4.7. Iteración 5

- **Objetivo:** Diseñar e implementar las interfaces gráficas de usuario para la aplicación asociadas a la gestión de proyectos de estadísticas y finalizar la construcción de las interfaces gráficas para las opciones de configuración de la aplicación cliente.
- **Planificación:**

  1. Investigación y desarrollo de componentes de tipo Groupbox, PreferedPane y tabpanel de XUL.
  2. Inclusión de los métodos de habilitación y deshabilitación de los proyectos de estadísticas.
  3. Inclusión de los métodos de activación y desactivación de las sesiones de los proyectos de estadísticas.
  4. Realización de las pruebas correspondientes a los nuevos métodos.
  5. Ajustes en la página de creación y visualización de los detalles de cada uno de los proyectos, aunado a la creación de clases java para la habilitación y deshabilitación de proyectos.

- **Desarrollo:** En la figura 7.8 se proporciona un ejemplo de uno de los componentes tipo XUL, utilizados para la creación de las interfaces de usuario de la aplicación cliente.




```
<groupbox>
  <caption label="Settings"/>
  <radiogroup>
    <radio label="Black and white"/>
    <radio label="Colour"/>
  </radiogroup>
  <checkbox label="Enabled"/>
</groupbox>
```

Figura 7.8: Componente tipo groupbox

- **Pruebas:** Los casos de pruebas se construyen en base a la ejecución de flujos de procesos para la creación de proyectos de estadísticas, comparando los datos ingresados como entrada en la aplicación web con la información almacenados en la base de datos, es decir, se ejecutan pruebas basadas en el estado final del sistema. Adicionalmente se implementan las clases para los casos de pruebas empleando JUnit. Luego de la ejecución de las pruebas se realizan cambios en los

componentes de validación de datos de entrada y finaliza la fase de pruebas de esta iteración.

7.4.8. Iteración 6

- **Objetivo:** Analizar y desarrollar el tercer conjunto de estadísticas recolectadas por la extensión cliente relacionadas con el uso de los botones de navegación del navegador Mozilla Firefox e integrar los menús de acceso a las opciones de configuración con el navegador.
- **Planificación:**
  1. Implementación y realización de pruebas de las funcionalidades de uso de comando del navegador: Atrás, Siguiente, Actualizar y Detener.
  2. Actualización visual de las diversas páginas de la aplicación Web para mantener un patrón en el diseño de las mencionadas.
  3. Incorporación de los mensajes de confirmación al eliminar elementos en las tablas de lista de proyectos y lista de sesiones de proyectos.
- **Desarrollo:** En la figura 7.9 se muestra la invocación de los métodos Atrás, Adelante, Parar y Siguiente, encargados de contabilizar la cantidad de veces que la funcionalidad correspondiente es invocada.

```
// Recolectar o no
if ((comando == this._ofv.COMANDO_NAVEGADOR_ATRAS)
    && (!this._contexto.isEstComandoAtrasActiva())) return;
if ((comando == this._ofv.COMANDO_NAVEGADOR_ADELANTE)
    && (!this._contexto.isEstComandoAdelanteActiva())) return;
if ((comando == this._ofv.COMANDO_NAVEGADOR_PARAR)
    && (!this._contexto.isEstComandoPararActiva())) return;
if ((comando == this._ofv.COMANDO_NAVEGADOR_ACTUALIZAR)
    && (!this._contexto.isEstComandoActualizarActiva())) return;
```

Figura 7.9: Funcionalidad - Atras - Adelante - Parar - Siguiente

```
isEstComandoAtrasActiva : function() {
    return this._isEstActiva(org.fireweb.visorweb.Util.ESTADISTICA_COMANDO_ATRAS);
},
isEstComandoAdelanteActiva : function() {
    return this._isEstActiva(org.fireweb.visorweb.Util.ESTADISTICA_COMANDO_ADELANTE);
},
isEstComandoPararActiva : function() {
    return this._isEstActiva(org.fireweb.visorweb.Util.ESTADISTICA_COMANDO_PARAR);
},
isEstComandoActualizarActiva : function() {
    return this._isEstActiva(org.fireweb.visorweb.Util.ESTADISTICA_COMANDO_ACTUALIZAR);
},
```

Figura 7.10: Funcionalidad - Atras - Adelante - Parar - Siguiente02

- **Pruebas:** Para esta iteración las pruebas consisten en definir el flujo de activación de los botones de navegación del navegador y comparar las estadísticas generadas con las reales definidas en el flujo. Al generar los resultados de los casos de pruebas se observan errores en la captura debido a que se capturaban usos con el ratón, mas no el uso de métodos abreviados por teclado. Posteriormente se realizan las correcciones para dar fin a las actividades de prueba de forma exitosamente.

7.4.9. Iteración 7

- **Objetivo:** Diseñar e implementar la comunicación realizada entre la aplicación cliente y la aplicación servidor.
- **Planificación:**

  1. Investigación acerca de la creación de servicios Web en JavaScript.
  2. Desarrollo de servicios Web REST en JavaScript, tales como:
     - Obtener información completa acerca de los proyectos de estadísticas que se encuentran habilitados.
     - Obtener información acerca de si un proyecto de estadísticas en específico está activo o no.
     - Obtener un nuevo ID de estadística de visita.
     - Cargar la infomación obtenida acerca de una estadística de visita.
  3. Pruebas sobre los servicios Web creados.
  4. Incorporación de las librerías y configuración de las mencionadas para la activación de los servicios Web REST implementados con JAX-RS. Se modifica el archivo web.xml y se incorpora un Servlet para el manejo de los servicios web JAX-RS.

5. Inclusión de las librerías de JAX-RS 1.0 y modificación de la librería de Hibernate debido a inconsistencias de la misma con la librería de JAX-RS.
6. Creación de clases que funcionan como recursos para los servicios Web REST: *ProyectosResource, ProyectoResource, EstadisticasResource, EstadisticaResource*. Se crea un único recurso principal (*ProyectosResource*) y el resto constituyen sub-recursos localizados a través de parámetros en el URL.

- **Desarrollo:** En las figuras 7.11, 7.12 y 7.13, se muestra la invocación e implementación de la obtención de cada uno de proyectos que han sido creados en la aplicación servidor.

```
_getXmlProyectos : function() {
    var ofv = org.fireweb.visorweb;
    var urlSW = this.prefs.getUrlServicioWeb();
    var sm;
    var xml;
    if (!urlSW) return null;

    // Se crea el cliente del servicio web y se invoca al servicio para
    // obtener los proyectos creados en el servidor
    sm = new ofv.ServiceManager(urlSW);
    xml = sm.getProyectos();
    sm = null;

    return xml;
}
```

Figura 7.11: Método encargado de obtener los proyectos creados en la aplicación servidor

```
org.fireweb.visorweb.ServiceManager = function(urlBase) {
    this.inicializar(urlBase);
}


/**
 * Prototipo para un ServiceManager.
 */
org.fireweb.visorweb.ServiceManager.prototype = {
    inicializar : function(urlBase) {
        var url;
        if (urlBase) {
            url = urlBase;
        } else {
            url = org.fireweb.visorweb.gVisorWeb.prefs.getUrlServicioWeb();
        }
        this.setUrlBase(url);
    },

    getUrlBase : function() {
        return this.urlBase;
    },

    setUrlBase : function(urlBase) {
        this.urlBase = org.fireweb.visorweb.ServiceManager.formatURL(urlBase);
        this.actualizarUrls();
    },
```

Figura 7.12: Clase encargada de obtener los proyectos creados en el servidor

```
getProyectos : function() {
    return org.fireweb.visorweb.svcHelper.get(this.urlProyectos,
        "text/xml", false, true);
},
```

Figura 7.13: Método encargado de realizar la petición de los proyectos existentes en la aplicación servidor

- **Pruebas:** En la presente iteración los casos de pruebas se distribuyen en dos vertientes: pruebas de los servicios web REST y pruebas de los servicios invocados por la extensión.
  Para los servicios web REST se crean clientes simulados que invocan a los servicios definidos en el servidor y posteriormente se compara el resultado obtenido con el esperado, lo cual significa que se realizan pruebas en base al estado final. Cabe destacar que dichas pruebas culminaron exitosamente.
  Por parte del cliente, las pruebas se realizan invocando a los servicios residentes en el servidor y comparando la respuesta obtenida por el cliente con la enviada por el servidor. Cabe destacar que las fallas encontradas se producen debido al manejo erróneo de casos de excepción al realizar la invocación de los servicios, las cuales fueron corregidas concluyendo las pruebas.

7.4.10. Iteración 8

- **Objetivo:** Investigar e implementar el cuarto conjunto de estadísticas capturadas por la extensión para el navegador Mozilla Firefox, asociadas a los tiempos de carga de los recursos de las páginas web; además diseñar y desarrollar la fase inicial para la gestión de las estadísticas en la aplicación web.
- **Planificación:**
  1. Investigación acerca de como implementar la funcionalidad Tiempo de Carga por Recurso.
  2. Revisión del código fuente de la extensión Firebug a objeto de determinar su funcionamiento, para posteriormente localizar la implementación de la funcionalidad comentada.
  3. Realización de pruebas acerca de la funcionalidad mencionada.
  4. Creación de la página inicial de administración de estadísticas (inicio.jspx).
  5. Creación de la página de selección del proyecto para visualizar las estadísticas (proyecto.jspx).
  6. Creación de la página de selección de sesión de proyecto, para visualizar las estadísticas (sesion.jspx).

7. Creación básica de la página principal de visualización de las estadísticas de un proyecto y sesiones seleccionadas (datos.jspx).
8. Ajuste del archivo menu.jpsf incluyendo la navegación a la página de administración de estadísticas.

- **Desarrollo:** En las figuras 7.14 y 7.15, se visualiza la implementación de los tiempos de carga por recurso de una página Web.

```
onModifyRequest : function(req, win, idTab) {
    var tiempo;
    // Ignorar redirecciones HTTP
    if (req.URI.spec != req.originalURI.spec) return;

    tiempo = new Date().getTime();

    // Sólo las solicitudes de documentos principales (no marcos - frames)
    // son los que deben generar el evento onSolicitudPagina
    if ((req.loadFlags & Components.interfaces.nsIHttpChannel.LOAD_DOCUMENT_URI) &&
        (win == win.parent)) {
        // Generación del evento de solicitud de página
        this._obs.onSolicitudPagina(req, win, tiempo);

        if (this._logger.DEBUG) {
            this._logger.debug("Solicitud de página para: " +
                this._ofv.Util.aURItoURLSimple(req.URI),
                "onModifyRequest", "RecolectorHTTPObserver");
        }
    }

    if (this._logger.DEBUG) {
        this._logger.debug("Solicitud de recurso para: " +
            this._ofv.Util.getNombreSolicitud(req),
            "onModifyRequest", "RecolectorHTTPObserver");
    }

    // Generación del evento de solicitud de recurso
    this._obs.onSolicitudRecurso(req, tiempo, win, idTab);
},
```

Figura 7.14: Implementación funcionalidad - Tiempo de Carga por recurso

```
onExamineResponse : function(req, win, idTab) {
    var tiempo;
    // Ignorar redirecciones HTTP
    if (req.URI.spec != req.originalURI.spec) return;

    tiempo = new Date().getTime();

    if (this._logger.DEBUG) {
        this._logger.debug("Respuesta de recurso para: " +
            this._ofv.Util.getNombreSolicitud(req),
            "onExamineResponse", "RecolectorHTTPObserver");
    }

    // Verificando si la respuesta HTTP fue un código de error
    if (this._ofv.Util.isErrorHTTP(req.responseStatus)) {
        this._generarErrorHTTP(req, win);
    }

    // Generación del evento de carga del recurso
    this._obs.onRespuestaRecurso(req, tiempo, win, idTab);
},
```

Figura 7.15: Continuación funcionalidad Tiempo de Carga por recurso

- **Pruebas:** Las pruebas realizadas en esta iteración se enfocan principalmente en la captura de estadísticas de tiempos de cargas de recursos. Al igual como se describe en la iteración 3, estas se realizan con ayuda de la extensión HttpFox, en este caso en lugar de comparar los tiempos de cargas de las páginas se comparan los tiempos de cargas de sus recursos. Las páginas utilizadas para estas pruebas son la página de inicio de sesión y la página del menú principal del sistema CONEST Estudiantes. Cabe destacar que el resultado de estas pruebas es exitoso.

7.4.11. Iteración 9

- **Objetivo:** Diseñar y desarrollar el proceso inicial para la visualización de los reportes estadísticos en la capa de aplicación del servidor y analizar e implementar el quinto conjunto de estadísticas recolectadas por la aplicación cliente asociadas al uso de funcionalidades del navegador y captura de códigos de error de solicitudes HTTP.

- **Planificación:**

  1. Implementación de las funcionalidades Búsqueda de Texto, Tamaño de Vista y Errores HTTP.
  2. Realización de pruebas sobre las funcionalidades descritas anteriormente.
  3. Extensión de la librería de Openflash Chart 2, incluyendo la capacidad de visualizar gráficos en forma de imágen, para posteriormente ser incluída en la aplicación Web.

- **Desarrollo:** En las figuras 7.16 y 7.17 se muestra el código fuente de la funcionalidad Búsqueda.

```
onBusqueda : function(e, barra) {
    var cadenaBusqueda;
    var elem;
    try {
        elem = e.originalTarget;

        // Filtrado de los eventos de búsqueda
        if (barra) {
            // ... el evento se produjo en la barra de búsqueda rápida

            // Si se presionó una tecla, debe ser un elemento de tipo
            // texto y ser la tecla Enter (13) --> Acción de busqueda en
            // cuadro de texto
            if ((e.type == "keypress")
                && ((elem.localName != "input")
                    || (e.keyCode != 13))) return;

            // Si se ejecutó el evento command, debe ser un botón de
            // herramientas y tener una cadena 'next' o 'previous' en el
            // nombre de la clase
            if (e.type == "command") {
                if (elem.localName != "toolbarbutton") return;

                if ((elem.className.indexOf("next") == -1)
                    && (elem.className.indexOf("previous") == -1))
                    return;
            }
        }
```

Figura 7.16: Implementación funcionalidad - Búsqueda

```
            // Obtención de la cadena de búsqueda
            try {
                cadenaBusqueda = gBrowser.fastFind.searchString;
            } catch (e) {
                try {
                    cadenaBusqueda = gBrowser.webBrowserFind.searchString;
                } catch (ex) {
                    if (this._logger.ERROR) {
                        this._logger.error("No se pudo obtener la cadena de " +
                            "búsqueda: " + ex,
                            "onBusqueda", "BusquedaListener");
                    }
                }
            }

            // Valida que sea una cadena de búsqueda válida
            if (org.fireweb.visorweb.Util.isCadenaEspacios(cadenaBusqueda))
                return;

            this._escuchador.onBusquedaCadena(gBrowser.selectedBrowser,
                cadenaBusqueda);
        } catch (e) {
            this._logger.error("Error en observación de evento de " +
                "búsqueda: " + e,
                "onBusqueda", "BusquedaListener");
        }
    }
```

Figura 7.17: Continuación - Implementación funcionalidad - Búsqueda01

- **Pruebas:** En esta iteración los casos de pruebas se basan exclusivamente en las estadísticas de usos de funciones del navegador para lo cual se definen casos para las funciones de tamaño de vista, búsqueda de texto y capturas de errores HTTP. Para las dos primeras se definen casos de pruebas en base al estado final, es decir, se interactúa de manera predefinida con la extensión y al final se valida que haya generado las estadísticas correctas. Para el último caso, se crea una página de prueba con errores de enlaces y de recursos preestablecidos y se prueba la extensión con la página creada, validando al final que haya generado las estadísticas correctas.
  Al ejecutar las pruebas se hallan errores en la función de captura de búsqueda de texto, ya que el navegador busca el texto a medida que el usuario escribe en el cuadro de búsqueda, por lo tanto se realiza la corrección en la manera como la extensión captura la cadena buscada.

7.4.12. Iteración 10

- **Objetivo:** Agregar capacidad de filtrado en la interfaz gráfica de usuario web para los reportes estadísticos e implementar el sexto conjunto de estadísticas recolectadas por el cliente con respecto al uso de los enlaces de las páginas web.

- **Planificación:**

  1. Investigación y realización de pruebas de las funcionalidades: Uso Funcionalidad, Forma Uso Funcionalidad, y Desplazamiento funcionalidad.

2. Definición del esquema general de la página.
3. Agregación de campos de filtros de fecha y hora al backing bean de la página.
4. Agregación de estilos adicionales en ***general.css*** que actuan sobre el esquema de la página.
5. Ajustes a la página datos.jspx.
6. Incorporación de cadenas de I18N en FireWeb.properties y en FireWebMsg.properties.
7. Ajuste en el Backing Bean EstadisticasBean mediante la incorporación de métodos para aplicar y quitar filtros, cambiar el proyecto y las sesiones.
8. Ajuste del bean básico AbstractBean mediante la incorporación de la cadena de navegación por defecto hacia atrás.

- **Desarrollo:**

  En la figura 7.18 se muestra el código fuente de la funcionalidad Uso Funcionalidad.

```
onUsoFuncionalidad : function(cont, func, evt, tiempo) {
    var est;
    var formaAcceso;

    // Verificando que la forma de acceso no sea guardar la página del
    // elemento activado o que no sea el botón secundario del mouse
    formaAcceso = whereToOpenLink(evt);
    if ((formaAcceso == "save") || (evt.button == 2)) return;

    est = new this._ofv.UsoFuncionalidad();
    if (func.getId()) {
        // existe la funcionalidad definida en el proyecto, solo se establece el id
        est.setIdFunc(func.getId());
    } else {
        // no existe la funcionalidad definida en el proyecto, se
        // establece el dom y el texto
        est.setIdDomFunc(func.getIdDom());
        est.setTextoFunc(func.getTexto());
    }

    // Forma de uso
    if (!evt.keyCode) {
        // ... por ratón
        est.setFormaUso(this._ofv.FORMA_USO_RATON);
    } else {
        // ... por teclado
        est.setFormaUso(this._ofv.FORMA_USO_TECLADO);
    }
```

Figura 7.18: Implementación funcionalidad - Uso Funcionalidad

- **Pruebas:** Los casos de pruebas en esta iteración se enfocan en las funcionalidades de la aplicación cliente, tal como se ha venido implementando las pruebas del lado del cliente en las iteraciones anteriores, estos casos se basan en el estado final de

la captura de estadísticas. Para este caso se emplea la página del menú principal del sistema CONEST Estudiantes como ámbito de captura de estadísticas de la extensión y de manera establecida se activan los enlaces de la página, que al final de la captura, la extensión genera las estadísticas de usos de enlaces los cuales se comparan con los usos reales realizados y se arrojan los resultados que son totalmente exitosos.

7.4.13. Iteración 11

- **Objetivo:** Analizar e implementar el séptimo conjunto de estadísticas capturadas por la extensión de Mozilla Firefox relacionadas con los tiempos de llenado de los formularios, y desarrollar los componentes para la generación de los datos de los reportes estadísticos.

- **Planificación:**

  1. Investigación y realización de pruebas de la funcionalidad Llenado del formulario.
  2. Creación de la interfaz pública Filtro, utilizada por el cliente (capa de presentación), para establecer restricciones de datos a los resportes.
  3. Creación de la interfaz privada FiltroAplicable, utilizada por la capa del núcleo para aplicar los filtros a los reportes.
  4. Creación del filtro general denominado FiltroGenerico, como base para los diversos filtros asociados los reportes.
  5. Creación de la fábrica (FabricaFiltro), para la obtención de filtros prefabricados y definidos por la capa de núcleo que pueden ser utilizados por los clientes.
  6. Creación de la interfaz pública Filtrable, para permitir que a un componente se le puedan agregar o establecer filtros. Dicha interfaz es implementada por los reportes a objeto de que los mencionados puedan ser filtrados.
  7. Creación de las clases de pruebas de JUnit para: FiltroGenerico y FabricaFiltro.

- **Desarrollo:** En las figuras 7.19 y 7.20, se muestra la implementación del Tiempo de llenado de un formulario.

```
onEventoDocumento : function(win, doc, evt, tiempo) {
    var debug = this._logger.DEBUG;
    var pag;
    var form;
    var elem;
    var contexto;
```

Figura 7.19: Implementación funcionalidad - Tiempo de llenado de un formulario 01

```
// Se valida que se desea agregar estadísticas para el formulario
// y el campo del evento
form = this._contexto.getFormulario(pag, elem.form);
if (!form) return;
if (!this._contexto.getCampo(form, elem)) return;

// Tipo de evento generado
// Si es focus, solo se inicia el llenado a aquellos elementos que
// no sean entrada de texto, ya que los campos de entrada de texto
// comienza el llenado cuando se empieza a escribir en el mismo
if ((evt.type == "focus" && !this._ofv.Util.isElementoEntradaTexto(elem)) ||
    (evt.type == "keypress")) {
    // ... inicio de entrada de datos en el campo
    contexto.inicioLlenadoCampo(elem.form,
        this._contexto.isEstCampoActiva() ? elem : null,
        tiempo);

    if (debug) {
        this._logger.debug("Inicio de llenado de " +
            "formulario: " + elem.form.action +
            ", campo: " + elem.id,
            "onEventoDocumento", "ManejadorEvento");
    }
} else if (evt.type == "blur") {
    // ... fin de entrada de datos en el campo
    contexto.finLlenadoCampo(elem.form,
        this._contexto.isEstCampoActiva() ? elem : null,
        tiempo);

    if (debug) {
        this._logger.debug("Fin de llenado de " +
            "formulario: " + elem.form.action +
            ", campo: " + elem.id,
            "onEventoDocumento", "ManejadorEvento");
    }
}
```

Figura 7.20: Continuación funcionalidad - Tiempo de Llenado de un formulario 02

- **Pruebas:** Para esta iteración los casos de pruebas unitarias se construyen utilizando la librería JUnit, implementando casos para los componentes de reportes estadísticos, específicamente las clases FiltroGenerico y FabricaFiltro. En efecto, los casos se implementan en base al estado final de los valores de los atributos de las instancias de las clases. Al final de las pruebas los resultados son totalmente exitosos.

7.4.14. Iteración 12

- **Objetivo:** Diseñar y construir los componentes de sistema para la creación de los diversos reportes estadísticos generados por la aplicación servidor e investigar y desarrollar las estadísticas de tiempo de respuesta de formularios capturadas por la aplicación cliente.
- **Planificación:**

  1. Investigación y desarrollo de la funcionalidad Tiempo de respuesta del formulario.
  2. Realización de pruebas sobre la funcionalidad desarrollada.
  3. Modificación de las interfaces de usuario de la aplicación cliente.
  4. Creación de la interfaz general de los reportes a ser generados en la aplicación.
  5. Creación de la interfaz *GeneradorData*, encargada de la generación de consultas y datos de los reportes. Se emplea el patrón de diseño Strategy para su implementación.
  6. Creación de la clase *DatoReporte*, encargada de almacenar los datos acerca del detalle y resumen obtenidos por los generadores de los reportes.
  7. Creación de la clase *GrupoConsulta*, encargada de almacenar los datos acerca de las consultas creadas por los generadores de los reportes.
  8. Creación de la clase encargada de la generación de reportes en la aplicación. Clase ReporteGenerico.
  9. Creación de la clase abstracta *AbstractGeneradorData*, que sirve como implementación base para las clases generadoras de datos para los reportes. Dicha clase junto a las clases *GeneradorData.java* y *ReporteGenerico.java*, forman parte del patrón Strategy.
  10. Creación de la clase *GeneradorDataPool.java*, que contiene los generadores de reportes de la aplicación. Cabe destacar que a raíz de que los generadores no almacenan información de estado entre llamadas a sus métodos, se emplea el patrón Flyweight para almacenar referencias únicas a dichos generadores sobre el pool mencionado.
  11. Creación de la clase *FabricaReporte.java*, encargada de la generación de instancias de los reportes disponibles en el sistema, y las estadísticas abarcadas por los mencionados, junto con el generador de reportes correspondiente (dicha asociación se realiza a través del nombre del reporte).
  12. Creación de clases de prueba para: *ReporteGenerico.java, GeneradorDataPool.java y FabricaReporte.java.*
  13. Pruebas sobre el servidor de aplicaciones para la creación de usuarios y proyectos de estadísticas.

- **Desarrollo:** En la figuras 7.21 se proporciona el Diagrama de Clases inicial para la construcción de los reportes del núcleo de la aplicación.

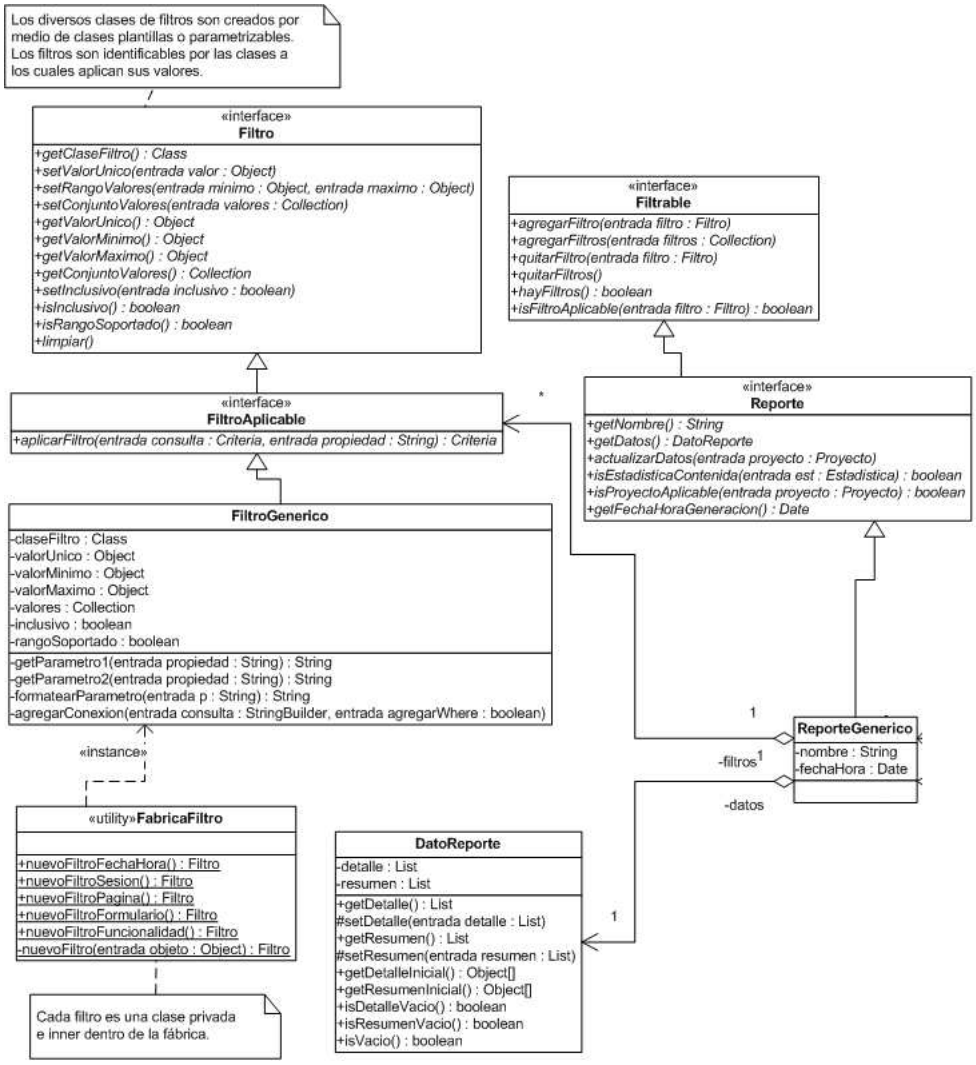


Figura 7.21: Clases de los reportes para el núcleo

- **Pruebas:** Al igual que en la iteración 11 los casos de pruebas unitarias y de interacción se construyen utilizando la librería JUnit, implementando casos para los componentes de generación de datos para los reportes, en específico las clases AbstractGeneradorData, GeneradorData, ReporteGenerico, GeneradorDataPool y FabricaReporte. En efecto, los casos se implementan en base al estado final de los valores de los atributos de las instancias de las clases. Al ejecutar las pruebas por parte del cliente se encuentra un comportamiento erróneo en el método isProyectoAplicable de la clase ReporteGenerico, por lo tanto se realizan los cambios y las pruebas se ejecutan correctamente.

7.4.15. Iteración 13

- **Objetivo:** Analizar e implementar la estadística errores de llenados de formularios recolectada por la extensión cliente, y desarrollar los componentes de la capa de presentación para formatear los datos generados por los reportes estadísticos.
- **Planificación:**

  1. Investigación y desarrollo de la funcionalidad Errores Formulario.
  2. Realización de pruebas sobre la funcionalidad desarrollada.
  3. Creación del modelo básico para cada dato contenido en los diversos reportes del núcleo. Clase *DatoModel*.
  4. Creación de la clase para el formateo de cada dato básico. Clase *FormatoDato*.
  5. Creación de generadores de textos para los diversos valores de datos. Se define la interfaz global de un generador, denominada *GeneradorTexto*. Posteriormente se definen los diversos generadores para los enumerados: *GeneradorFormaUso, GeneradorFormaAcceso, GeneradorComandoNavegador, GeneradorColoresPantalla*.
  6. Creación del almacenador de los diversos conjuntos de formatos de detalle y resumen para todos los reportes disponibles en el núcleo. Clase *FormatoDatoPool*.
  7. Creación de clases de prueba sobre los módulos de reportes creados.

- **Desarrollo:** En la figura 7.22 se proporciona el Diagrama de Clases inicial para la presentación de los reportes en la GUI Web.

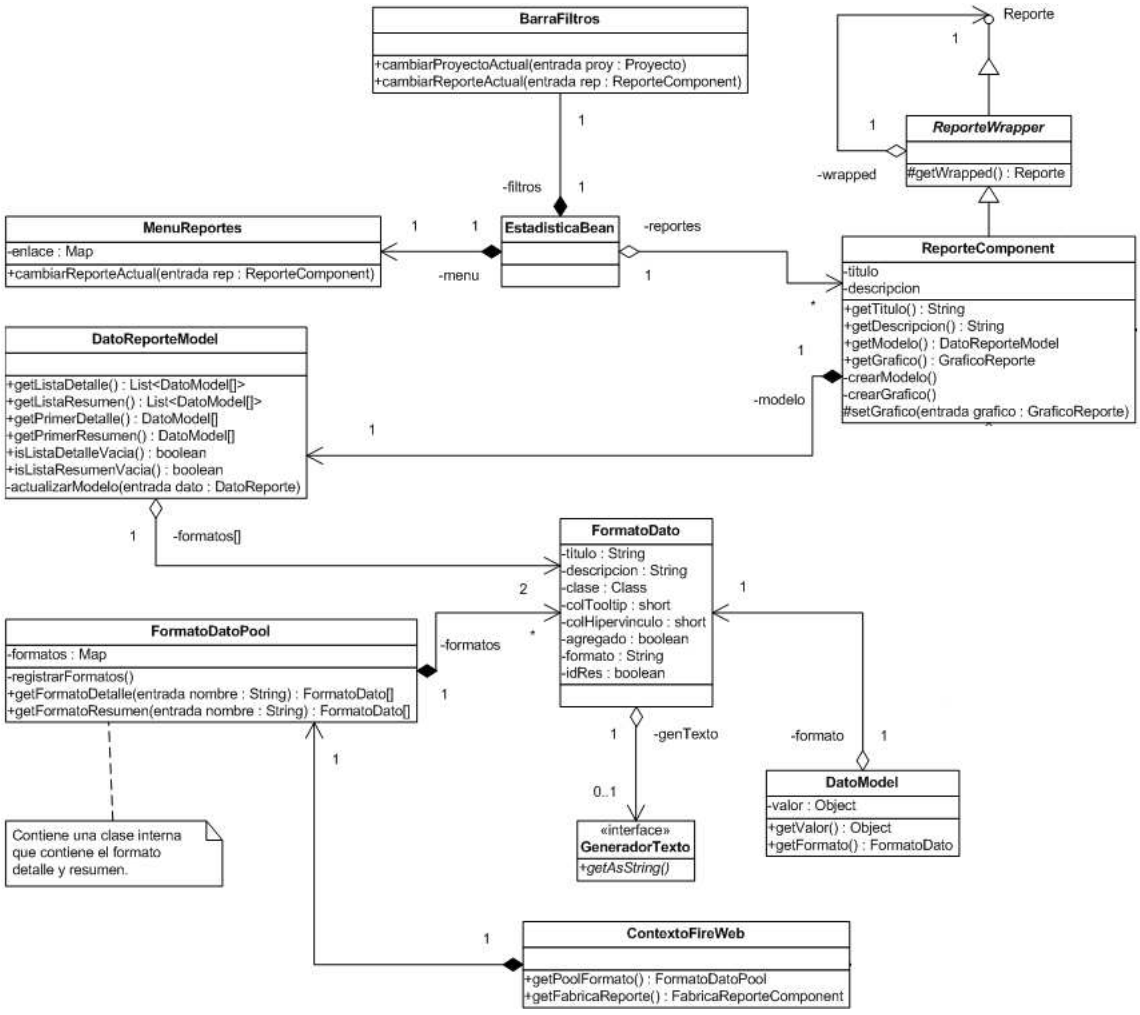


Figura 7.22: Clases para la presentación de los reportes en la GUI Web

- **Pruebas:** De la misma manera como se describe en la iteración 12 los casos de pruebas unitarias se arman utilizando la librería JUnit, definiendo casos para los componentes de formateo de datos asociados los reportes estadísticos, en concreto para las clases DatoModel, FormatoDato, FormatoDatoPool y GeneradorTexto. De hecho, los casos se implementan en base al estado final de los valores de los atributos de los objetos. Al final el resultado de las pruebas es exitoso.

7.4.16. Iteración 14

- **Objetivo:** Finalizar la construcción del documento en formato XML con los datos capturados por la aplicación cliente, y desarrollar los componentes para la generación de reportes estadísticos en la interfaz gráfica de usuario en la aplicación web.

- **Planificación:**

  1. Construcción del segmento XML asociado a cada una de las funcionalidades desarrolladas.
  2. Construcción del XML que es enviado al servidor para su posterior procesamiento.
  3. Realización de pruebas sobre los archivos XML creados a objeto de poder establecer la comunicación entre la aplicación cliente y la aplicación servidor.
  4. Creación del modelo para el componente de los reportes en el GUI Web. Clase *DatoReporteModelo*.
  5. Creación del componente que decora un Reporte del núcleo y le añade funcionalidades al mencionado, a objeto de que pueda ser visualizado en la capa Web (Clase *ReporteComponent*).
  6. Creación de la fábrica encargada de instanciar los reportes para su posterior visualización en la interfaz Web (Clase *FabricaReporteComponent*). Cabe destacar que existe una única instancia de dicha fábrica, por cada instancia de la aplicación Web. Lo mencionado se logra a través del patrón Singleton.
  7. Creación del componente MenuReporte, encargado de manejar la visualización del menú, y el acceso a los diversos reportes de estadísticas del sistema.
  8. Creación del componente ModeloReporte, que contiene la información del proyecto y el reporte actual seleccionado en la interfaz Web.
  9. Ajuste de la página web de estadísticas datos.jspx, a objeto de poder obtener el modelo de la clase MenuReporte en lugar de crearlo manualmente. De esta manera el menú se crea dinámicamente.

- **Desarrollo:** En la figuras 7.23 se muestra como se construye el archivo XML, que es enviado a la aplicación servidor.

```
getDocumentoXML : function() {
    var doc = null, elem;
    if (!this._validarVisitaActiva) return null;

    try {
        // Primero se crea el elemento raíz 'Proyecto' con su atributo
        // 'nombre'
        doc = document.implementation.createDocument(null, "proyecto", null);
        doc.documentElement.setAttribute("nombre",
            this._contexto.getProyecto().getNombre());

        // Ahora se crea la estadística de visita de este almacenador
        elem = this._visita.toElementoXML(doc);

        // Procesamiento de todas las estadísticas de todas las páginas
        for (var i = 0; i < this._pags.length; i++) {
            elem.appendChild(this._getDocXMLPagina(doc, this._pags[i]));
        }

        // Se agregan todas las estadísticas al documento
        doc.documentElement.appendChild(elem);
    } catch (e) {
        doc = null;
        if (this._logger.ERROR) {
            this._logger.error("Ocurrió un error creando el documento " +
                "XML de las estadísticas de la visita: " + this.getIdVisita() +
                ", error: " + e,
                "getDocumentoXML", "AlmacenadorEstadistica");
        }
    }
    return doc;
},
```

Figura 7.23: Implementación funcionalidad - XML que será enviado a la aplicación servidor

- **Pruebas:** Al igual como se expone en la iteración 12 los casos de pruebas unitarias y de interacción se desarrollan utilizando la librería JUnit, definiendo casos para los componentes de presentación de los reportes estadísticos en la interfaz web, en concreto para las clases DatoReporteModelo, ReporteComponent, FabricaReporteComponent, MenuReporte y ModeloReporte. En este caso, las pruebas se implementan en base al estado final y en base a la interacción entre clases. Al ejecutar las pruebas por parte del cliente se obtiene un error de visualización del menú de reportes estadísticos en la clase MenuReporte, el cual se corrige y se continuan las pruebas culminando con esta iteración luego de obtener la aceptación del cliente.

7.4.17. Iteración 15

- **Objetivo:** Incorporar el cifrado en los datos de cuentas de usuario y realizar ajustes generales en las interfaces gráficas de usuario tanto en la extensión como en la aplicación web.

- **Planificación:**

  1. Realización de pruebas sobre las funcionalidades desarrolladas en la aplicación cliente y de los archivos XML construidos.
  2. Modificación de la interfaz de usuario opcionesVisor.xul.
  3. Ajustes en la creación de una página para un proyecto determinado, de manera que solo acepte URLs sin Schema.
  4. Modificación sobre el validador URL para que acepte los dos tipos de URLs (con schema y sin schema).
  5. Eliminación de la preferencia de modo de envío en la creación de un proyecto de estadísticas, por defecto se selecciona el modo de envío por sesión.
  6. Definición de la interfaz general para un cifrador (Cifrador) y se crea la clase abstracta *AbstractCifrador*, con la finalidad de implementar contraseñas cifradas en la aplicación.
  7. Agregación de clases para el manejo de diversos métodos de cifrado para la aplicación web, específicamente para el área de manejo de contraseñas. Se definen tres cifradores/descifradores: *CifradorNulo*, *CifradorDES* y *CifradorTripleDES*.
  8. Ajustes de los nombres de las tablas de base de datos de minúscula a mayúscula, durante la instalación de la base de datos en Linux (servidor Yagrumo), la cual es sensible a mayúsculas/minúsculas.

- **Desarrollo:** En la figuras 7.24 y 7.25 se muestran los diversos archivos de configuración de la aplicación cliente.

Start Page | chrome.manifest | install.rdf

```
1 content {appname}   content/
2
3 skin    {appname}   classic/1.0                        skin/classic/
4
5 locale  {appname}   es-VE                               locale/es-VE/
6
7 style   chrome://global/content/customizeToolbar.xul    chrome://{appname}/skin/overlay.css
8
9 overlay chrome://browser/content/browser.xul           chrome://{appname}/content/overlay.xul
```

Figura 7.24: Contenido archivo - chrome.manifest

```
<?xml version="1.0"?>
<RDF:RDF xmlns:em="http://www.mozilla.org/2004/em-rdf#"
         xmlns:NC="http://home.netscape.com/NC-rdf#"
         xmlns:RDF="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <RDF:Description RDF:about="urn:mozilla:install-manifest">

        <em:id>{appid}</em:id>
        <em:type>2</em:type>
        <em:name>{appdisplayname}</em:name>
        <em:version>{appversion}</em:version>
        <em:description>{appdescription}</em:description>
        <em:creator>{auther}</em:creator>
        <em:developer>{developer1}</em:developer>
        <em:developer>{developer2}</em:developer>
        <em:optionsURL>chrome://{appname}/content/opcionesVisor.xul</em:optionsURL>
        <em:iconURL>chrome://{appname}/skin/iconos/visor_16.png</em:iconURL>
        <!-- URL Acerda -->
        <em:aboutURL>chrome://{appname}/content/acerca.xul</em:aboutURL>
        <em:homepageURL>{homepage}</em:homepageURL>

        {targetApplications}

        <em:file>
            <RDF:Description about="urn:mozilla:extension:file:{appname}.jar">
                <em:package>content/</em:package>
                <em:locale>locale/es-VE/</em:locale>
                <em:skin>skin/classic/</em:skin>
            </RDF:Description>
        </em:file>
    </RDF:Description>
</RDF:RDF>
```

Figura 7.25: Contenido archivo - install.rdf

- **Pruebas:** El conjunto de casos de pruebas se enfoca en el componente de cifrado de datos de contraseñas de cuentas de usuarios. Para este fin los casos se implementan empleando la librería JUnit en particular para las clases: AbstractFactory, CifradorNulo, CifradorDES y CifradorTripleDES. Además, se especifican pruebas basadas en el estado final de los atributos de las clases. Por lo demás es importante señalar que el resultado de las pruebas es completamente satisfactorio y se obtiene la aceptación del cliente.

7.4.18. Iteración 16

- **Objetivo:** Configurar los componentes de la extensión y del servidor para preparar y ejecutar el caso de estudio, y finalizar el desarrollo de los reportes estadísticos del sistema.

- **Planificación:**

  1. Instalación de la aplicación cliente en 12 máquinas de la División de Control de Estudios de la Facultad de Ciencias de la Universidad Central de Venzuela por el profesor Sergio Rivas.
  2. Monitoreo de la información recolectada por la aplicación cliente y de la data almacenada en la aplicación servidor.
  3. Respaldo de la data almacenada en la aplicación servidor.
  4. Pruebas sobre los reportes generados.

- **Desarrollo:** En la figura 7.26 se proporciona el Diagrama de Clases asociado a la visualización de los gráficos de los reportes en la GUI Web.

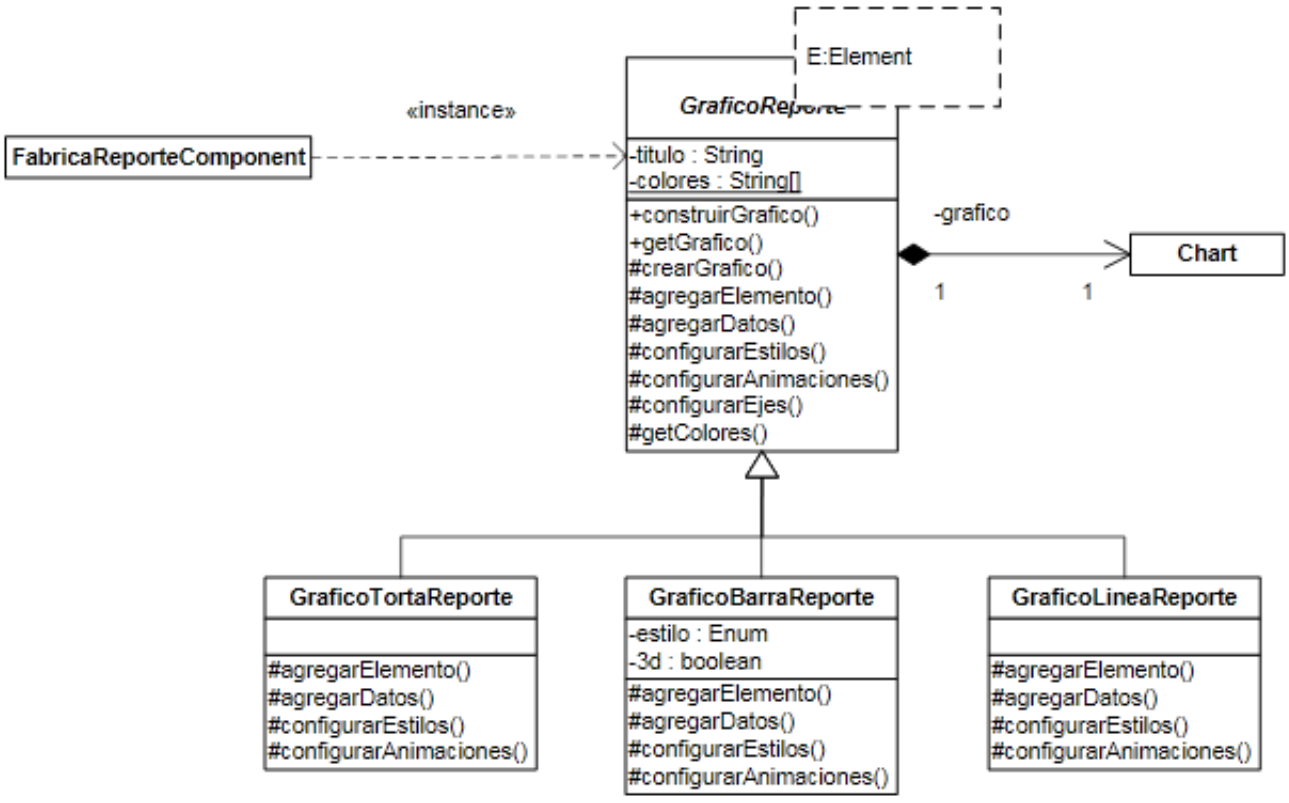


Figura 7.26: Clases para la visualización de los gráficos de los reportes en la GUI Web

- **Pruebas:** Finalmente las pruebas de esta iteración corresponden a pruebas de integración y de sistema como un todo. Para tal fin, se definen flujos de procesos

para las pruebas de manera tal que se contemplen la mayor parte de los componentes del sistema. Los casos se dividen en dos grupos: pruebas del cliente y pruebas del servidor.

Por una parte el cliente interactúa por completo con la extensión, instalándola, configurándola e iniciando la recolección de estadísticas. Luego de la ejecución exitosa de las pruebas se obtiene la aceptación del cliente con respecto a la aplicación cliente.

Por otra parte el cliente interactúa de manera general con todas las páginas de la aplicación web, creando proyectos, ajustando sesiones de proyectos y desplegando reportes estadísticos. Cabe indicar que en este punto se detecta un error al momento de realizar la carga del reporte de estadísticas globales, ya que se muestra de manera incompleta. Luego se procede a la corrección del error de visualización, y se ejecutan nuevamente las pruebas, las cuales resultaron exitosas y se obtiene la aceptación del cliente de la aplicación en general.

Capítulo 8

Caso de Estudio

En el presente capítulo se definen los parámetros de configuración establecidos en la aplicación para iniciar el proceso de recolección de información. Para ello fue instalada una extensión cliente en 12 computadores utilizados por el personal de la División de Control de Estudios, de la Facultad de Ciencias de la UCV. Dicha extensión fue ejecutada bajo las versiones 3.5 y 3.6 del navegador Mozilla Firefox.

Para el presente caso, la recolección se realiza en el horario de trabajo establecido por la División de Control de Estudios, en el cual el personal ejecuta una serie de actividades, de Lunes a Viernes de 8:30am a 12:00 y de 1:30 pm a 4:30pm. Además el periodo de recolección se inicia el viernes 22/10/2010, hasta el viernes 05/11/2010, es decir, tuvo una duración de 2 semanas.

Cabe destacar que dicha recolección fue aplicada sobre el sistema de Gestión Académica módulo CONEST Administrativo, utilizado el personal de la División de Control de Estudios para operaciones de índole administrativa en el ámbito académico.

Antes de inciar la recolección se realiza la configuración del proyecto CONEST Administrativo en el Sistema Servidor, indicando la información básica acerca de dicho proyecto, y las estadísticas a ser recolectadas por la aplicación cliente, como se visualiza en las figuras 8.1 y 8.2, expuestas a continuación :

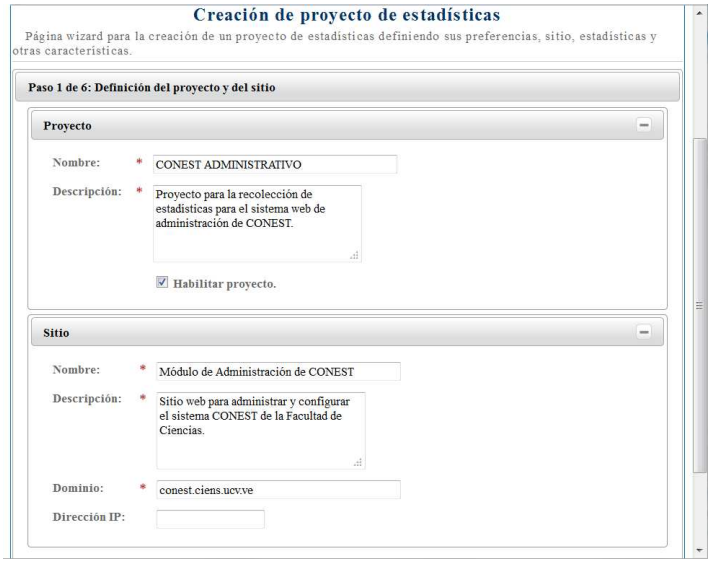


Figura 8.1: Ingreso de información - Creación de Proyecto de Estadísticas

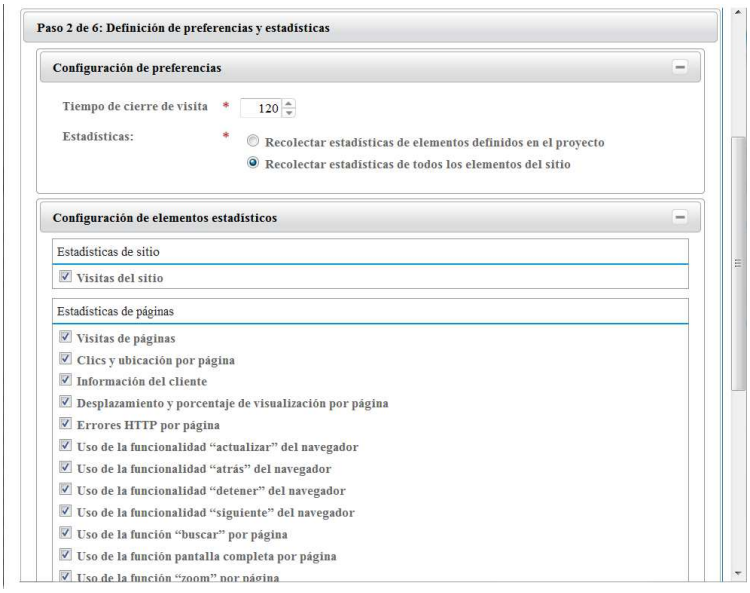


Figura 8.2: Definición de Preferencias y estadísticas

Para iniciar el proceso de recolección de datos se procede a instalar la extensión cliente en cada una de los equipos mencionados, ejecutando el archivo con extensión .xpi. Una vez culminada dicha instalación, se visualiza lo mostrado a continuación en la figura 8.3:

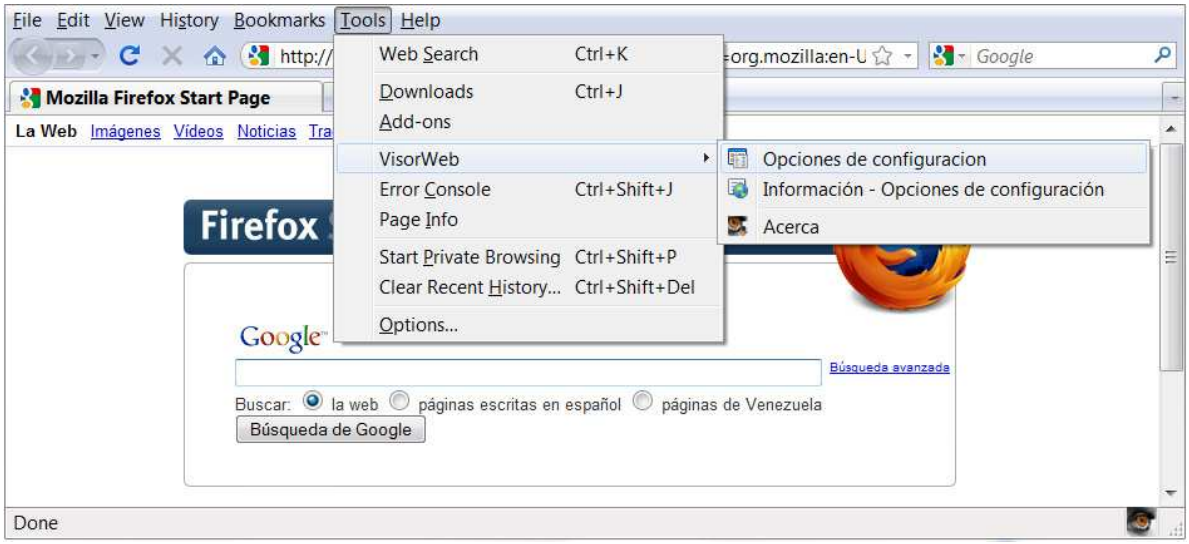


Figura 8.3: Opciones de configuración

Posteriormente, se procede a la configuración del proyecto CONEST Administrativo, para su posterior activación como se ilustra en las figuras 8.4 y 8.5:

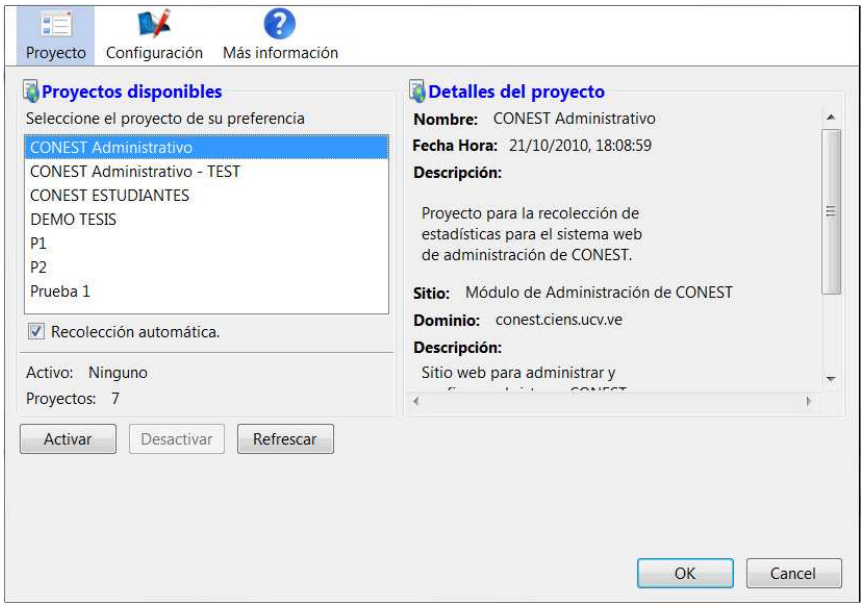


Figura 8.4: Opciones de Configuración - Detalles del Proyecto

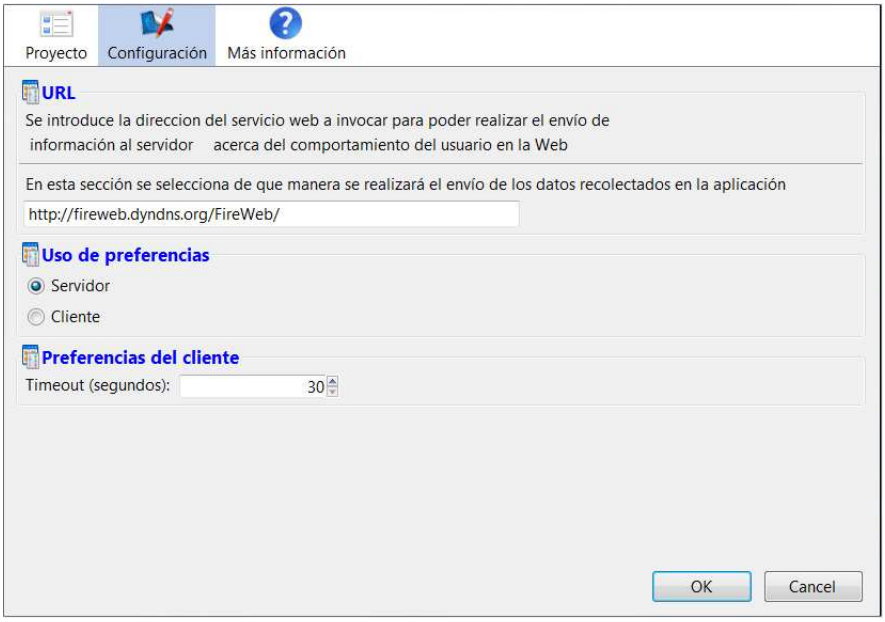


Figura 8.5: Panel Configuración - Opciones

Nota: Para seleccionar el proyecto sobre el cual se realizara la recolección de estadísticas; verifique que el proyecto haya sido seleccionado, y posteriormente presione el botón Activar, en la pestaña Proyecto.

Cabe destacar, que solo puede seleccionar un proyecto para la recolección.

Finalmente se inicia la captura de datos estadísticos, acerca de la interacción del usuario con el sitio Web, en la aplicación cliente, que serán enviados a la aplicación servidor, quien procesa los datos obtenidos y genera un conjunto de reportes, tales como los mostrados en la siguiente sección.

8.1. Resultados

A continuación se proporciona una explicación acerca de los resultados obtenidos en el proceso de recolección de estadísticas por parte de la aplicación y su posterior visualización en la aplicación Web.

8.2. Estadísticas Globales

Figura 8.6: Reporte - Estadísticas Globales

- **Promedio páginas vistas por visita:** Probablemente sea el dato que más se presta para la confrontación, ya que puede asumir distintos significados según el tipo de sitio examinado. Por ejemplo, un alto número de páginas vistas por visita asociado a un breve tiempo de permanencia en las páginas, tendrá un significado diferente según los contenidos de las páginas Web de las que se han detectado tales valores.

  En base a los resultados obtenidos es útil para determinar:

  - Si existe contenido duplicado en las páginas que se han visitado.
  - Existencia de enlaces rotos, mal redirigidos o no disponibles.
  - Claridad de los shortcuts establecidos; los cuales deben ser predictivos y que no induzcan a error de interpretación, es decir, que indiquen hacia donde el usuario va a dirigirse, cuando sea seleccionado, sin que el mismo tenga que reflexionar hacia dónde va a dirigirse o sobre su posible significado.
  - La existencia de información relevante en el sitio Web, que sea original, y que le haga sentir al visitante, que está obteniendo algo que sea de utilidad.

  Se induce que en un promedio de 4 páginas vistas por visita, el usuario realiza sus tareas dentro del sitio Web interactuando con pocas páginas.

- **Promedio clics por visita:** Útil para determinar el grado de actividad o inactividad del usuario existente en la aplicación, o en determinadas páginas web, tiempos de permanencia, entre otros.

- **Promedio tiempo de carga de páginas/visita:** El tamaño y complejidad de los marcos o frames utilizados en las páginas Web, determina en la mayoría de los casos, la velocidad de visualización inicial. Aquellas páginas web que poseen tablas anidadas, archivos CSS, código JavaScript, o imágenes asociadas en formatos no recomendados como el .PNG, retrasan el tiempo que tarda el contenido útil de la página en ser visualizado. Observando los tiempos de carga de una página Web, podremos determinar si el tiempo de respuesta es óptimo, así como el ayudar a determinar un equilibrio entre apariencia y funcionalidad del sitio Web, utilizando técnicas basadas en estándares para optimizar la velocidad de respuesta de las aplicaciones. Entre algunos estándares de diseño podemos mencionar:

  - El posicionar el contenido CSS en la parte superior (cabeza) y el JavaScript en la parte inferior(cuerpo) del HTML permite un procesamiento progresivo del contenido de la misma.
  - El poseer múltiples CSS o JavaScripts que son invocados desde diversas partes de la página puede retrasar el despliegue de contenidos.
  - La presencia de archivos de audio y/o video.
  - Existencia de enlaces construidos con JavaScript o no. En este caso, se recomienda, evitar la construcción de enlaces con dicho lenguaje, ya que en un alto porcentaje de estos casos, es totalmente prescindible; pero, en caso de no ser posible, se debe garantizar su funcionamiento sin tener el Javascript activado.

  Del resultado obtenido anteriormente, podemos que un tiempo de carga de 1,33 segundos, indica una velocidad óptima de carga, lo que cumple uno de los principios básicos de Interacción-Humano-Computador [King, 2008].

- **Tiempo promedio por visita:** El tiempo promedio de duración de la visita de un usuario varía según el tipo de sitio o información que sea consultada por el usuario, por lo que no podemos tomar un valor general.

  Pero dicho valor es muy importante para determinar que tan bien estructurado se encuentra un sitio web y si este facilita su navegación, si el sitio posee contenidos y servicios de interés, así como un lenguaje directo, claro y comprensible para el usuario, algo vital para generar concurrencia por parte de los mencionados, o si por el contrario; ofrecen contenidos irrelevantes que no aportan nada más que ruido visual y estructural a la página.

  Haciendo conjunción con el item especificado anteriormente, un tiempo de 560 segundos, o 9 minutos con 33 segundos, indica un alto nivel de interactividad con el sitio Web.

- **Porcentaje promedio de inactividad por visita:** Es asumido de forma genérica que la lectura por pantalla es bastante más lenta e incómoda que la ejercida sobre el medio impreso, lo que conduce a una percepción no tan clara del contenido. Es por ello, que el usuario no visualiza todo el contenido, sino escanea la página en busca de lo que le interesa y centra su atención en aquellos puntos que le llaman su atención y que cree a primera vista que será la opción más correcta en relación a lo que le interesa.

  El tiempo de inactividad puede indicar:

  - Si el usuario no es capaz de encontrar lo que busca de forma rápida, y por ende tendrá una percepción negativa del sitio web.
  - Si el sitio web presenta problemas de tráfico, carga o de funcionamiento.

  El tomar en cuenta los valores mostrados en la presente métrica contribuye a mejorar la experiencia y usabilidad del usuario, en el proceso de interacción con los sitios Web. No se debe hacer pensar al usuario en su proceso de interacción con una página web, sino que ésta sea tan clara, obvia y fácil de entender, que implique la acción directa y no la pérdida de tiempo en su intento de exploración y comprensión de lo que se le ofrece por pantalla.

  Un porcentaje de 31,59 % indica un alto grado de actividad por parte del usuario con el sitio Web, lo cual de forma implícita indica que el usuario no debe esperar una gran cantidad de tiempo, por la carga de recursos asociados a una página luego de que es mostrada inicialmente.

- **Promedio errores HTTP por visita:** Cada objeto único en una página web requiere un viaje de ida y vuelta al servidor, es decir, una solicitud HTTP y una respuesta. Cada objeto presenta retrasos indeterminados. Generalmente, cuando el número de objetos es mayor que cuatro, son los objetos de la página los que tienden a determinar los tiempos de descarga.

  Para minimizar el número de peticiones HTTP, se recomienda minimizar el número de objetos en sus páginas web, así como el combinar y convertir archivos gráficos basados con técnicas de CSS.

  El obtener 16 errores HTTP, en promedio por visita, puede indicar que sería recomendable revisar los elementos asociados al sitio Web de CONEST Administrativo, tales como: imágenes, enlaces, código javaScript, entre otros.

8.3. Resolución de Pantalla

Figura 8.7: Reporte - Resolución de Pantalla

De lo antes expuesto se deduce lo siguiente:

1. 1024 x 768 vistas 310 veces, correspondiente al 22 % de las vistas.
2. 1152 x 864 vistas 417 veces, correspondiente al 29 % de las vistas.
3. 1280 x 1024 vistas 434 veces, correspondiente al 30 % de las vistas.
4. 1280 x 720 vistas 279 veces, correspondiente al 19 % de las vistas.

De lo que se concluye que la visualización de las páginas deben estén acorde a las siguientes resoluciones: 1280 x 1024 y 1152 x 864, que constituyen las más empleadas por los usuarios.

8.4. Textos buscados

Figura 8.8: Reporte - Textos buscados

En el presente gráfico, se visualizan los textos más buscados en la página de menú principal de CONEST Administrativo. Con los datos observados, se puede obtener:

- 1. **estado:** Con la que se realizó una búsqueda de 30 veces, y la cual esta relacionada con el estado de la calificación o el estado de la inscripción del estudiante.
- 2. **foto:** Con la que se realizó una búsqueda de 18 veces, de la cual puede concluirse que los usuarios intentan ubicar la opciones de Agregar y/o Administrar foto del estudiante.
- 3. **horario:** Con la que se realizó una búsqueda de 14 veces, de lo que puede inferirse que se realizan la búsqueda de opciones asociadas al horario, tales como: Consultar horarios por Semestre.
- 4. **tesis:** Con la que se realizó una búsqueda de 10 veces, que está asociada a la búsqueda de opciones asociadas al trabajo especial de grado.
- 5. **requisito:** Con la que se realizó una búsqueda de 8 veces, la cual suele estar asociada a las opciones de: Consultar requisitos académicos de un estudiante, consultar requisitos a todos los estudiantes inscritos en tesis y consultar requisitos de una materia para un estudiante.
- 6. **CALIFICAR es, calificar estud:** Con la que se realizó una búsqueda de 9 veces, y que se encuentra asociada a la opción Calificar estudiante.

Por lo antes expuesto, se recomienda ubicar las opciones asociadas a las palabras mencionadas, en la parte superior de la página o del menú principal, a objeto de incrementar la interacción por parte del usuario, y por ende el obtener un acceso más rápido a las opciones indicadas.

8.5. Uso de Enlaces

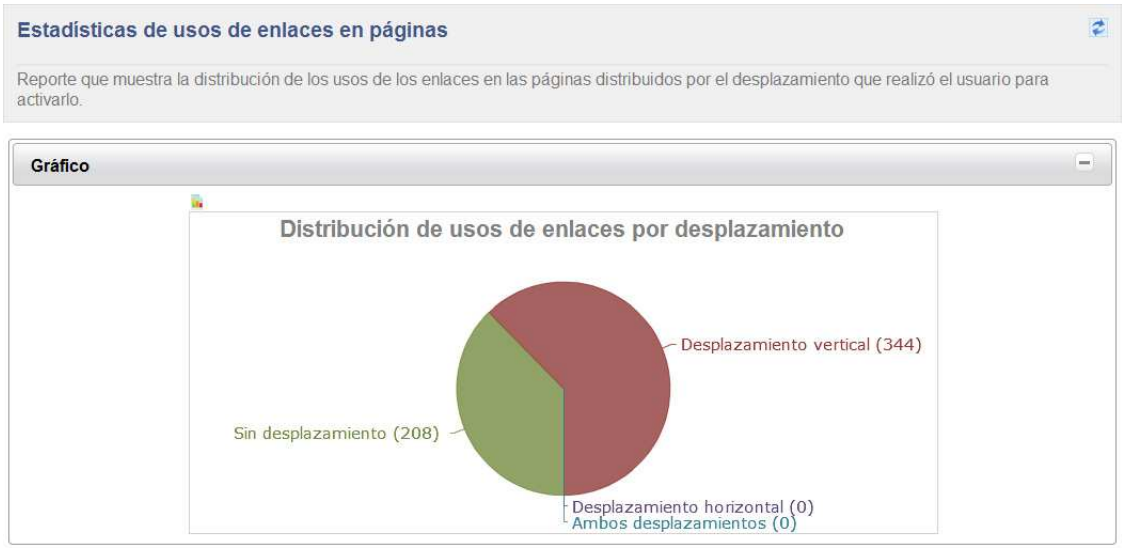


Figura 8.9: Reporte - Uso de Enlaces

El presente reporte (ver figura 8.9), muestra el uso de los enlaces de la página del menú principal de CONEST Administrativo, así como los desplazamientos realizados para su ubicación, a través del uso de las barras de desplazamiento del navegador.

Del gráfico expuesto, se concluye que el 62,32 % de los enlaces requieren de un desplazamiento vertical, lo que contradice uno de los principios de Interacción-Humano-Computador, que establece que los mencionados deben estar ubicados en la parte superior de la página; por lo que se sugiere tomar en cuenta dicho principio, para la re-ubicación de los enlaces asociados a la aplicación.

Aunado a esto, se puede apreciar que los enlaces más utilizados que necesitaron de un desplazamiento vertical para su ubicación, se pueden mencionar:

1. **desbloquear solicitudes de constancias:** 42 veces
2. **calificar estudiante:** 36 veces
3. **modificar inscripción:** 26 veces
4. **modificar número de depósito bancario:** 22 veces

5. **solicitudes pendientes de trabajo especial de grado:** 21 veces
6. **imprimir constancia de estudio:** 15 veces
7. **imprimir constancia de culminación de estudios:** 11 veces

8.6. Usos Comandos del Navegador

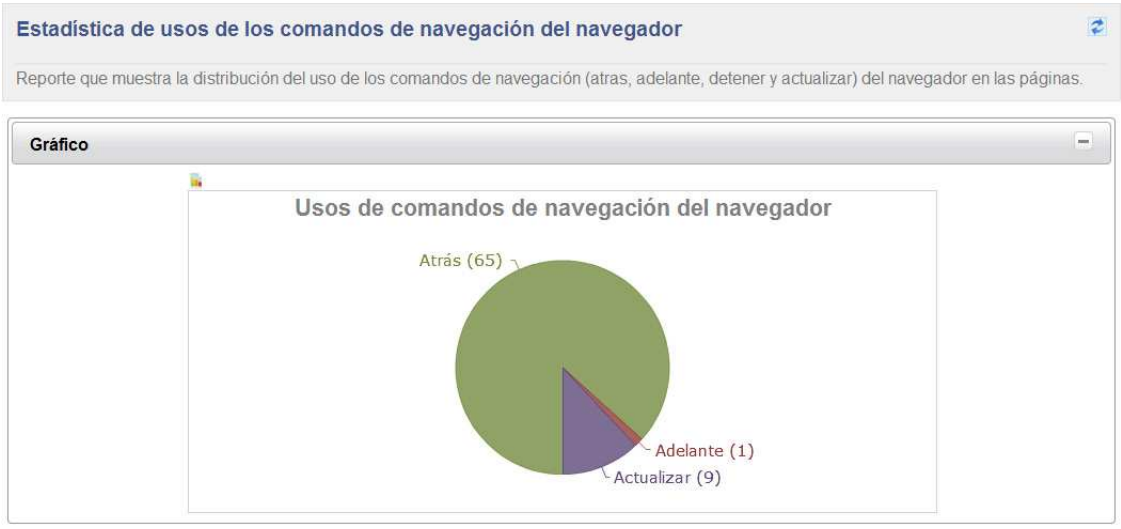


Figura 8.10: Reporte - Usos Comandos del Navegador

Del presente gráfico, se observa que de las 700 visitas realizadas al sitio de CONEST Administrativo, solo se evidencian 75 usos de los botones principales del navegador Mozilla Firefox, tales como: Atrás, Adelante, Actualizar y Detener.

El uso de los mencionados se resume de la siguiente manera:

1. **Atrás:** 65 veces, correspondiente al 86,67 % del total de usos.
2. **Adelante:** 1 vez, correspondiente al 1,33 % del total de usos.
3. **Actualizar:** 9 veces, correspondiente al 12 % del total de usos.
4. **Detener:** 0 veces, correspondiente al 0 % del total de usos.

De lo antes expuesto, se concluye que en general el sitio de CONEST Administrativo, ofrece enlaces de navegabilidad entre páginas, que son muy utilizados por los usuarios, en lugar de los botones de navegabilidad asociados al navegador. Además se observa que el enlace que sustituye al botón Atrás del navegador, es el enlace denominado **Regresar a Principal**, como se visualiza en la figura 8.11, expuesta a continuación:

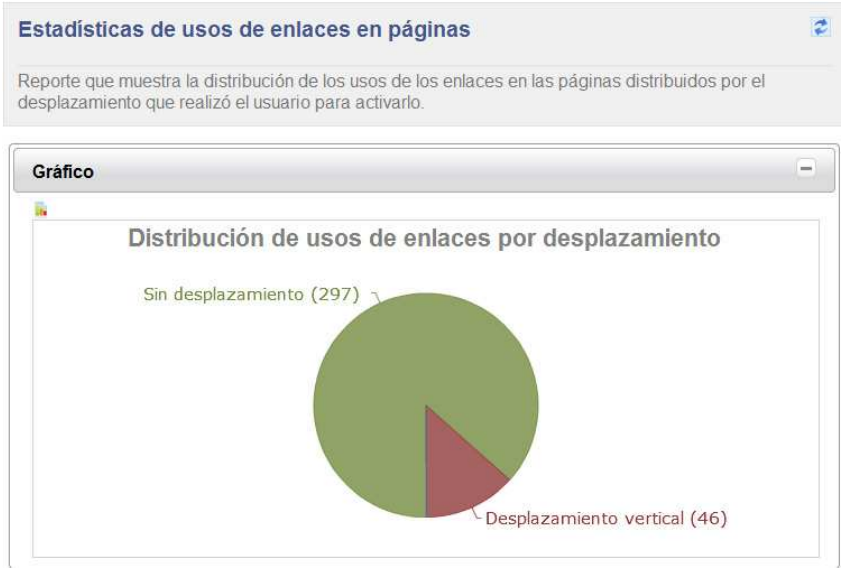


Figura 8.11: Estadísticas de usos de enlaces en páginas

En la figura expuesta(ver figura 8.11), se observa que el enlace **Regresar a Principal**, constituye el enlace más utilizado, con un total de 343 usos, seguido del enlace **(Cambiar)**, lo que indica que el usuario debe regresar al menú principal de la aplicación, si desea interactuar con alguna otra opción asociada al aplicativo.

Por ello, se sugiere la creación de un menú que pueda ser utilizado en cualquier parte del aplicativo, para mejorar la navegabilidad del usuario en dicho sitio.

Además se evidencia una interactividad continua del usuario con las páginas del aplicativo, de lo que se infiere que existe interés por parte del usuario en navegar en el mismo.

Esto es debido al poco uso de los botones de navegación (menos de 80 usos) de las 700 visitas realizadas al sitio.

Capítulo 9

Conclusiones

La investigación realizada para establecer el marco teórico, la adaptación del método XP, y la posterior elaboración de una aplicación permiten en el presente trabajo especial de grado presentar como resultado un software para la visualización de estadísticas en la web, orientadas al usuario.

Gracias a la investigación y a la ejecución de las actividades en la adaptación del método XP, se logra cumplir con los objetivos propuestos, logrando crear una herramienta de Estadísticas Web que obtiene información acerca del interacción del usuario con la página y el navegador.

Por otra parte, se realizan adaptaciones sobre el método XP, pero durante la codificación del sistema, se comienza a laborar durante 40 horas a la semana, dedicando de 2 a 3 horas diarias en promedio a la ejecución de las actividades propuestas, y en algunas ocasiones no se labora durante algunas semanas, lo que trae como consecuencia que el desarrollo del presente producto se extienda, más de lo previsto, obteniendo 17 iteraciones en el marco aplicativo.

Del método XP, se aplica el trabajo en parejas, el cual se realiza la mayor parte del tiempo, mas no en su totalidad, lo que permite el incremento de las habilidades de los programadores. Además se mantiene una comunicación directa con el cliente para determinar los requerimientos funcionales del sistema a desarrollar, logrando orientar en buen término el desarrollo del aplicativo. Al finalizar cada iteración, se realizan entregas de versiones parciales del sistema, que son desarrolladas de forma incremental, y en las cuales se aplica la reutilización de código, lo que ayuda a agilizar la construcción de la herramienta mencionada, no haciendo énfasis en la documentación, sino en su funcionamiento; hasta obtener el producto final.

Respecto al despliegue de los gráficos de los reportes estadísticos, se tuvo que realizar la integración entre la librería de OpenFlashChart 2 y el framework para el desarrollo de aplicaciones Web Java Server Faces(JSF) 1.2, para lo cual se realizó una investigación

acerca de la creación de componentes en JSF, de manera que pudiera ser incrustado un gráfico OpenFlashChart 2. Posteriormente se pudo realizar la integración entre las librerías y el framework descrito de forma exitosa.

Durante la construcción del producto se tuvo dificultad en la obtención de información acerca del funcionamiento de las interfaces **XPCOM**, para poder establecer la comunicación entre las funcionalidades del aplicativo y las funciones del núcleo del navegador Mozilla Firefox, que permiten la captura de información de datos estadísticos de interés, lo que trajo como consecuencia la realización de pruebas de ensayo y error a ciegas para tratar de entender como funcionaban las mencionadas.

Aunado a lo antes expuesto, la mayor parte del aprendizaje acerca de la instalación, funcionamiento de una extensión, creación de interfaces de usuario, y el desarrollo de las funcionalidades contenidas en la aplicación cliente, fue producto de revisar el código fuente de otras extensiones.

En la recolección de datos estadísticos realizados, se puede apreciar que el sitio CONEST módulo Administrativo, ofrece enlaces de navegabilidad entre páginas, que son muy utilizados por los usuarios, en lugar de los botones de navegabilidad asociados al navegador. Sin embargo, el enlace Regresar a Principal, constituye el enlace más utilizado, lo que indica que el usuario debe regresar al menú principal de la aplicación, si desea interactuar con alguna otra opción asociada al aplicativo. Por ello, se sugiere la creación de un menú que pueda ser utilizado en cualquier parte del aplicativo, para mejorar la navegabilidad del usuario en dicho sitio.

También se pueden apreciar altos tiempos de permanencia en el sitio y tiempos de carga óptimos, ya que no se debe esperar una gran cantidad de tiempo por la carga de recursos asociados a una página, luego de que es mostrada inicialmente. Sin embargo se recomienda revisar los recursos y enlaces asociados a cada una de las páginas del sitio Web, debido al número de errores HTTP encontrados, que pueden estar asociados a elementos del sitio Web, tales como: imágenes, enlaces, código javascript, entre otros.

Adicionalmente se observa que en la página del menú principal existen enlaces como por ejemplo, "desboquear solicitudes de constancias", y çalificar estudiante", que se consideran, deben estar ubicados en la parte superior de la página, debido a que son muy utilizados y los usuarios ubican los mencionados utilizando otros mecanismos tales como: búsqueda de texto, o mediante el uso de barras de desplazamiento.

Con el software desarrollado, se provee una solución de menor costo, de fácil manipulación que no requiere de personal con entrenamiento especializado, y que permite recolectar estadísticas enfocadas en el comportamiento del usuario, tales como: tiempos de carga de páginas y sus recursos, usos de enlaces, uso de las funciones de búsqueda de texto y navegación del navegador, tiempos de llenado de un formulario y de sus campos

asociados, entre otras.

Dicha aplicación puede ser utilizada en campos como: Estadística, Interacción-Humano-Computador (IHC), Ingeniería del Software, Publicidad y Mercadeo, Minería de Datos (mediante el análisis e interpretación de la data obtenida), el campo de los negocios, la Computación, entre otros; para contribuir a realizar mejoras relacionadas a la interacción del usuario, diseño, funcionalidad, y rendimiento de las aplicaciones, a la construcción de software de calidad; la predicción de patrones de comportamiento por parte de los usuarios mientras navega por la Web, el determinar hábitos de compra por parte de los mencionados; para el análisis de los datos obtenidos, y proceder a la creación de Sistemas Expertos.

El T.E.G generó un conjunto de aprendizajes y experiencias que contribuirán al desenvolvimiento profesional, entre las cuales cabe destacar: La interacción con un cliente para la definición formal de requerimientos y objetivos para la entrega de versiones del software, la aplicación de un modelo de desarrollo, así como la utilización de diversos patrones de diseño tales como Observer, Prototype, Singleton, DAO, Filter, Abstract Factory, Factory Method, Flyweight, Decorator, Strategy, Template Method, Composite View; el uso de nuevas tecnologías; el rehacer código con la finalidad de ajustar el mismo para cumplir con las espectativas del cliente, y el tener un período de pruebas para la validación de cada una de las funcionalidades de la aplicación, entre otras.

Capítulo 10

Recomendaciones

Respecto a la aplicación cliente se recomienda el permitir la recolección de estadísticas de varios proyectos, el implementar la extensión cliente en otros navegadores como Internet Explorer y Google Chrome, y mantener al día las actualizaciones de la extensión con respecto a la versión del navegador.

Respecto a la aplicación servidor se recomienda el permitir la modificación de proyectos en la aplicación Web, y el almacenamiento de los gráficos estadísticos visualizados, en formato Word y PDF.

Bibliografía

[3DStats, 2004a] 3DStats (2004a). ¿ What are your visitors doing while they are there? http://www.3dstats.com/. [citado en p. 53]

[3DStats, 2004b] 3DStats (2004b). ¿How Does It Work? http://www.3dstats.com/. [citado en p. 53]

[3DStats, 2004c] 3DStats (2004c). Live Demo. http://www.3dstats.com/cgi-bin/showuni1.cgi?usr=00000001P000. [citado en p. 54, 56]

[Alex, 2007] Alex, R. (2007). *RESTful Web services: The basics.* Copyright IBM Corporation 1994,2007. e-mail: arodrigu@us.ibm.com. [citado en p. 43]

[Analytics, 2009] Analytics, G. (2009). Google Analytics. http://www.google.com/analytics/es-ES/. [citado en p. 45, 46, 47, 48, 49]

[Apache Software, 2010] Apache Software, F. (2010). Logging Services. http://logging.apache.org/log4j/1.2/. [citado en p. 65]

[Chris Chalk, 2007] Chris Chalk, Ed Burns, J. H. (2007). JavaServerFaces: The Complete Reference. http://www.amazon.com/JavaServer-Faces-Complete-Reference/dp/0072262400/. [citado en p. 65]

[Costello, 2002] Costello, R. L. (2002). Building Web Services the REST Way. http://www.xfront.com/REST-Web-Services.html. roger.costello@gmail.com. [citado en p. 43]

[Erich Gamma, 1994] Erich Gamma, Richard Helm, R. J. J. V. (1994). Design Patterns: Elements of reusable object-oriented software. http://amzn.to/dnn9Lt. [citado en p. 64, 65]

[EstadisticasGratis, 2004a] EstadisticasGratis (2004a). Calidad del Color. http://www.estadisticasgratis.com/stats.php?id_project=39&report_name=Browsers#color. [citado en p. 53]

[EstadisticasGratis, 2004b] EstadisticasGratis (2004b). Estadísticas y contadores gratis para tu web. http://www.estadisticasgratis.com/. [citado en p. 49]

[EstadisticasGratis, 2004c] EstadisticasGratis (2004c). Tráfico Diario. http://www.estadisticasgratis.com/stats.php?id_project=39&report_name=TraficDay. [citado en p. 51, 52]

[EstadisticasGratis, 2004d] EstadisticasGratis (2004d). Tráfico Horario. http://www.estadisticasgratis.com/stats.php?id_project=39&report_name=TraficHour. [citado en p. 49]

[Feldt, 2007] Feldt, K. C. (2007). Programming Firefox : Building Applications in the Browser. http://www.amazon.com/Programming-Firefox-Building-Internet-Applications/dp/0596102437//ref=sr_1_1?ie=UTF8&s=books&qid=1289316386&sr=1-1#_. [citado en p. 17]

[Fielding, 2000] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm. [citado en p. 65]

[Firefox, 2007a] Firefox, M. (2007a). MDN XPConnect. https://developer.mozilla.org/en/XPConnect. [citado en p. 20]

[Firefox, 2007b] Firefox, M. (2007b). Mozilla Developer Center. The Joy of XUL. https://developer.mozilla.org/en/About_JavaScript. [citado en p. 28, 29]

[Firefox, 2007c] Firefox, M. (2007c). Mozilla Developer Center. XUL. Gecko. https://developer.mozilla.org/es/Gecko. [citado en p. 20]

[Firefox, 2008a] Firefox, M. (2008a). Extensiones. https://developer.mozilla.org/es/Extensiones. [citado en p. 20]

[Firefox, 2008b] Firefox, M. (2008b). Mozilla Developer Center. Uso de Componentes xpcom. https://developer.mozilla.org/index.php?title=Es/Creacinentes_XPCOM/Uso_de_Componentes_XPCOM. [citado en p. 20]

[Firefox, 2009a] Firefox, M. (2009a). Mozilla Developer Center. JavaScript. https://developer.mozilla.org/es/JavaScript. [citado en p. 19, 24]

[Firefox, 2009b] Firefox, M. (2009b). Firefox architecture is very similar to web pages that use dynamic HTML. http://www.soi.wide.ad.jp/class/20070050/slides/01/index_15.html. [citado en p. 17, 18, 19]

[Firefox, 2009c] Firefox, M. (2009c). Mozilla Developer Center. CSS. https://developer.mozilla.org/es/CSS. [citado en p. 19]

[Firefox, 2009d] Firefox, M. (2009d). Mozilla Firefox Extension Development Tutorial. http://www.soi.wide.ad.jp/class/20070050/slides/01/index_10.html. [citado en p. 18]

[Firefox, 2009e] Firefox, M. (2009e). XPIDL. https://developer.mozilla.org/en/XPIDL. [citado en p. 20]

[Firefox, 2009f] Firefox, M. (2009f). XUL. https://developer.mozilla.org/es/XUL. [citado en p. 19, 25]

[Gavin king, 2009] Gavin king, Christian Bauer, M. R. A. E. B. S. E. (2009). Hibernate Reference Documentation. http://docs.jboss.org/hibernate/core/3.5/reference/en-US/pdf/hibernate_reference.pdf. [citado en p. 64]

[King, 2008] King, A. B. (2008). Website Optimization. http://my.safaribooksonline.com/book/web-development/9780596515089/firstchapter. [citado en p. 102]

[librosweb.es, 2009] librosweb.es (2009). Especificaciones oficiales. http://www.librosweb.es/javascript/capitulo1/especificaciones_oficiales.html. [citado en p. 22, 23]

[librosweb.es, 2010] librosweb.es (2010). Mozilla Firefox. https://developer.mozilla.org/en/About_JavaScript. [citado en p. 23]

[maestros del web, 2007] maestros del web (2007). Herramientas para medir las estadisticas de tu web. http://www.maestrosdelweb.com/editorial/herramientas-para-medir-las-estadisticas-de-tu-web/. [citado en p. 44]

[Oracle, 2010] Oracle (2010). The Java EE 6 Tutorial. http://download.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf. [citado en p. 64, 65]

[PrimeFaces, 2009] PrimeFaces (2009). Primefaces. http://www.primefaces.org/. [citado en p. 65]

[Producciones, 2008] Producciones, C. (2008). JavaScript-DHTML. Creación de Páginas Web Dinámicas. http://www.ceneac.com.ve/. [citado en p. 22, 23]

[Scribd, 2008] Scribd (2008). PRESENTACION XUL. http://www.scribd.com/doc/37886018/PRESENTACION-XUL. [citado en p. 25]

[Shailesh K. Mishra, 2007] Shailesh K. Mishra, S. E. I. (2007). RESTful Web services and their Ajax-based clients. http://www.ibm.com/developerworks/webservices/library/ws-restajax/. [citado en p. 43]

[WikiBooks, 2009a] WikiBooks (2009a). XUL/Introducción. http://es.wikibooks.org/wiki/XUL/Introduccion. [citado en p. 25, 31]

[WikiBooks, 2009b] WikiBooks (2009b). XUL/Primer ejemplo. http://es.wikibooks.org/wiki/XUL/Primer_ejemplo. [citado en p. 31]

[xprogramming, 2001] xprogramming (2001). What is Extreme Programming? http://xprogramming.com/xpmag/whatisxp. [citado en p. 62]