Universidad de las Ciencias Informáticas
Facultad 8

Título: Arquitectura de la capa de presentación para el proyecto CICPC.

Trabajo de Diploma para optar por el título de

Ingeniero en ciencias Informáticas

Autor: Marcel Mesa Martínez

Tutor: Ing. Maikell Frómeta Flores

Julio 2008

AGRADECIMIENTOS

En el trascurso de mi vida como estudiante he pasado por diversas situaciones difíciles. He podido seguir adelante gracias a un grupo de personas que me han ayudado en los momentos más críticos de mi vida. A estas personas les debo todo mi ser y mi existencia. Le doy las gracias a mi tía Sucel Martínez Mengana, Lilian Torres Bravo, Zeida Bravo Mengana y a toda mi familia en general; y en especial a mi mamá Maricela Martínez Mengana por amarme y quererme.

También les doy las gracias a todos los profesores que he tenido en todos estos años de estudiante, porque sin sus pequeños aportes no estaría escribiendo estas líneas. Les doy las gracias a mis amigos (Leudys Romero, Alex Miranda, Yennis Portales, Guillermo Portuondo, David Aldana, Osvaldo Muguercía, Jorge Luis Oliva, Aned Corzo, Yunieski Zamora, Ana María Fornaris, Yutmila Soto), a mis compañeros de apto y grupo (Alexis Naranjo, Enrique José Altuna, Amado Soria, Carlos Estrada, Dayrán Álvarez, Hennry Carmenate, Oliday Almenarez) por aportar de una forma u otra un poquito de su ser en mi persona y sobre todo en la realización de este trabajo de diploma.

Les doy las gracias a la Revolución y a la Universidad de las Ciencias Informáticas por permitirme graduarme como Ingeniero Informático y así posibilitarme la realización de uno de mis sueños.

DEDICATORIA

Con infinito amor...

A mi madre y familia por estar siempre a mi lado, por ayudarme en los momentos más difíciles de mi vida y amarme con todo su ser.

RESUMEN

El proyecto CICPC está desarrollando el nuevo sistema SIIPOL para el Cuerpo de Investigación Científicas Penales y Criminalística (CICPC) de la República Bolivariana de Venezuela. Entre los principales objetivos del sistema se encuentra brindar una interface de usuario operativa y amigable, que permita una mejor interacción entre los usuarios y el sistema. Al ser un sistema que debe soportar un elevado número de usuarios concurrentes, se hace necesario optimizar y reducir la mayor cantidad de peticiones al servidor por parte del navegador, para brindar mejor los servicios que oferta el sistema. El nuevo sistema SIIPOL es una aplicación orientada a la Web, y la capa de presentación de dicha aplicación es compleja. Se hace necesario controlar y organizar el proceso de desarrollo de dicha capa; definir mecanismos de codificación para los desarrolladores de la presentación, análisis de componentes y framework de terceros para acelerar el desarrollo del sistema y buscar mecanismos que permitan que los servicios que brinda el sistema a los usuario sean lo más eficaz y rápido posibles.

Para cumplir estas necesidades se realizó un estudio profundo en las diferentes tecnologías y prácticas de desarrollo de aplicaciones Web enfocadas a la capa de presentación. También se analizaron las mejores técnicas y mecanismos de optimización de frontend de una aplicación orientada a la Web.

Como resultado se obtuvo elementos de arquitectura que respondían a las necesidades de la aplicación y del grupo de desarrollo, posibilitando que se cumplieran los objetivos del proyecto y del sistema en general.

ÍNDICE

Introducción

El proyecto CICPC actualmente desarrolla un sistema policial para el Cuerpo de Investigaciones Científicas Penales y Criminalísticas (CICPC) de la República Bolivariana de Venezuela. Entre los principales objetivos de este sistema se encuentra brindar una interfaz de usuario amigable y operativo, de manera que facilite la interacción de los usuarios con el sistema. Al ser el mismo un sistema que debe soportar un elevado número de usuarios concurrentes se hace necesario optimizar y reducir la cantidad de peticiones desde los clientes al servidor. Por otro lado, dado que la capa de presentación de cualquier aplicación de gestión es la de mayor complejidad y el número de desarrolladores de capa de presentación para el mencionado proyecto es grande, se hace necesario, para organizar y controlar el proceso desarrollo, definir estándares de codificación, evaluar y reutilizar componentes y frameworks de presentación de terceros; además de desarrollar componentes propios que encapsulen y abstraigan la complejidad subyacente de los frameworks y componentes de terceros utilizados. Todo esto implica que el esfuerzo de todo el equipo se centre en definir y codificar más sobre la perspectiva y expectativas del usuario final y no en resolver problemas de interacción e integración entre los diferentes frameworks utilizados.

Además de lo planteado anteriormente, se une la naturaleza de la aplicación que debe realizar el mencionado proyecto, el cual es un sistemas de gestión para la Web, donde muchos de estos planteamientos se hacen más evidentes y complejos, provocando un mayor esfuerzo a la hora de definir y realizar las tareas por los desarrolladores de presentación para lograr los objetivos que tiene trazado el proyecto.

Todo lo anterior conlleva a la necesidad de definir elementos de arquitectura que resuelvan las necesidades de la capa de presentación y además se integre con las capas inferiores de la aplicación sin esfuerzo adicional.

En dicho proyecto el análisis y diseño de los casos de uso y el diseño de las interfaces de usuario de las especificaciones de dichos casos de uso no están orientados a las características de una aplicación Web. Además, se evidencia la no definición de componentes de software que resuelvan los problemas más complejos del dominio de la aplicación en la capa de presentación, la falta de estándares que ayuden a linear el esfuerzo de desarrollo en la capa de presentación y el uso de un mecanismo de descripción y renderizado de las vistas que permita ganar en velocidad de desarrollo y en eficiencia.

Los equipos de desarrollo en la capa de presentación han experimentado baja productividad en el desempeño de sus tareas producto de los elementos citados anteriormente. Han tenido que realizar un esfuerzo mayor para lograr los objetivos de su trabajo y los del proyecto en general. Además, todo esto conlleva que el nivel de errores a la hora de codificar sea mayor.

De lo anterior se puede definir como **problema científico:**¿Cómo mejorar el proceso de desarrollo en la capa de presentación para el nuevo sistema SIIPOL mediante la definición de mecanismos para regir el esfuerzo de codificación en la capa de presentación, de componentes que resuelvan los problemas más comunes y complejos del dominio de la aplicación en la capa de presentación y el uso de frameworks de terceros que permitan la descripción y renderizado de las vistas para ganar en velocidad de desarrollo y eficiencia?

Para resolver este problema se realiza el presente trabajo, que tiene como **objeto de estudio:** Frameworks de presentación y mejores prácticas para el desarrollo de aplicaciones empresariales orientadas a la Web.

Se propone como **objetivo general**: Definir elementos de arquitectura necesarios para resolver las necesidades de la capa de presentación y proporcionar un mecanismo de integración con las capas inferiores.

Dentro de este objetivo general se desglosan los siguientes objetivos específicos:

- Crear estándares de codificación a utilizar por los programadores de la capa de presentación de manera que agilicen el desempeño de los desarrolladores.
- Crear componentes de software a utilizar por los programadores de la capa de presentación de manera que agilicen su trabajo.
- Elaborar mecanismos para optimizar el frontend de la aplicación.
- Elaborar un mecanismo de integración con capas inferiores y adyacentes.
- Realizar definiciones arquitectónicas que permitan organizar y controlar el esfuerzo de los desarrolladores de la capa de presentación para el nuevo sistema SIIPOL.

El objetivo delimita el **campo de acción,** que en este trabajo es: Definición y desarrollo de los elementos arquitectónicos de la capa de presentación para el nuevo sistema SIIPOL.

Esta investigación será guiada por la siguiente **idea a defender:** Mediante el uso de mecanismos de descripción y renderizado optimizados, de acuerdo a las características de la presentación y la definición de mecanismos y componentes de software en la capa de presentación orientados al dominio del sistema, se podrá aumentar la productividad de los desarrolladores de la capa de presentación en el desempeño de sus tareas, aumentar la eficiencia del sistema y así alcanzar los objetivos trazados por el proyecto.

Para cumplir con los objetivos y la situación problemática planteada se propone realizar las siguientes **tareas de investigación**:

- Investigar sobre componentes de software para la capa de presentación de una aplicación de gestión Web.
- Investigar sobre la definición de estándares para la capa de presentación de una aplicación de gestión para la Web.
- Investigar mecanismos para acelerar el desarrollo de la capa de presentación de una aplicación de gestión para la Web.
- Investigar formas que permitan flexibilidad y mantenibilidad de la capa de presentación de una aplicación de gestión para la Web.
- Buscar formas de reducir la cantidad de peticiones del navegador al servidor Web.
- Investigar cómo reducir los tiempos de respuestas de las aplicaciones de gestión orientadas a la Web.
- Investigar sobre la optimización y rendimiento de frontend en las aplicaciones de gestión orientadas a la Web.
- Investigar sobre la tendencia de la Web.
- Investigar sobre las técnicas de desarrollo de aplicaciones Web.
- Investigar sobre las tecnologías para la Web
- Buscar mecanismos que permitan acelerar el renderizado de las vistas de presentación.
- Buscar framework que definan mecanismos de validaciones del lado del servidor en las aplicaciones orientadas a la Web.
- Buscar framework que definan mecanismo de comunicación con capas adyacentes en las aplicaciones de gestión para la Web.
- Buscar librerías que permitan reutilizar funcionalidades en el marco de las aplicaciones orientadas a la Web.
- Investigar las tecnologías utilizadas para la plataforma Java para la Web.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se brinda una aproximación general y detallada de los temas relacionados con las aplicaciones web, un primer acercamiento al concepto de arquitectura de software y todas las tecnologías y prácticas relacionadas con las aplicaciones Web, así como relación del modelo cliente – servidor con las aplicaciones Web.

1.2 World Wide Web o Web. Historia y propósito

La World Wide Web o documentos con referencias cruzadas, en sí mismo, no constituye un concepto nuevo. Las referencias a otros documentos, en forma de notas, existían ya en los escritos medievales. La diferencia es que la Web es más global, rápida y fácil de usar. Todo ello fue posible gracias a los avances tecnológicos de las últimas décadas del siglo XX. (1)

En 1945, el Doctor Vannevar Bush, Director de la Oficina de Desarrollo e Investigación Científica (EE.UU), redactó el articulo "As We May Think" para la revista "The Atlantic Online", en que mostraba su preocupación por la gran cantidad de información que existía y que estaba siendo generada en poco tiempo y los ineficientes sistemas que había para encontrarla. Así, y apoyándose en la tecnología existente de la época, diseño un dispositivo personal, al que llamó "Memex", y que lo concebía como un suplemento íntimo a su memoria. Este aparato permitía a cada individuo almacenar su información en microfilmes, consultarlos eficientemente y, lo que es más importante, dejaba crear vínculos entre los documentos, de modo que durante la lectura de un documento se le recordara al lector que este contenía información que se encontraba en otros documentos. Esta era una visión de lo que pasaría 45 años después.

El término de "hipertexto" fue visto por Ted Nelson en 1965 en su artículo "A File Structure for the Complex" de la revista "The Changing and the Indeterminate". Ted Nelson ideo un módulo para enlazar documentos electrónicos.

La World Wide Web fue creada en 1989 por un informático del CERN (Organización Europea de Investigación Nuclear) llamado Tim Berners-Lee. Era un sistema de hipertexto para compartir información basado en internet, concebido originalmente para servir como herramienta de comunicación entre los científicos nucleares del CERN. Tim Berners-Lee estuvo experimentado con hipertextos desde 1980, año en que diseño y desarrollo Enrique, un programa para almacenar documentos de información y enlazarlos. Enrique se ejecutaba en un entorno multiusuario y permitía

acceder a varias personas a los mismos datos. Tim Berners – Lee le entregó su propuesta al CERN en 1989 y así a finales de 1990 el primer browser de la historia, World Wide Web, ya tenía forma.

Los tres componentes importantes del sistema propuesto eran los siguientes:

- Una interfaz consistente.
- La habilidad de incorporar un extenso rango de tecnologías y diferentes tipos de documentos.
- Un instrumento para leer los documentos en forma universal. Esto significa que cualquier persona en cualquier lugar que esté conectada a la red podrá leer el mismo documento al mismo tiempo que otra persona y podrá hacerlo de forma fácil y eficiente.

Desde entonces se han desarrollado varios lenguajes y protocolos evolucionados dentro de la estructura creciente de Internet. Es importante recordar que el Web es sólo una parte de todo Internet. Mucha gente piensa que la Web en Internet es la misma cosa, pero no están en lo cierto. La World Wide Web es un conjunto de servicios basados en hipermedias, ofrecidos en todo el mundo a través de Internet. Se le llama WWW (World Wide Web - Telaraña de Cobertura Mundial). No existe un centro que administre esta red de información, sino más bien está constituida por diversos servicios que se conectan entre sí a través de referencias en los distintos documentos, por ejemplo, un documento contenido en un computador en Canadá, puede tener referencias a otro documento en Cuba ó a un archivo en Venezuela, o a una imagen en Inglaterra.

La Web es el equivalente a una oficina de una empresa, donde puede poner a disposición pública productos, servicios, áreas de soporte al cliente y en general cualquier información de la empresa. Las informaciones que las empresas y personas ponen a disposición pública a través de la Web están en un lenguaje especialmente diseñado para esto, HTML (HyperText Markup Language). Dicho lenguaje está siendo adoptado como el estándar universal para la creación y distribución de información no sólo en ambientes públicos, sino también en los entornos privados como compañías y corporaciones. Actualmente tenemos una variedad de programas gratuitos y comerciales capaces de crear documentos en HTML de una manera muy fácil y rápida.

No olvidar mencionar que el lenguaje de intercambio HTML y el protocolo de red HTTP se diseñaron para ser realmente muy simple el uso de la World Wide Web.

Hoy, el Web es algo cotidiano para una gran parte de los más de 600 millones de usuarios de Internet que hay en todo el mundo. Sus utilidades son diversas y su impacto en la economía mundial es apreciable. No sólo podemos encontrar en ella documentos de texto, sino que también podemos ver:

- Imágenes.

- Videos.
- Música.

1.3 Modelo Cliente- Servidor

La arquitectura cliente-servidor, también conocido como modelo cliente-servidor, es una forma de especializar terminales y programas de forma que las actividades y tareas que cada uno de ellos realice se efectúe con la mayor eficiencia posible. Dicha arquitectura es cualquier combinación de sistemas que puedan colaborar entre sí para dar a los usuarios toda la información que ellos necesiten, sin que tengan que saber donde está ubicada.

El escenario de un modelo cliente-servidor ocurre cuando un proceso desea un servicio que es proporcionado por otro proceso. Así, el primer proceso le envía un mensaje solicitando ese servicio al segundo proceso. El servicio es ejecutado por el segundo proceso y envía un mensaje al proceso solicitante que ha ejecutado el servicio. El mensaje que envía el primer proceso se llama petición ó *Request* y el mensaje que envía el segundo se llama respuesta o *Response*. El proceso que ejecuta el servicio se llama servidor y el solicitante se llama cliente. En este escenario, el cliente es el que inicia un requerimiento de servicio y el servidor es el que ejecuta dicho servicio y envía una respuesta al cliente que lo ha solicitado.

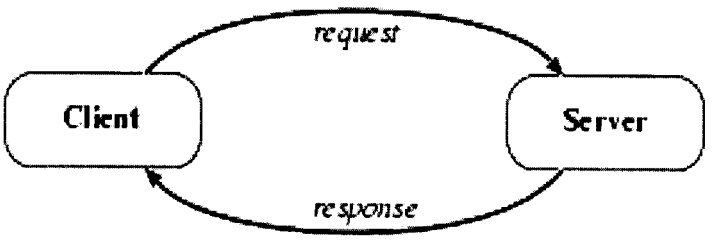


Figura 1.2.1: Modelo cliente – servidor

El modelo cliente-servidor ha de seguir un protocolo de comunicaciones que define:

- Cómo se codifican las peticiones.
- Cómo se sincronizan entre sí los procesos.

Los clientes y los servidores han de estar de acuerdo en cómo se escriben los mensajes, en qué orden van los posibles parámetros de la petición, cuántos bytes ocupan, entre otras cuestiones.

La forma de sincronización dice si el cliente puede seguir adelante justo después de enviar la petición (no bloqueante), o por el contrario, tiene que esperar a que el servidor le envíe una respuesta (bloqueante). Si la comunicación es no bloqueante, habrá que definir un mecanismo para que el cliente pueda saber si la respuesta del cliente está disponible. En esta práctica se adoptará una comunicación bloqueante: el cliente siempre esperará hasta recibir una respuesta del servidor.

El diálogo cliente-servidor es casi siempre bidireccional. Por un lado, el cliente envía información al servidor (el tipo de servicio solicitado más los parámetros); por otro, el servidor devuelve información al cliente (los resultados del servicio, códigos de error en caso de producirse, etc.).

Los elementos principales del modelo cliente-servidor son justamente los elementos llamados cliente, servidor y red de comunicación. (2)

Las características principales que identifican al modelo son:

- Pueden actuar como una sola entidad y también como entidades separadas, realizando actividades y tareas separadas.
- Las funciones del cliente y el servidor pueden estar en plataformas separadas, o en la misma plataforma.
- Un servidor da servicio a múltiples clientes de forma concurrente.
- Cada plataforma puede ser escalable e independiente. Los cambios realizados en las plataformas de los clientes o de los servidores, ya sean por actualización o por reemplazo tecnológico, se realizan de una manera transparente para el usuario final.
- La interrelación entre el hardware y el software están basados en una infraestructura poderosa, de tal forma que el acceso a los recursos de la red no muestra la complejidad de los diferentes tipos de formato de datos y de los protocolos.
- Un sistema de servidores realiza múltiples funciones y tareas, al mismo tiempo que presenta una imagen de un sólo sistema a las estaciones clientes. Esto se logra combinando los recursos de cómputo que se encuentran físicamente separados en un solo sistema lógico, proporcionando de esta manera el servicio más efectivo para el usuario final. También es importante hacer notar que las funciones cliente-servidor pueden ser dinámicas. Un servidor puede convertirse en cliente cuando realiza la solicitud de servicios a otras plataformas dentro de la red.

1.4 Aplicaciones Web

Una aplicación Web es un sitio Web donde la interacción a través del sitio y la incorporación de datos por parte de los usuarios afectan el estado de la lógica del negocio del mismo; es decir, el sitio Web procesa los datos introducido por los usuarios y les muestra a estos un resultado en dependencia de la información introducida. En esencia, una aplicación Web usa un sitio Web como entrada a una aplicación típica. Si no existe un procesamiento diferente en el servidor en dependencia de la petición del usuario, el sitio Web no puede ser considerado como aplicación Web.

Las aplicaciones Web son en estos días unos de los sistemas más comunes que podemos encontrar en la red a través de Internet. Ejemplo de estas aplicaciones Web son los sistemas que brindan la posibilidad de comunicarnos con personas en todo el mundo a través de correo electrónico o mensajería instantánea (o ambos) como GMAIL y HOTMAIL; los sistemas que permiten comprar productos en internet como Amazon; las aplicaciones que permite buscar información en internet como Google o las que permiten estar al día sobre lo que pasa en el mundo de hoy como el sitio oficial de la CNN.

En general una aplicación Web es parecida al modelo cliente-servidor. Las aplicaciones Web constan de un cliente (navegador), un servidor (servidor Web) y una red de comunicación que posibilita la comunicación entre el cliente y el servidor. Aparte de está semejanza con el modelo cliente servidor, las aplicaciones web constan de tres niveles fundamentales: (3)

- Interfaz de usuario.
- Lógica de negocio.
- Datos.

El nivel de interfaz de usuario está compuesto por documentos escrito en lenguaje HTML con los datos que se han obtenido de un servidor web y se muestran a los usuarios mediante un cliente web (generalmente un navegador web).

El nivel de lógica de negocio está constituido por módulos que implementan la lógica de negocio. La lógica de negocio se divide en dos: lógica de la aplicación y lógica del dominio. Esta se ejecuta en un servidor de aplicaciones. En la actualidad los servidores Web funcionan a la vez como servidor de aplicación.

El nivel de datos está integrado por la información, que generalmente está manejado por un sistema de gestión de base de datos, el cuál es accedido mediante el servidor de aplicaciones.

Otra característica de las aplicaciones Web es que está sustentada por tres tipos de servidores:

- Servidor Web.
- Servidor de Aplicaciones.
- Servidor de base de datos.

Para comprender mejor como funciona una aplicación Web se ilustra la siguiente imagen.

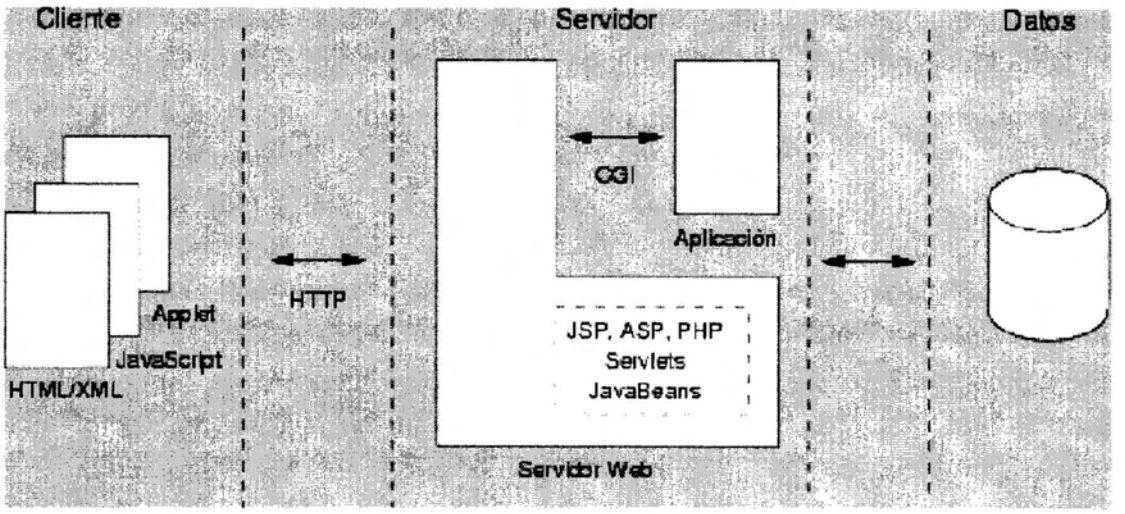


Figura 1.4.1: Elementos de una aplicación Web

1.5 Web 2.0

La Web 2.0 representa en la actualidad la evolución que han tenido las aplicaciones web tradicionales hacia las aplicaciones Web dirigidas al usuario final. Hay que destacar que el término de la Web 2.0 no es en sí misma una tecnología, sino que es una actitud que están adoptando las nuevas aplicaciones Web en el mundo actual. Es un nuevo enfoque con el que se está desarrollando, construyendo y diseñando dichas aplicaciones en nuestros días.

En sus inicios, cuando la Web fue concebida por su autor, Tim Berners-Lee, las páginas HTML sufrían pocas modificaciones o actualizaciones, ya que estaban constituidas por muchos contenidos estáticos; lo que provocaba que la interacción entre estas y el usuario fuera muy escasa, o fuera nula en muchas ocasiones. En pocas palabras, la Web no era dirigida al usuario. En nuestros días esto ha cambiado; las aplicaciones Web que se están concibiendo son dirigidas al usuario, aumentando la interacción entre este y la aplicación.

Se puede decir que la Web 2.0 es la transición que se ha dado de las aplicaciones tradicionales hacia las aplicaciones que funcionan a través de Internet que son dirigidas o enfocadas a los usuarios. (4)

Las aplicaciones Web que son elaboradas, construidas o diseñadas siguiendo el paradigma de la Web 2.0 deben cumplir varios principios, tales como:

- La web es la plataforma.
- La información es el procesador.
- Efectos de la red movidos por una arquitectura de participación.
- La innovación surge de las características distribuidas de los desarrolladores independientes.
- El fin del círculo de adopción de software ("Servicios en beta perpetuo").

Como se ha explicado anteriormente, la Web 2.0 no es una tecnología en sí misma. Para que las aplicaciones web cumplan con el paradigma de la Web 2.0 tienen que usar algunas tecnologías que son las que le dan vida a esta. Algunas de estas tecnologías son:

- XHTML
- CSS
- Java Script
- XML
- AJAX
- FLASH.
- Ruby on Rails
- Servicios Web
- RSS.

Se debe decir que para que una aplicación Web cumpla con el paradigma de la Web 2.0 en su elaboración no debe cumplir todas las tecnologías expresadas anteriormente. Por ejemplo, muchas de las aplicaciones Web actuales no utilizan Ruby on Rails para programar las páginas dinámicas. Ejemplo de esto tenemos a GMAIL, que utiliza el lenguaje Java para este propósito.

Para mayor comprensión de la Web 2.0 se citarán algunas aplicaciones que cumplen con lo estipulado en la Web 2.0, para poder ver la evolución de las aplicaciones tradicionales.

Web 1.0	Web 2.0
Doubleclick	Google AdSense (Servicios Publicidad)
Ofoto	Flickr (Comunidades fotográficas)
Akamai	BitTorrent (Distribución de contenidos)
mp3.com	Napster (Descargas de música)
Britannica Online	Wikipedia (Enciclopedias)

Sitios personales	Blogs (Páginas personales)
Especulación con dominios	Optimización en motores de búsqueda
Page views	Cost per click
CMSs	Wikis (Manejo de contenidos)
Categorías/Directorios	Tagging

Figura 1.5.1: Sitios web basados en la Web 1.0 y la Web 2.0

1.6 AJAX

AJAX (JavaScript y XML Asincrónico) es una técnica de desarrollo Web para elaborar aplicaciones interactivas. (5) Estas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantienen comunicación asincrónica con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página Web sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y eficacia de las aplicaciones basadas en la Web.

AJAX no es una tecnología Web, sino el uso coordinado de diversas tecnologías que en conjunto permiten una mayor rapidez y eficacia para las aplicaciones basadas en la Web.

Hasta hace muy poco tiempo unas de las cosas que distinguían a las aplicaciones de escritorio de las aplicaciones Web era su rapidez de ejecución. A medida que el volumen de datos iba aumentando la lentitud de las aplicaciones se hacía más evidente. Una de las grandes prestaciones de AJAX es acelerar la velocidad de ejecución de las aplicaciones basadas en la Web; haciéndola muy similares a las aplicaciones de escritorio. La clave para alcanzar esta rapidez es la modificación de la comunicación que se estable entre el navegador Web y el servidor.

AJAX es una combinación de varias de las tecnologías existentes:

- XHTML
- CSS
- JavaScript
- DOM

- XMLHTTPRequest
- XML.
- JSON.

XHTML y las hojas de estilos en cascada (CSS) se usan para el diseño que acompaña a la información.

JavaScript permite acceder al DOM para mostrar e interactuar dinámicamente con la información presentada.

El objeto XHTMLRequest es utilizado para intercambiar datos de manera asincrónica entre el navegador y el servidor.

XML y JSON son los formatos que comúnmente se utilizan para la transferencia de datos devueltos por el servidor. También las dos tecnologías pueden ser usadas como formato de transferencia desde el navegador Web y el servidor.

El uso de AJAX en las aplicaciones Web tiene varias ventajas e inconvenientes. Entre las ventajas tenemos que:

- Acelera las aplicaciones basadas en la web.
- Brinda mayor amigabilidad e interactividad de las aplicaciones.
- Mejora el tiempo de respuesta.
- Permite comunicación asincrónica.
- Válido en cualquier plataforma.
- No es difícil de usar.
- Independiente del tipo de tecnología que se utilice en el servidor.

Entre sus inconvenientes tenemos:

- No es soportado por todo los navegadores.
- Los navegadores no obtienen de la misma forma el objeto XMLHTTPRequest.

Para poder comprender las ventajas que aporta el uso de AJAX en las aplicaciones basadas en la Web se compara los escenarios de las aplicaciones Web tradicionales contra los escenarios de las aplicaciones Web que usan AJAX.

En las aplicaciones Web tradicionales, cuando teníamos una pantalla que contenía un formulario para introducir datos por parte del usuario, una vez que este introducía los datos y los enviaba al servidor, la página que enviaba la información tenía que esperar que la información fuera procesada en el servidor. Mientras que la información se procesaba en el servidor la página se quedaba congelada sin permitir que hubiera interacción entre ella y el usuario. Una vez que la información ya estuviera procesada en el servidor la página tenía que recargarse completa para poder mostrarle al usuario la información nueva devuelta por el servidor.

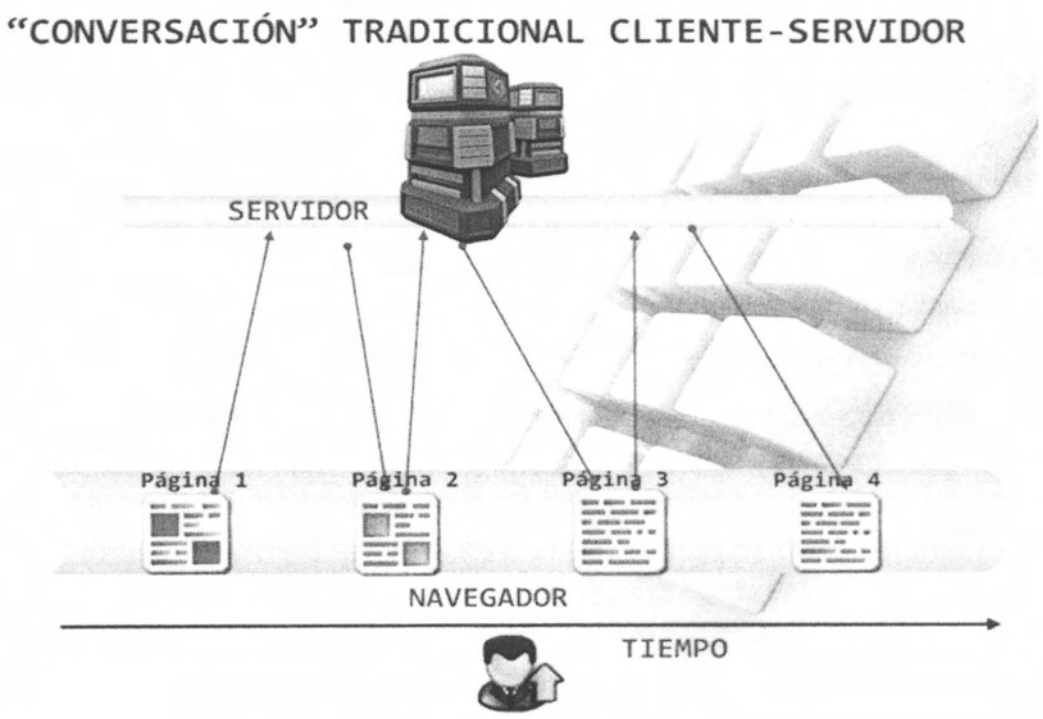


Figura 1.6.1: Aplicaciones Web tradicionales

En las aplicaciones Web que usan AJAX, cuando el usuario introduce los datos en el formulario y los envía al servidor la página Web no se comunica directamente con el servidor. Esa comunicación la realiza el motor AJAX, que es el que espera que la petición sea procesada por el servidor. Una vez terminado el procesamiento de la información por el servidor, el motor AJAX le comunica al navegador que la información ya ha sido proceda y le entrega la información devuelta por el servidor para que este le muestre dicha información al usuario. En este escenario la página Web no se congele ni se recarga completamente, lo que permite que siga habiendo interacción entre el usuario y el navegador Web.

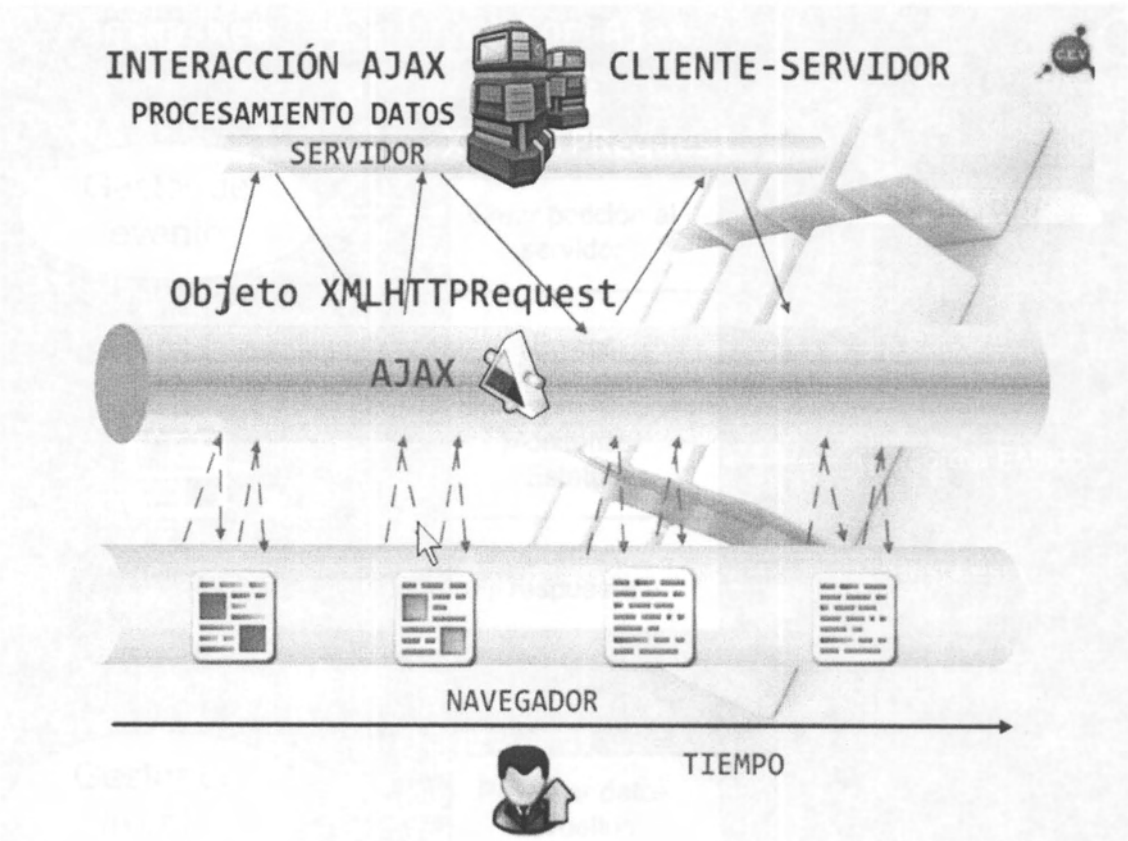


Figura 1.6.2: Aplicaciones Web con AJAX

Un escenario completo de un proceso de una petición AJAX sería el siguiente:

1. La página web, mediante los manejadores de evento, invoca al motor AJAX para que realice una petición HTTP.
2. El objeto HTTPRequest crea la petición al servidor, envía la petición, monitoriza la petición y obtiene los datos devueltos por el servidor.
3. El servidor obtiene la petición, procesa los datos y los devuelve al motor AJAX.
4. El motor AJAX procesa los datos devueltos y se los da a la página.
5. La página web obtiene los datos y actualiza su contenido en dependencia del resultado.

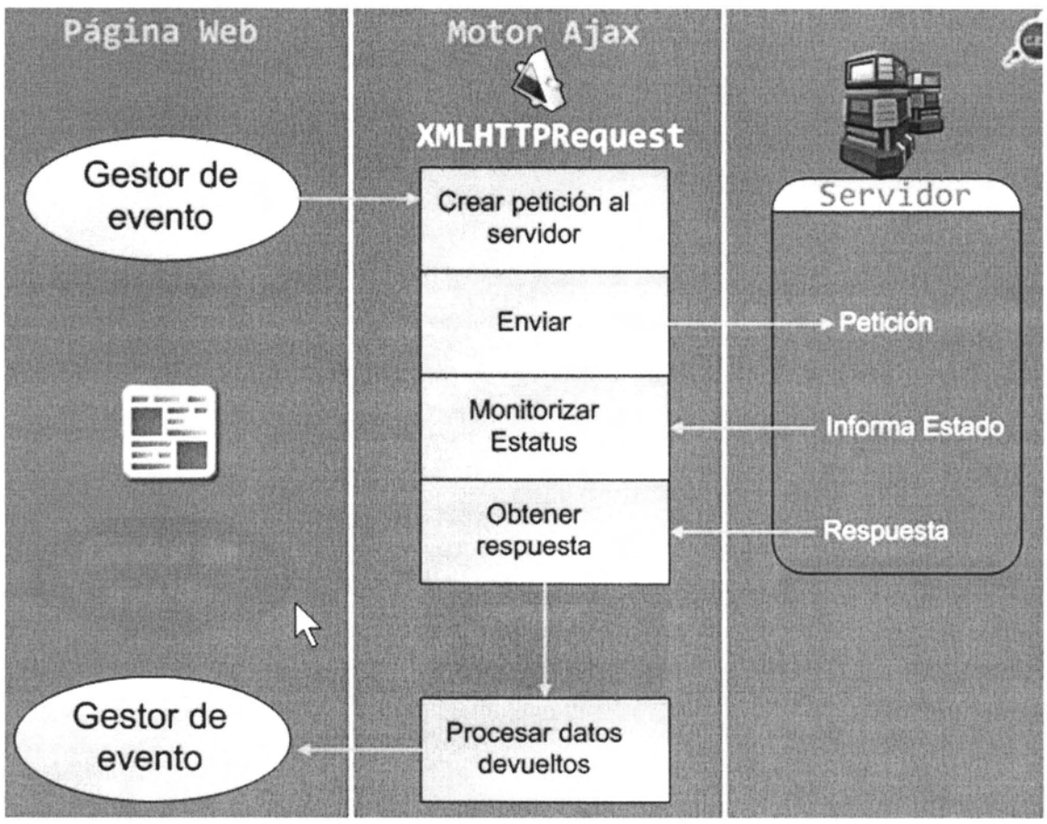


Figura 1.6.3: Procesamiento de una petición HTTP con AJAX

1.7 Motores de Plantillas

En la programación en general y en especial en la programación orientada para la web, es común separar los datos de la presentación, es decir, separar la lógica de negocio de la capa de presentación. Esto es lo que hace el conocido patrón Modelo Vista Controlador. En términos de la programación para la web esto consiste en separar el código del programador de lógica de negocio del diseñador de las páginas web o del programador de la página web. En este sentido las plantillas son una solución para lograr la separación del contenido de la forma en que se va a mostrar finalmente.

Los motores de plantillas son librerías que leen un fichero de texto que contiene la presentación ya preparada en HTML o en otro formato que se utilice para mostrar información, como XML, e inserta en el fichero información dinámica o datos para después ser presentado al usuario por la presentación (6). Los motores de plantillas son utilizados por los controladores para, una vez que hayan obtenido los datos devueltos por el modelo, se los deleguen a la plantilla para que formatee los datos en formato

correcto y dárselo a la presentación para que se los muestre al usuario. Los motores de plantillas por lo general suelen tener un lenguaje que es el que les permite generar código dinámico, tales como listas, arreglos, entre otros. También los motores de plantillas suelen tener una sintaxis para definir las plantillas, la cual difiere entre diferentes motores de plantillas.

Para comprender mejor que es un motor de plantilla se ilustrará el siguiente ejemplo.

```
<html>
  <body>
    Hola ${nombre}
  </body>
</html>
```

Figura 1.7.1: Formato de una plantilla

Cuando el controlador obtiene el dato nombre devuelto por el modelo, se lo dará al motor de plantilla para que procese el fichero que contiene la información a mostrar en HTML. El motor de plantilla recorrerá, paseará el fichero y sustituirá la etiqueta clave ${nombre} por el texto que le haya indicado el controlador, en este caso lo sustituirá por el valor que contiene el dato representado por nombre (6).

En el ejemplo se ilustro como los motores de plantillas pueden procesar ficheros y sustituir su contenido por elementos simples de datos. A parte de sustituir las etiquetas por datos simples, los motores de plantillas también pueden sustituir dichas etiquetas con los atributos simples que se encuentran en estructuras complejas como objetos, arreglos, listas o colecciones.

En la actualidad existen diversos motores de plantillas que son utilizados para realizar aplicaciones informáticas en varios lenguajes de programación.

Lenguaje Java	*Lenguaje PHP*
• Webmacro • Velocity • Tiles	• Smarty • TinyButStrong • NokTemplate

El uso de motores de plantillas provee muchas ventajas a las aplicaciones que los usan. Podemos citar por ejemplo:

- Separación de la estructura y forma del contenido.
- Separación de los roles de desarrollo.
- Reutilización de plantillas por varios controladores.
- Aumento de la eficiencia de los desarrolladores.
- Estandarización del diseño de la presentación.

1.8 Frameworks

Muchas de las personas que se han dedicado o que se dedican al desarrollo del software conocen o se han tropezado con el concepto de framework (cuya traducción aproximada al castellano sería el de "marco de trabajo"). Sin embargo, la definición de framework no es fácil de definir, a pesar de que cualquiera con experiencia en programación captará su significado de manera casual.

Entonces, podemos decir que un framework es "un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación" (7). También se puede definir como:"una estructura de soporte en el cual otro proyecto de software puede ser organizado y desarrollado".

Típicamente, un framework puede incluir soporte de bibliotecas, programas y un lenguaje interpretado, entre otros software, para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extienden o utilizan las aplicaciones del dominio. Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de un sistema funcional.

Es preciso destacar que los frameworks no estén necesariamente ligados a un lenguaje de programación concreto, aunque sea así en muchas ocasiones. En este sentido se encuentra Ruby on Rails, donde Rails es un framework de desarrollo y Ruby es un lenguaje de programación. Sin embargo, nada impide concebir un mismo framework para diferentes lenguajes de programación. Por ejemplo, existe un marco de desarrollo llamado "Biscuit" cuyo objetivo es convertirse en un "PHP on Rails". (7)

La utilización de marcos de trabajos proporciona diversas ventajas:

- El programador no necesita plantearse una estructura global de la aplicación.
- Facilita la colaboración.
- Reutilización de componentes.
- Estandarización del desarrollo.
- Agiliza el desarrollo.
- Reduce tiempo de desarrollo.

En la actualidad existen diversos frameworks que son utilizados para realizar aplicaciones informáticas en varios lenguajes de programación.

Lenguaje Java	Lenguaje PHP	Lenguaje JavaScript
• Struts	• ZendFramework	• Prototype
• Myfaces	• Symfony	• Dojo
• Tapestry	• Propel	• YUI
• Seam	• Codeigniter	• Ext
• Ibatis	• Prado	• JQuery
• Ehcache	• Doctrine	• Mootools

La utilización de un frameworks en la construcción de un sistema informático implica un cierto coste inicial durante el aprendizaje, aunque a largo plazo es probable que facilite el desarrollo como el mantenimiento de la solución informática.

1.9 Arquitectura del software

La Arquitectura del Software es unos de los temas actuales que ha dominado prácticamente la década de los 90, junto con otros temas de Ingeniería de Software. Es importante destacar que Arquitectura del Software no es Ingeniería de Software; sino que junto a los patrones de diseño, patrones de arquitectura y metodologías de desarrollo; forma parte de los temas de la Ingeniería del Software.

La Arquitectura del Software es unos de los temas de Ingeniería del Software que tiene un surgimiento tardío. La primera vez que se escucho sobre el tema fue en una conferencia de la OTAN que se

celebró en 1969, pero después del evento las referencias sobre Arquitectura del Software no tuvieron seguimiento. El término oficialmente comienza a escucharse gracias a un artículo publicado por la revista Dewayne Perry de Alexander Wolf en 1992 llamado "Foundations for the study of software architectura". En este artículo el autor imagina que la década del 90 va hacer la década de la Ingeniería del Software; también usa el término de Arquitectura en contraste con diseño, donde diferencian el significado de cada término. En el artículo el autor plasma una cantidad de ideas, entre ellas la de los estilos arquitectónicos. Estas ideas fueron tomadas de inmediato por la Universidad de Carnegie Mellon, en particular en el Instituto de Ingeniería del Software.

Existen varias definiciones sobre la arquitectura del software, todas ellas girando sobre 3 ideas distintas:

- La Arquitectura del Software como un proceso dentro del ciclo de vida.
- La Arquitectura del Software como la topología o distintas formas de articular los estilos y componentes dentro de una solución.
- La Arquitectura del Software como una disciplina.

La definición oficial de arquitectura del software fue elaborada por la IEEE y plantea que: "La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."

En un sentido amplio se podría estar de acuerdo en que la Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, programa o aplicación. Define estilo o combinación de estilos, se concentra en requerimientos no funcionales y tiene la responsabilidad de:

- Definir los módulos principales.
- Definir las responsabilidades que tendrá cada uno de estos módulos.
- Definir la interacción que existirá entre dichos módulos.
- Control y flujo de datos.
- Secuenciación de la información.
- Protocolos de interacción y comunicación.
- Ubicación en el hardware.

Hay que destacar y poner en claro que existe una gran variedad de estereotipos que han dominado el panorama en cuanto a lo que se supone que es lo que sería la arquitectura del software. En este sentido se refiere a que arquitectura del software no es:

- Una normativa madura.
- Igual en la academia y en la industria.
- Diseño de software con UM.
- Naturalmente vinculada a una metodología (XP).
- Naturalmente relacionada con modelado orientado a objeto.
- Hay vínculo "natural" entre requerimientos (casos de uso) y clases.

1.9.1 Estilos Arquitectónicos

Los estilos arquitectónicos se identifican como un conjunto de reglas de diseño que identifica las clases de componentes y conectores que se pueden utilizar para componer un sistema o subsistema, junto con las restricciones locales o globales de la forma en que la composición se lleva a cabo (8). Carnegie Mellon define el estilo como una entidad consistente en cuatro elementos: un vocabulario de elementos de diseño, componentes y conectores tales como tuberías, filtros, clientes, servidores, parsers, bases de datos, etcétera; reglas de diseño o restricciones que determinan las composiciones permitidas de esos elementos; una interpretación semántica que proporciona significados precisos a las composiciones; análisis susceptibles de prácticas sobre los sistemas construidos en un estilo, por ejemplo, análisis de disponibilidad para estilos basados en procesamiento en tiempo real, o detección de abrazos mortales para modelos cliente-servidor.

Los estilos de arquitectura expresan una organización estructural para un sistema software, definen los patrones posibles a utilizar en la solución y proveen un conjunto de subsistemas predefinidos e incluyen reglas y lineamientos para conectarlos.

1.9.1.1 Estilo arquitectónico Modelo – Vista – Controlador (MVC)

Una de los escenarios más comunes de las aplicaciones es tomar datos de un almacenamiento y mostrárselo al usuario. Una vez mostrado estos datos al usuario, el usuario introduce modificaciones a los datos y los manda a almacenar. Dado que el flujo de información ocurre entre el almacenamiento y la interfaz de usuario, lo más común de hacer es unir ambas partes para ganar en rendimiento y reducir la cantidad código. Pero debido a que la interfaz de usuario cambia mucho y aparte que las aplicaciones suelen incorporar reglas de negocio que vas más allá de la prestación de datos, esta idea es poco adoptada.

El estilo arquitectónico conocido como Modelo–Vista–Controlador resuelve este problema, ya que separa el modelo del dominio, la presentación y las acciones del usuario en tres clases diferentes.

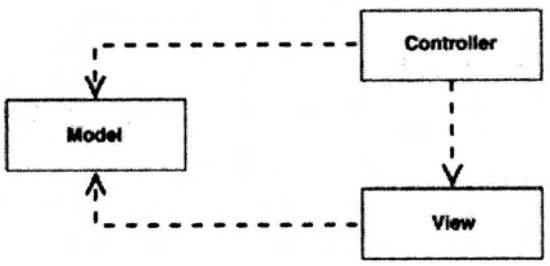


Figura 1.9.1.1.1: Elementos del patrón MVC

- Modelo: Contiene los datos del dominio de la aplicación y las reglas de negocio de la aplicación.
- Vista: Es la responsable de mostrarle la información al usuario.
- Controlador: Responde a las acciones del usuario, informando al modelo y a la vista que cambien según el cambio realizado.

Debido a esta separación el estilo MVC tiene numerosas ventajas y desventajas. Entre sus ventajas están: soporte de múltiples vistas y adaptación al cambio. Algunas de sus desventajas son: complejidad y costo de actualizaciones frecuentes. El estilo MVC es uno de los estilos arquitectónicos que más se utiliza para darles solución a las mayorías de las aplicaciones web actuales.

1.9.1.2 Estilo arquitectónico en capas

El estilo en capas es uno de los estilos que aparece con mayor frecuencia en las soluciones informáticas. Se define al estilo en capas como una organización jerárquica tal que cada capa proporciona servicios a la capa superior inmediata y se sirve de las prestaciones que le brinda la capa inferior inmediata []. Descompone la aplicación en un conjunto de capas independientes y ordenadas jerárquicamente.

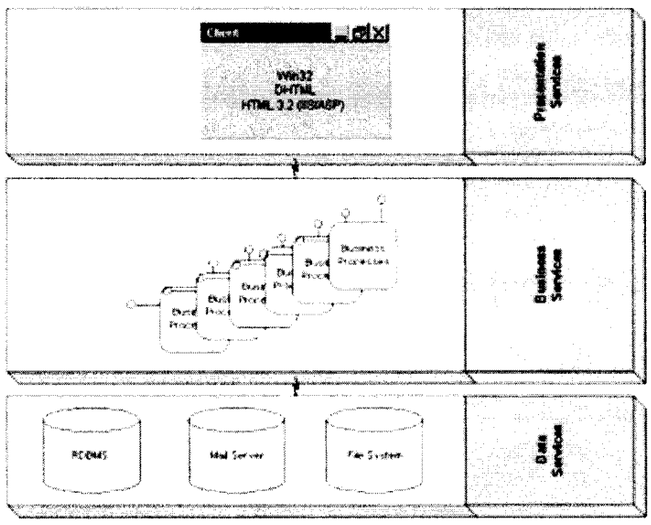


Figura 1.9.1.2.1: Elementos del patrón 3 capas, subconjunto del estilo N capas

El estilo en capas posee varias ventajas y desventajas. Algunas de sus ventajas son:

- Reutilización de un mismo nivel en varias aplicaciones.
- El cambio de un nivel no afecta al resto.
- Permite optimización y refinamiento en las capas.
- Permite la estandarización.

Y como desventajas:

- Trabajo innecesario de paso de argumentos.
- Si el número de niveles es excesivo puede ser muy ineficiente.
- Si hay pocos niveles tenemos el diseño poco organizado.
- Un mal diseño o cambio importante de funcionalidad pueden forzar cambios que se trasmiten en cascada de un nivel a otro.

Capítulo 2: Tecnologías utilizadas en el nuevo sistema SIIPOL

2.1 Introducción

A la hora de desarrollar un proyecto hay que ver las diferentes plataformas de desarrollo existentes para la construcción de un software. En el mundo actual existen dos plataformas que llevan el liderazgo en el aspecto de producir aplicaciones empresariales: Java y .Net. En esta directriz no se tuvo que profundizar mucho a la hora de elegir la plataforma de desarrollo del software. En nuestro caso elegimos para desarrollar nuestro software la plataforma Java, que fue la elegida por el cliente de antemano, o sea, desde el inicio de las negociaciones el cliente pidió que se realizara el software utilizando esta plataforma.

El proyecto CICPC debe construir un sistema orientado a la web utilizando la plataforma Java, por lo que hay que concentrar los esfuerzos a investigar las diferentes tecnologías utilizadas en dicha plataforma para la construcción de un sistema orientada a la web.

A la hora de desarrollar una aplicación web existen dos grandes clasificaciones de tecnologías a utilizar, que se agrupan en: tecnologías del lado del cliente y tecnologías del lado del servidor. Las aplicaciones web realizadas utilizando la plataforma Java no escapan de estas dos grandes clasificaciones.

En este capítulo se brinda una aproximación general y detallada de las tecnologías que se encuentran dentro de las dos clasificaciones antes mencionadas y de los framework utilizados para la construcción del nuevo sistema SIIPOL.

2.2 Tecnologías que se ejecutan en el cliente

En la construcción del nuevo sistema SIIPOL se utilizaron diversas tecnologías que se ejecutan del lado del cliente. Debido a que la naturaleza de dicha aplicación se encuentra enmarcada dentro de los sistemas orientados a la web, dichas tecnologías se deberían ejecutar en un cliente elaborado para estos tipos de aplicaciones, en este caso un navegador web.

Las tecnologías usadas para la construcción de dicho sistema que se ejecutan en el cliente son:

- JavaScript
- CSS
- HTML
- Prototype

- Applet

2.2.1 JavaScript

JavaScript es un lenguaje interpretado que no requiere de compilación. Es utilizado generalmente en las páginas Web y es interpretado por un programa, normalmente un navegador Web. Esta tecnología es utilizada para ejecutar instrucciones en una página web, como validar datos de entradas por el usuario, personalizar las páginas web, entre otras cosas.

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como aparición y desaparición de texto, animaciones, acciones que se activan al pulsar botones u otros elementos y ventanas con mensajes de aviso al usuario. (9)

Para que el lenguaje JavaScript pueda ser interpretado por el navegador este debe estar embebido en una etiqueta <script> dentro de cualquier de los formatos utilizados para mostrar información en las páginas web, generalmente dentro de un documento HTML.

Desde la aparición del lenguaje JavaScript ha sido utilizado de forma masiva por casi todos los sitios de internet. Pero debido a la aparición de otras tecnologías, como FLASH por ejemplo, uso ha ido disminuyendo gradualmente, ya que con dichas tecnologías se podían realizar acciones que con JavaScript era imposible de ejecutar. Sin embargo con la aparición del objeto XMLHTTPRequest y la puesta en práctica de la técnica de AJAX, JavaScript vuelva a ser usado por los sitios en internet.

Uno de las principales ventajas del uso de JavaScript es que permite elaborar aplicaciones Web que simulen características de aplicaciones escritorios, ya que logra establecer una mejor interacción entre las páginas web y los usuarios. Otra ventaja que aporta JavaScript es que se pude utilizar en la elaboración de páginas web para ser vistas desde cualquier plataforma (Windows, Linux), debido a que es soportado por una gran variedad de navegadores y exploradores Web. JavaScript también le permite a los desarrolladores de interfaces de usuario tener mejor control de las páginas Web. Esto se logra gracias a que a través de JavaScript, los desarrolladores de interfaces de usuario pueden acceder a las diferentes partes y elementos de una página Web (botones, textos, imágenes, videos, etiquetas).

Por desgracia el lenguaje JavaScript no resuelve todas las expectativas de los desarrolladores como herramienta. En ocasiones los desarrolladores deben realizar diferentes implementaciones de código JavaScript para sus aplicaciones, debido a que no todos los navegadores interpretan de la misma manera el código. Unos de los inconvenientes que tiene este lenguaje es que tiene que estar activado

en los navegadores para que su uso sea posible por estos. Por otro lado no le proporciona al programador control total de la página web. El control total no es posible ya que no es un lenguaje tan potente como para brindar funcionalidades que le sean útiles a los desarrolladores. Es una tecnología que pertenece a los estándares Web de la W3C.

2.2.2 CSS

Las Hojas de Estilo en Cascada o CSS es un lenguaje de hojas de estilos creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML, XHTML o XML (10). Las Hojas de Estilos en Cascada son un mecanismo que define cómo se va a mostrar un documento electrónico en el navegador web o incluso cómo va a ser pronunciada la información de un documento a través de un dispositivo de lectura.

En la actualidad CSS es la mejor alternativa para separar en un documento electrónico el contenido del documento de la forma en que se va a presentar en un navegador web, por lo que se ha vuelto imprescindible para crear páginas web complejas en el presente. CSS se utiliza para darle estilos a los documentos electrónicos como HTML y XHTML, delimitando el contenido de la presentación. En general, los estilos definen la forma de mostrar los elementos HTML y XML (11).

Debido a que los CSS son un lenguaje interpretado se necesita de un programa que interpreta las instrucciones, generalmente es un navegador web. El navegador, para poder interpretar las CSS, debe estar embebido dentro de cualquiera de los formatos existentes para confeccionar documentos electrónicos para la web. Existen varias formas para poder incorporar los CSS a los documentos electrónicos. Una de ellas es usando hojas de estilos externas que están vinculadas al documento a través de etiquetas <link> situadas dentro del elemento <head>. Otra manera es embeber las instrucciones CSS dentro del elemento <style> en la cabecera o el cuerpo del documento. La primera forma es la que más se utiliza y la más recomendada.

Las Hojas de Estilo en Cascada funcionan a bases de reglas o declaraciones de los estilos de uno a más elementos. Estas están compuestas por una o más reglas aplicadas a los documentos electrónicos como HTML y XML. Las reglas de las CSS constan de dos partes: el selector y la declaración. La declaración está compuesta por una propiedad y un valor que se le asigna a la propiedad.

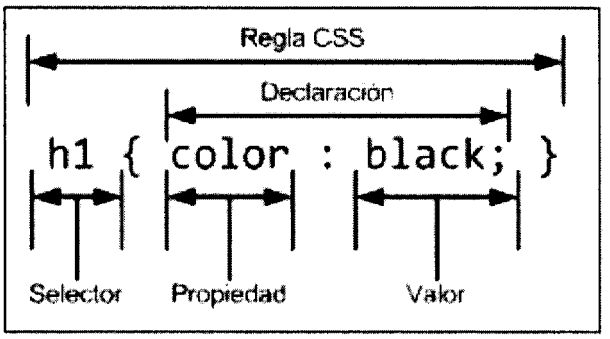


Figura 2.2.2.1: Componentes de un estilo CSS

Unos de los principales beneficios que aporta el uso de las Hojas de Estilos en Cascada es permitir que las páginas web se muestren con elegancia en el navegador. Esta elegancia se debe a que las CSS permiten la separación de los contenidos de la presentación, mejorando la accesibilidad de los documentos. Otra ventaja que aporta la utilización de CSS es que permite que las páginas web se presenten en buen estado en los navegadores, permitiendo que se elaboren documentos electrónicos bien definidos. Permiten a los diseñadores reducir los costos de mantenimiento de las páginas web al reducir la complejidad los documentos. Por otro lado, gracias a las CSS los documentos electrónicos se puedan visualizar en diferentes dispositivos (navegadores, celulares).Los CSS son una tecnología que pertenece a los estándares Web de la W3C.

2.2.3 HTML

HTML (lenguaje de marcado de hipertexto) es un lenguaje de marcado diseñado para estructurar textos y presentarlos en forma de hipertextos, que es el formato estándar de las páginas web. HTML es un lenguaje interpretado, por lo que necesita de un programa que interprete las etiquetas que están dentro del cuerpo del documento en formato HTML y los muestre en navegador.

Se dice que HTML es un lenguaje script porque su contenido está formado por varias etiquetas o guiones que se encuentran embebidas en el cuerpo de un documento, que son los que le dan el formato. Básicamente un documento HTML está formado por una etiqueta <html></html> que es la que define el inicio del documento y es la que le indica al navegador que lo que viene a continuación debe ser interpretado como código HTML. Dentro de esta etiqueta podemos encontrar varias etiquetas, como las etiquetas <head>,<body>, entre otras. La etiqueta <body> es la que define el contenido principal o cuerpo del documento HTML, o lo que es lo mismo, la parte del documento HTML que se muestra en el navegador.

La principal ventaja que tiene HTML la gran variedad de navegadores y exploradores que lo soportan. Debido a esto se ha convertido en el formato más usado para la transmisión de documentos

electrónicos a través de Internet. Otra ventaja que aporta es su compatibilidad con los navegadores más antiguos, donde estos navegadores aún pueden interpretar documentos electrónicos en HTML. Otro beneficio que aporta HTML es a la hora de estudio: disminuye el costo de aprendizaje de las personas que empiezan a aprender a leer y a escribir HTML, siendo un lenguaje muy fácil de utilizar y comprender. Pero también tiene sus inconvenientes. No se pude definir etiquetas personalizadas en los documentos HTML. Existe un número definido de etiquetas que pueden estar embebidas dentro de un documento HTML. Otro inconveniente que tiene es lo difícil que resulta darse cuenta de los errores de contenido de un documento HTML, ya que la depuración a la hora de codificar es complicada. HTML es una tecnología que pertenece a los estándares Web de la W3C.

2.2.4 Prototype

Prototype es un framework de JavaScript construido para elaborar de manera fácil aplicaciones Web dinámicas (12). Está orientado el desarrollo sencillo y dinámico de aplicaciones basadas en la web. Prototype es una herramienta que brinda muchas funcionalidades a las aplicaciones web que lo utilizan. Entre las funcionalidades que brinda está la de habilitarle a las aplicaciones la capacidad del uso de AJAX de manera fácil. También posibilita a los desarrolladores el uso de otras funcionalidades definidas por la librería de manera sencilla, tales como funciones para buscar en colecciones, iteradores, funciones que devuelven elementos HTML, entre otros.

Lo más notable de Prototype es la gran variedad de funcionalidades que aporta al desarrollo, logrando disminuir los tiempos de elaboración de las aplicaciones. Disminuye costo de esfuerzo a los desarrolladores cuando utilizan JavaScript en la búsqueda de soluciones complejas.

Debido a las diversas prestaciones y posibilidades que aporta su uso, se ha convertido en unos de los framework de JavaScript más utilizado en la solución de un número considerable de aplicación web. También ha sido adoptado por una gran cantidad de compañías en la confección de sus productos. Prototype es un framework de código abierto (open source).

Compañías que utilizan Prototype		
• APPLE • NASA • GUCCI • ESPN • SONY • FIFA • CNN	• AMAZON • SPICEWORKS • REDDIT • DREAMHOST • MICROSOFT • GRAVATAR • TIVO	• IIPROPERTY • Hi5 • MAPQUEST • PRADA • SCRIBD • SAYNOW • MAGNOLIA

2.2.5 Applet

Un applet es un programa Java que se inserta en una página Web (13). Visto de otra manera, es un pequeño programa hecho en Java, que se encuentra almacenado en el servidor y se ejecuta en la máquina del usuario que lo está utilizando. Debido a que los applet se ejecutan en el cliente, se necesita tener instalado la maquina virtual de Java (JVM) para que él pueda ser interpretado y ejecutado.

La principal ventaja de uso de los Applet en las aplicaciones web es que los se pueden ejecutar en cualquier plataforma, debido a su independencia de la máquina y del sistema operativo. Esto posibilita a los desarrolladores no hacer ningún cambio en el código de los applet para que se puedan ejecutar en diferentes plataformas. Otra ventaja es que evitan los problemas de actualización y distribución. Si los programadores necesitan hacer una nueva versión del applet, lo depositan en el servidor web y automáticamente todos los clientes que acceden al servidor a partir de ese momento descargan la nueva versión del applet sin que estos se den cuenta. También a través de los applet los desarrolladores pueden conocer características de las maquinas clientes que lo están utilizando, tales como dirección IP, características de hardware, entre otras cosas.b

Los applet a veces son llamados clientes pesados, ya que consumen mucho tiempo de descargas. Por eso cuando hay que descargar applet que son muy voluminosos, se recurre a empaquetarlos en archivos JAR y comprimirlos para que sea más rápido la descarga. El applet forma parte de las especificaciones de Java.

2.3 Tecnologías que se ejecutan en el servidor

En la construcción del nuevo sistema SIIPOL para CICPC se utilizaron varias tecnologías que se ejecutan del lado del servidor. Debido a que la naturaleza de dicha aplicación se encuentra enmarcada dentro de los sistemas orientados a la web, estas tecnologías tenían que ejecutarse en un contenedor web.

Las tecnologías usadas para la construcción del nuevo sistema SIIPOL que se ejecutan en el servidor son:

- Servlets
- JavaServer Pages
- JavaBean
- Hibernate
- Spring
- Acegi
- MyFaces

2.3.1 Servlets

El servlet es una de las tecnologías utilizada en la plataforma Java para procesar las peticiones enviadas por el cliente en el servidor. Tienen la tarea de ser intermediario entre el cliente y el servidor en las aplicaciones web que utilizan la plataforma Java. Técnicamente un servlet es un programa que se ejecuta en un contenedor Web (Apache Tomcat) (14). Aportan la misma funcionalidad a la programación en el lado del servidor que los tradicionales CGI, con la única diferencia que el servlet se crea una sola vez en el servidor y procesa las peticiones de los clientes con la misma instancia en hilos diferentes. Debido a estas diferencias los servlets surgieron para sustituir a los tradicionales CGI en la plataforma Java. No constituyen la única tecnología disponible para procesar las peticiones HTTPRequest en un servidor Web, sino que existen otras tecnologías como ASP.NET y PHP que se utilizan para el mismo propósito. Es importante destacar que los servlets forman parte de las especificaciones de Java, por lo que técnicamente se pude implementar en cualquier plataforma.

Se han convertido en la piedra angular del desarrollo de aplicaciones Web en Java debido a que aporta diversas ventajas a las aplicaciones. Uno de las principales ventajas de uso es que proporcionan a las aplicaciones la posibilidad de que puedan ser ejecutadas en cualquier plataforma, brindando gran portabilidad. Las aplicaciones que utilizan los servlets ganan rendimiento, debido a que cada petición es procesada por un único proceso en el contenedor de servlets. Proporcionan a los

desarrolladores un rápido desarrollo ya que proporcionan acceso a las librerías de Java. También su uso aporta robustez a las aplicaciones debido a que son gestionados por la máquina virtual de Java.

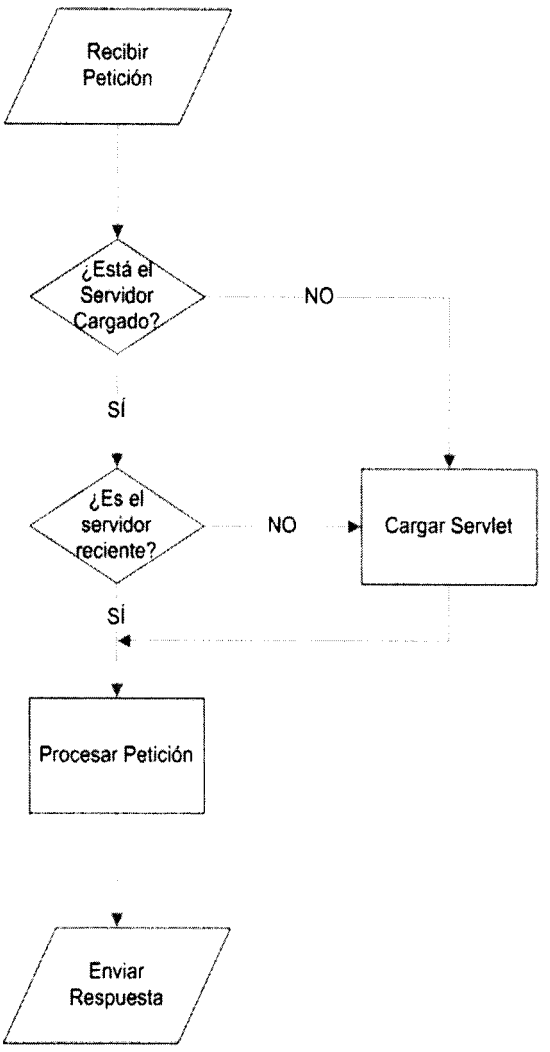


Figura 2.3.1.1: Funcionamiento de un servlet

La Imagen anterior muestra gráficamente como es el funcionamiento de un servlet en un contenedor de servlet. Cuando la petición HTTP es recibida, se verifica si el servidor este cargado. En caso de que el servidor no esté cargado se inicia la carga del servlet. En caso contrario se verifica si el servidor es reciente. Si esto resulta afirmativo se vuelve a cargar el servlet. Después de estos pasos, una vez cargado el servlet, este está listo para procesar la petición y enviar una respuesta al navegador.

Arquitectónicamente para que los servlets puedan ejecutarse se necesita de un Servlet Container. Un Servlet Container no es más que un servidor capaz de ejecutar servlets.

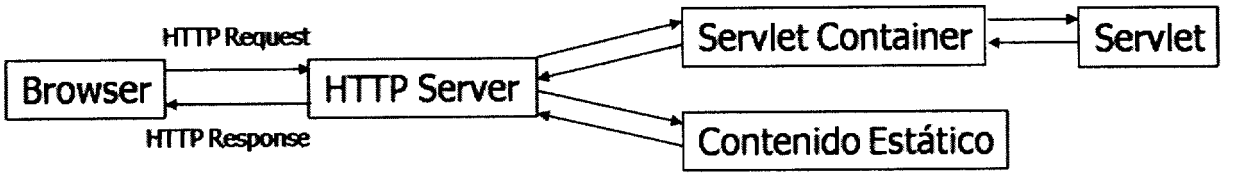


Figura 2.3.1.2: arquitectura de un servlet

2.3.2 JavaServer Pages

JavaServer Pages (JSP), su traducción al español sería páginas servidoras de Java y forma parte de las especificaciones de Java. JSP es una tecnología orientada a crear aplicaciones web dinámicas que se ejecutan en un servidor usando el lenguaje Java (15). En ese sentido las JavaServer Pages son similares a otros lenguajes o tecnologías tales como PHP, ASP.NET, CGI; programas que generan páginas web en un servidor. Las JSP se ejecutan en la máquina virtual de Java, lo cual permite que esta tecnología se pueda se pueda usar en cualquier ordenador, siempre que exista una máquina virtual Java en el mismo.

Los JavaServer Pages son en esencia en Servlet. Cuando una página JSP se compila en un programa por primera vez se invoca y se crea una clase que empieza a ejecutarse en el servidor como si fuera un servlet. Las paginas JSP están compuestas de código HTML/XML mezclado con etiquetas especiales para programar script de servidor en sintaxis Java (16); aunque esta última posibilidad nos es muy recomendada. El enfoque de programación es la principal diferencia que difiere a los JSP de los servlets, debido a que un JSP es una página Web con etiquetas HTML y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web como respuesta.

La tecnología JavaServer Page aporta diversos beneficios a las aplicaciones web que lo utilizan. Permite crear páginas dinámicas, cuyo contenido varía en cada ejecución. También con el uso de JSP podemos responder a peticiones o información que los usuarios envían. Otros de los beneficios que brinda JavaServer Pages es que a través de ellas se puede acceder a diferentes bases de datos. Programar con instrucciones JSP es más rápido, porque no necesitamos compilar de nuevo los archivos JavaServer Pages.

Como tecnología, JavaServer Pages tiene sus inconveniencias. Para ejecutar las páginas JSP necesitamos de un servidor capaz de ejecutarlas, es decir, el servidor web debe ser un contenedor web. Otro inconveniente es que en el contenedor web donde se vaya a ejecutar las paginas JSP debe estar instalado la maquina virtual de Java para que dicho contenedor pueda interpretar las instrucciones JavaServer Pages. Pero la principal desventaja que tiene el uso de JSP en las

aplicaciones web es que no se pude tener un control total del formato del contenido de las páginas JSP porque el contenido HTML que genera las instrucciones JSP puede estar con errores de sintaxis en las etiquetas HTML.

La siguiente imagen muestra gráficamente los pasos a realizar por el servidor para un procesamiento JSP.

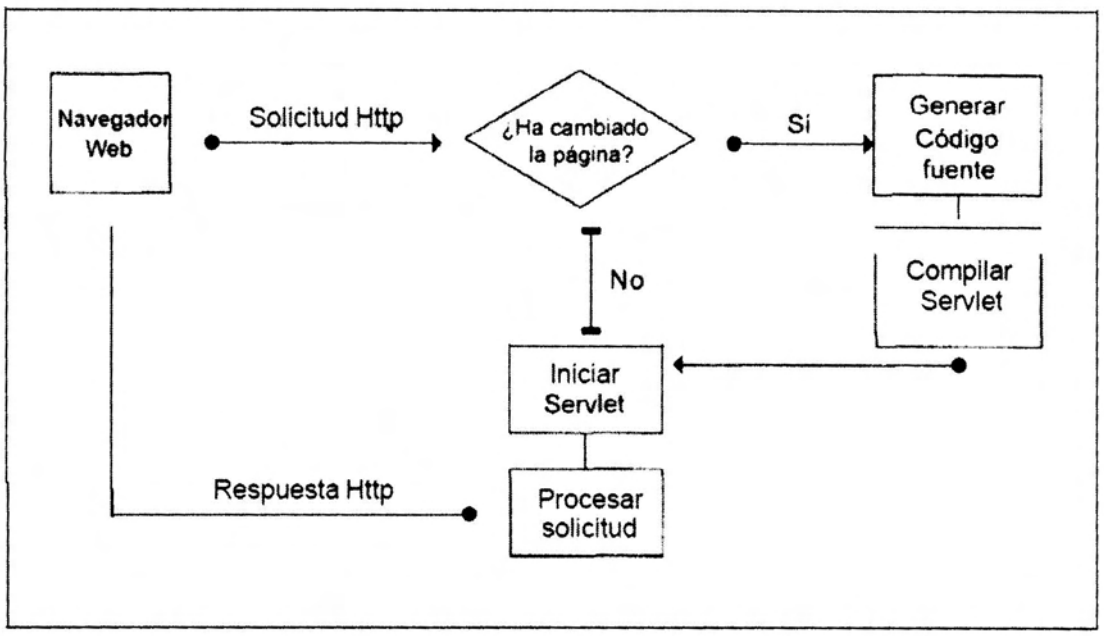


Figura 2.3.2.1: Pasos del procesamiento JSP

La primera vez que se solicita una página JSP, el motor de servlet lee el archivo y genera código fuente para crear el servlet correspondiente a la página JSP. El código se compila y se inicia el servlet. Una vez iniciado, administra las solicitudes del mismo modo que cualquier otro servlet. La única diferencia es que cada vez que se hace una solicitud de un archivo JSP, el registro del archivo se compara con otro registro servlet. En caso del que el archivo servlet sea más nuevo, entonces tiene que volver a ser compilado en un servlet y después volver a inicializarse.

2.3.3 JavaBean

Un JavaBean o bean es un componente hecho en un software que se puede reutilizar y que puede ser manipulado visualmente por una herramienta de programación en lenguaje Java (17). Un bean es una clase de Java. Las clases de Java, para poderse definir como bean o componente, deben obedecer a determinas reglas. Deben poseer un constructor por defecto, o sea, un constructor sin argumentos. Estas clases también tienen que cumplir con el término de introspección; los IDE de desarrollo en Java

deban reconocer ciertas pautas de diseño, nombre de métodos o funciones miembros y definiciones de clases que permitan a la herramienta de programación inspeccionar el bean para así conocer sus propiedades y conducta.

Debido a que los bean son simple clases Java se pueden crear bean individuales para que puedan realizar diversas funcionalidades, las cuales pueden ser simples o complejas. Aunque los bean difieren en cuanto a funcionalidades, poseen características comunes. Dichas características son: Introspección (está característica le permite analizar a la herramienta de programación o IDE como trabaja el bean), Customization (le posibilita al programador poder alterar la apariencia y la conducta del bean), Events (le informa al IDE de los sucesos que puede generar en respuesta a las acciones del usuario o del sistema y también los sucesos que puede manejar), Properties (permite cambiar los valores de las propiedades del bean para personalizarlo (customization)), Persistence (posibilita guardar el estado de los beans que han sido personalizados por el programador, cambiando los valores de sus propiedades) (17).

Los JavaBean aportan varios beneficios al desarrollo de software en la plataforma Java, permitiendo a los desarrolladores elaborar componentes reutilizables en la elaboración de aplicaciones. Esto posibilita tener clases complejas que realicen tareas y funcionalidades diversas. Una de la ventaja más importante que aporta los JavaBean es que le permite a los desarrolladores modificar el estado de los bean en tiempo de ejecución y persistir los cambios de estos en cualquier medio de persistencia como en memoria o un fichero.

Los JavaBean juegan un importante papel en la arquitectura MVC. Suelen almacenar el modelo o los datos de una aplicación. El controlador es responsable de actualizar el modelo y la página únicamente necesita leer los datos de modelo para presentarlos en el navegador. Esto se pude lograr mediantes los JavaBean y las etiquetas que JSP usa para acceder a los JavaBean. JavaBean forman parte de las especificaciones de Java.

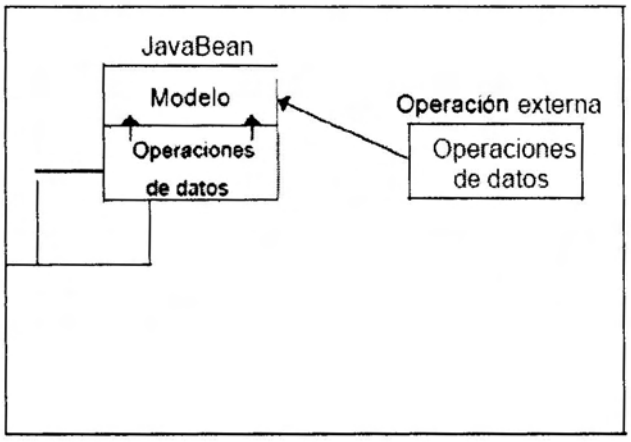


Figura 2.3.3.1: JavaBean y JSP

2.3.4 Hibernate

Hibernate es un entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y al mismo tiempo la consulta de estas bases de datos para obtener objetos (18). El marco de trabajo Hibernate trata de relacionar las entidades del modelo de dominio con las tablas de las bases de datos relacionales para así permitir un acceso al gestor de base de datos de modo objetual.

En el desarrollo de software utilizando el paradigma de la programación orientada a objeto es muy difícil de implementar o llevar a cabo la persistencia de objetos en un soporte relacional (base de datos relacional). El problema es que existe un divorcio total entre la estructura relacional y la objetual, ya que no se puede persistir objetos en tablas de las base de datos relacionales. Esto provoca que los desarrolladores elaboren con frecuencia soluciones que hacen perder de vista el mundo objetual y hacen que la aplicación valla cediendo las ventajas que da el paradigma orientado a objeto.

Como solución a estos problemas que eran comunes en las aplicaciones de Java que utilizan el paradigma orientado a objeto surgió Hibernate. Este relaciona las entidades persistentes con las tablas de las base de datos, permitiéndole a los desarrolladores seguir utilizando las ventajas del paradigma de programación orientada a objeto. De este modo facilita a los programadores persistir entidades de negocio y obtener consultas objetuales. Hibernate logra esa relación debido a que implementa el patrón de diseño Object Relational Mapping (ORM), que es el que plantea la solución al problema de la persistencia de objetos en soportes relacionales.

El uso de Hibernate aporta diversas ventajas a las aplicaciones que lo usan. Uno de los principales beneficios que aporta el uso de Hibernate es que permite mapear un modelo de clases a un modelo relacional sin imponer ningún tipo de restricción en ambos diseños. Otro beneficio que aporta el uso de

Hibernate es que posibilita la generación de clases Java persistentes a partir de las tablas de la base de datos y viceversa. Con el uso de Hibernate se pude obtener un producto con un pequeño intervalo de tiempo, ya que agiliza el proceso de desarrollo debido a las diferentes funcionalidades reutilizables que brinda. También permite la especialización de roles de desarrollo en la capa de acceso a datos independizando la BBDD de la aplicación.

Hay que destacar que Hibernate, debido a las grandes a prestación que brinda, se ha convertido en uno de los marco de trabajo más utilizado en la solución de aplicaciones empresariales. Cuenta con una amplia documentación, tales como publicaciones de libros propietarios y bibliografías disponibles gratuitas en internet. Es un framework de código abierto (open source).

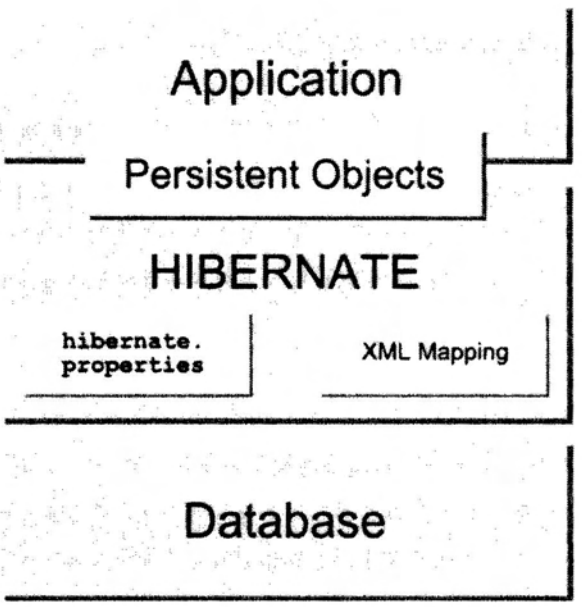


Figura 2.3.4.1: Primer nivel de la arquitectura de Hibernate (19)

La figura anterior muestra como Hibernate usa la base de datos y la configuración de los datos para proveer los servicios de persistencias y los objetos persistentes a la aplicación.

Nota: Los ORM son los encargados de generar el SQL correspondiente de la relación que se establezca entre las entidades persistentes y las tablas de la base de datos.

2.3.5 Spring

Spring es un conjunto de librerías, de las cuales podemos elegir aquellas que nos permitan elaborar la solución de un sistema informático. Spring posibilita configurar las clases en archivos XML y definir en

él las dependencias de las clases. Otra alternativa que usa Spring para configurar las dependencias de las clases sin la necesidad de utilizar archivos XML son las notaciones. Esto significa que las clases definen en ellas mismas sus dependencias mediante anotaciones y Spring se encarga de crear a partir de estas anotaciones los ficheros de configuración de dependencias de dichas clases. Esta forma de configuración es la más recomendada, ya que disminuye archivos físicos hechos por los desarrolladores en las aplicaciones y es una forma de evitar los errores de configuración que cometen los programadores a la hora de editar los archivos XML de configuración de Spring. De esta manera las aplicaciones se tornan modulares y a la vez no adquiere dependencias con Spring.

Spring es un framework de desarrollo que posee y brinda una gran variedad de posibilidades y funcionalidades. Entre sus posibilidades más potentes está su contenedor de Inversión de Control, llamado en ocasiones Inversión de Control o Inyección de dependencias. Esta es una técnica alternativa a las búsquedas clásicas de recursos vías JNI.

Otras de las posibilidades que brinda Spring es la posibilidad de la utilización de aspectos en diferentes partes de las aplicaciones. Se integra con otros frameworks de desarrollo como Hibernate, Acegi, Ibatis, JavaServer Faces y a la vez le permite a estos comunicarse sin la necesidad de tener dependencias entre ellos. Spring es uno de los proyectos actuales mas sorpréndete en la plataforma Java, en el grado que ayuda a los diferentes componentes que conforman una aplicación para que trabajen entre sí, sin que se establezca dependencia entre los componentes, componentes aplicación y consigo mismo. Esta es una de las características más significativas del framework Spring, por lo que sería posible retirar el framework sin cambiar prácticamente ninguna línea de código. Si se retirara Spring de una aplicación, lo único necesario sería añadir algún framework que se encargara de lo que hacía Spring o añadir código para sincronizar las dependencias de los componentes.

El framework Spring está compuesto por una gran cantidad de módulos que son los que conforman junto con el core la estructura del framework Spring. Es un framework de código abierto (open source).

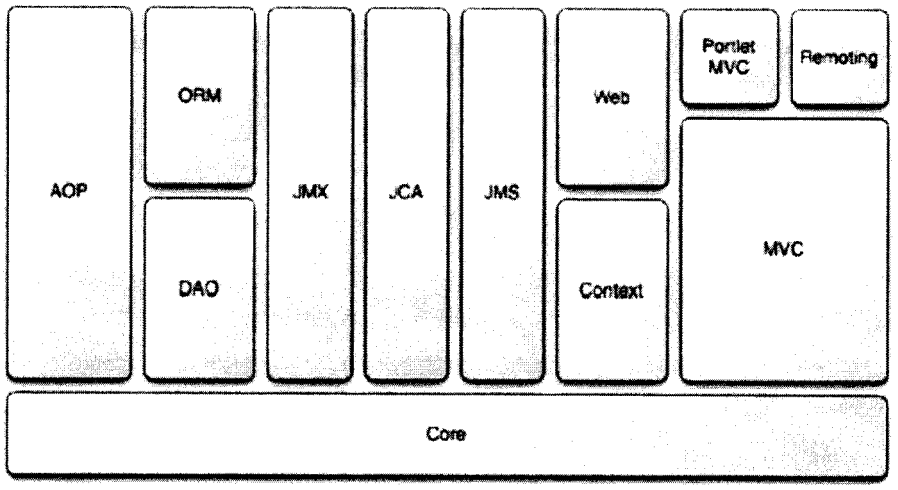


Figura 2.3.5.1: Estructura del framework Spring (20)

2.3.6 Acegi

Acegi es un marco de trabajo que permite manejar la autenticación y autorización de aplicaciones web en Java. La autenticación es posible realizarla desde diversos proveedores: ldap, gestores de base de datos, sistema de único logueo e integrado del contenedor de aplicación. La autorización se aplica en diversos niveles de la aplicación como: acceso a URL, llamadas a métodos de negocios y acceso a objetos del dominio. El acceso a URL se controla mediantes filtros web, los métodos de negocio se aseguran a través de aspectos de Spring y los objetos del dominio utilizando listas de control de acceso. Acegi es un gestor de seguridad que está diseñado fundamentalmente para ser usado con Spring (21). Otra de las características de Acegi es que permite la comunicación con otros frameworks de desarrollo (Spring, Hibernate, Ehcache, Jcaptcha).

Uno de las principales ventajas del uso de Acegi es que permite la declaración de la seguridad de una forma declarativa y no intrusiva: declarativa porque permite realizar la configuración de la seguridad en archivos XML y no intrusiva debido a que no molesta el funcionamiento de los demás componentes. Otra ventaja que proporciona Acegi es que permite manejar la seguridad en varias capas, cubriendo la capa de presentación y la de negocio. Acegi es un framework de código abierto (open source).

2.3.7 MyFces

MyFaces es un proyecto de la fundación Apache que ofrece una implementación en código abierto de JavaServer Faces, así como un amplio conjunto de componentes adicionales. Entre ellos se dispone de un menú, arboles, pestañas, componentes para gestionar el estado de los diálogos, entre otros. Aparte de estos componentes MyFaces también brinda los componentes estándares de la

especificación de JSF para tener compatibilidad con la especificación. Esto lo tiene que cumplir ya que cualquier proyecto que elabore un producto, que implemente una especificación de Java, debe cumplir en su totalidad lo que cumple la especificación. De esta manera se logra que si se cambia la librería de MyFaces por otra que cumpliera con la especificación de JSF no se tenga que cambiar muchos códigos y configuraciones en la aplicación.

Lo más sobresaliente de esta librería es la integración con otras librerías hechas para trabajar con las especificaciones de JSF: Myfaces Tomahawk, MyFaces Tinidad y MyFaces. Utilizar Myfaces en el desarrollo de aplicaciones Web en Java trae muchos beneficios, ya que permite acelerar el proceso de desarrollo de esas aplicaciones, debido a la gran variedad de componentes reutilizables que posee. También MyFaces permite que las aplicaciones se desarrollen en un menor intervalo de tiempo, que sean robustas y escalables.

En su inicio el proyecto MyFaces contaba de una gran aceptación por la gran cantidad de componentes reutilizables que aportaba al desarrollo. Pero debido a la mala programación de la librería a perdido liderazgo en el desarrollo de aplicaciones empresariales utilizando JSF. Esto se debe a que diversos componentes y configuraciones de MyFaces no están bien programados o no realizan las funcionalidades por la que fueron creadas. Por otro lado están las débiles relaciones con las diferentes librerías que funcionarían con MyFaces. A esto se une la falta de documentación de la librería, sea en formato duro (libros) o publicaciones digitales gratuitas en Internet. Lo que dicho de otra forma, las diversas posibilidades que MyFaces brindaría no se cumplen en todo su totalidad.

Capítulo 3: Propuesta de Solución

3.1 Introducción

Para cumplir los objetivos propuestos se optó por utilizar una arquitectura en capas, para separar las distintas capas posibles que pudiera tener el nuevo sistema SIIPOL. Para lograr las cosas que plantea la arquitectura en capas se eligió el patrón Modelo–Vista–Controlador (MVC), que permite una separación limpia entre las distintas capas de la aplicación. Para la capa de presentación (la vista) se buscó un framework que proporcionara mayor flexibilidad en la elaboración de las pantallas; mapeo entre formularios y sus clases en el servidor; validación y conversión de datos en cualquier dirección; gestión de errores; posibilidad de inclusión de componentes complejos (menús, árboles, AJAX) de una forma sencilla; que brindara un mecanismo sencillo de comunicación con las capas adyacentes y que fuera muy fácil de mantener. Para esta capa se eligió JavaServer Faces. Debido al framework seleccionado para la capa de presentación, se necesita de un gestor de plantilla que manipule la forma y la manera en la que se muestren los campos de las páginas, para de esta manera separar la gestión de los contenidos de la forma en que se debería mostrar. En este sentido se opta por el uso del framework Facelets y XHTML como plantilla. También se decide el use del framework Rich Faces para dotar a la aplicación de la capacidad de AJAX.

3.2 JavaServer Faces

JSF es un framework de Java para la capa de presentación de aplicaciones Web que hace más fácil el desarrollo de dichas aplicaciones, brindándole una interfaz basada en componentes ricos y poderosos (22). Se basa en la experiencia adquirida con tecnologías como HTTP, Servlet, JavaBeans y JSP.

Por años los desarrolladores basaron sus aplicaciones en los conocidos Servlet, en JavaServer Pages (JSP) o Struts, siendo estos los padres de JSF. El auge vertiginoso del desarrollo web en los últimos tiempos ha dado como resultados interfaces más ricas e interactivas y con ello la necesidad de nuevas tecnologías para lograr estas interfaces. Otro problema que emergió con el desarrollo de Internet fue la creación de componentes personalizados con un alto grado de complejidad. JSF nace para darle solución a estas necesidades.

Como ilustra la siguiente figura, JSF se ejecuta del lado del servidor renderizando las páginas que el cliente solicita.

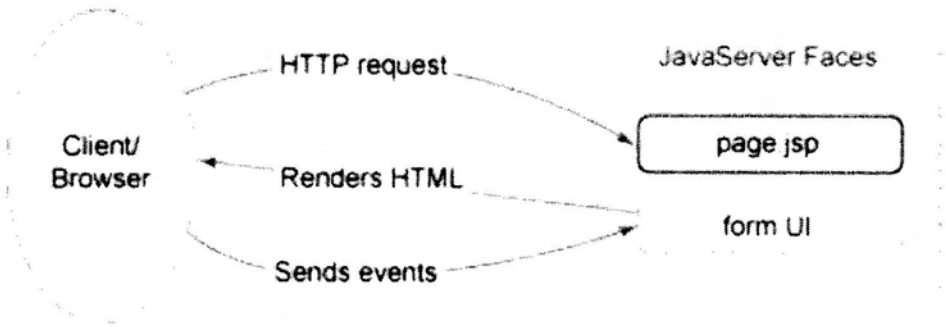


Figura 3.2.1: Proceso de una petición JSF (23)

Por ser un framework que se ejecuta del lado del servidor el mismo necesita de un contenedor Web como Apache Tomcat y el servidor de aplicaciones de Oracle o el de IBM (WebSphere) para funcionar.

Los dos componentes fundamentales de la tecnología son: la librería de etiquetas para JSP y el API para manejo de eventos, validadores, conversores, etc.

Resumiendo, JSF es un framework de Java basado en el patrón MVC que define una arquitectura de componentes y contiene un conjunto estándar de componentes de interfaz de usuario, así como una librería de etiquetas JavaServer Pages (JSP) personalizadas para dibujar componentes UI dentro de una página JSP. Este modelo de programación y la librería de etiquetas para componentes de interfaz de usuario facilitan de forma significativa la tarea de la construcción y mantenimiento de aplicaciones Web con interfaz de usuario del lado del servidor. JSF hace al desarrollador pensar en componentes, eventos, beans de respaldos y sus interacciones en vez de peticiones, respuestas y marcas.

Beneficios de la Tecnología JavaServer Faces

Una de las ventajas más reconocida de JSF es la clara separación que establece entre el comportamiento y la presentación. JSP fue un buen motor impulsor en esta tendencia, aunque no lo logró del todo. Ya con JSF es posible manejar eventos complejos, mantener el estado de los componentes, mapear peticiones HTTP, entre otras características que avalan su apogeo. Su arquitectura está pensada para fomentar la reutilización, la separación de roles y la facilidad a la hora de usar las herramientas.

Relacionado con la ventaja anterior está la eficiencia en el desarrollo de aplicaciones Web enfocada desde el punto de vista organizativo. El bajo acoplamiento entre la lógica y la presentación propicia que cada integrante del equipo de desarrollo centre sus esfuerzos en la fracción que le corresponde.

Lo más importante de la tecnología JavaServer Faces es que proporciona una rica arquitectura para manejar el estado de los componentes, procesar los datos, validar la entrada del usuario y manejar eventos.

Una de las ventajas de JSF es que por ser una especificación existen implementaciones de distintos fabricantes. Esto permite no atarse con un proveedor y poder seleccionar el que más se adecúe a las necesidades según: número de componentes que proporciona, rendimiento, soporte, precio, política,...

Para este trabajo se utiliza la implementación JSF RI de Sun Microsystem. Aunque existen otras implementaciones como la de Apache MyFaces es la de Sun la que ha pulido más las deficiencias encontradas. Además, posibilita el uso de las librerías Tomahawk de la misma compañía. [1]

Conceptos fundamentales de JSF

El primer concepto y el más visible para el usuario final son los componentes de la interfaz. Dichos componentes no son más que JavaBeans que se ejecutan del lado del servidor y por ello es posible mantener el estado de los mismos en todo momento. La organización de los componentes UI se realiza en arboles de vistas que poseen una representación especifica (*renderer*) o que pertenecen a una familia de representaciones (*kits de renderer*). Estos componentes luego de renderizarse llegan al usuario final en una representación HTML.

Asociado a los componentes se encuentran los eventos que generan estos componentes. El modelo para tratar los eventos que define JSF está basado en *Listeners (oyentes)* que se implementan en los beans de respaldo. Básicamente existen cuatro tipos de *Listeners*: Value-change events, Action events, Data model events y Phase events.

Los validadores es otro de los pilares de JSF. Como su nombre lo indica son los encargados de asegurar que los datos proporcionados por el usuario sean correctos. Con la gestión de validación de JavaServer Faces se evitar tener que escribir código Java o JavaScript para lograr la validación en la mayoría de los casos. Aunque JSF define un conjunto de validadores estándar como: campo con valor requerido, validador de longitud de cadena y validador de rango para enteros y decimales, aunque es posible definir uno personalizado.

Para ajustarse a las nuevas tendencias JSF también soporta la internacionalización y la localización. La internacionalización es la habilidad de una aplicación para soportar varios lenguajes en dependencia del lugar donde nos encontremos; mientras que la localización es el proceso de modificar una aplicación para que soporte la lengua de una región. JSF posee la infraestructura para estas dos cuestiones, no las traducciones.

Otra de las prestaciones del framework en cuestión son los conversores. Su función es la de convertir el valor de un componente a una cadena y viceversa. JSF provee algunos e igual que los validadores se pueden personalizar. Fundamentalmente lo usa el *renderer* para mostrar los datos al usuario.

El manejo de la navegación es otro de los puntos fuertes y novedosos de este framework. En JSF la navegación es controlada por el *manejador de navegación.* Este componente es el que se encarga de la redirección hacia otras páginas en dependencia de los resultados o de las reglas de navegación.

El último concepto de la tecnología y no por ello menos importante son los beans de respaldo. Estos no son más que JavaBeans especializados que almacenan valores de los componentes de interfaz de usuario e implementan los métodos para los oyentes de eventos. Los mismos pueden en algunas circunstancias almacenar referencias a componentes visuales.

Como es de suponer entre todos estos elementos existe una relación. Por ejemplo, los componentes UI son los que actualizan los beans de respaldo y generan los eventos según las entradas de los usuarios; los renderizadores muestran los componentes y pueden además generar eventos y mensajes sobre los mismos; los convertidores traducen y formatean los valores de los componentes a mostrar y los validadores aseguran que los datos de estos componentes sean correctos.

Ciclo de vida de una petición JSF

Hasta el momento se ha hablado de JSF de un modo general y ya es hora de comenzar a adentrarse en el funcionamiento del framework. Los conceptos descritos en la sección anterior son parte también del procesamiento de una petición del cliente. Los pasos por los que transita este proceso serán el tema central de este epígrafe.

El tratamiento de la petición comienza en el momento en que el Servlet de JSF recibe la petición del cliente. Básicamente son seis fases por las que debe transitar la petición. Posteriormente a la mayoría de estas es que los eventos son procesados.

La primera de estas fases es la restauración de la vista. En esta fase es donde se encuentra o se crea el árbol de componentes para la vista en cuestión. Existe un grupo de componentes UI que pueden

generar eventos de tipo *action* en esta etapa. El siguiente hito es aplicar los valores de la petición, o sea, actualizar los valores de los componentes con los valores de la petición utilizando los convertidores en caso de ser necesarios. El uso de los convertidores puede generar mensajes de error por el proceso de conversión. En esta etapa generalmente se generan eventos basados en los parámetros del *request*. A esto sigue el proceso de validación, instando a cada componente a realizar su propia validación o incluso a utilizar validadores externos. La validación también puede generar mensajes de error que deben ser mostrados al usuario final. El próximo peldaño es la actualización de los datos del modelo, es decir, actualizar todos los datos de los beans de respaldo o de los componentes asociados con el modelo de objetos. Este es el momento en que los errores de conversión pueden ser reportados. Seguidamente se pasa a la invocación de la aplicación llamando a algún oyente de acción (*action listener*). Este oyente ejecuta los métodos de acción referenciados por los componentes y selecciona la próxima vista a mostrar. El último de estos pasos es renderizar la respuesta utilizando la alguna tecnología de presentación como JSP.

Todo el proceso que anteriormente se describió puede ser resumido con la siguiente imagen:

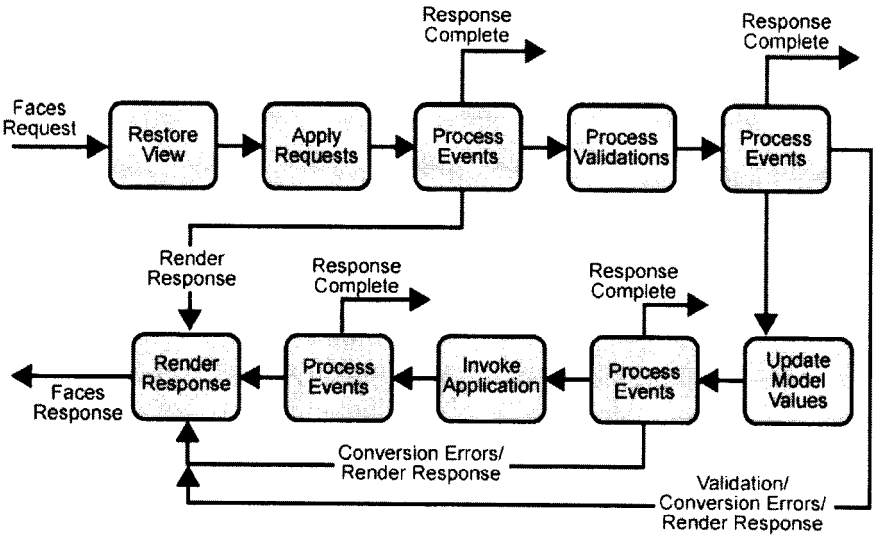


Figura 3.2.2: Ciclo de vida de una petición JSF (23)

El entendimiento de este procedimiento no es fundamental para trabajar a pequeña escala en JSF, aunque se vuelve de importancia cuando se intenta crear componentes complejos o detectar errores poco documentados en la capa de presentación. Es recomendable para una comprensión amplia del framework dominar las fases anteriormente explicadas y otras asociadas a ellas, aunque en la práctica no sea tan palpable su utilidad.

3.3 Rich Faces

Rich Faces es un framework de código abierto para añadir comportamiento AJAX a una aplicación JSF sin necesidad de adentrarse en JavaScript. Con este framework es posible intensificar el poder de JSF al incorporarle AJAX y poder seguir utilizando los convertidores, validadores y demás bondades de JavaServer Faces dentro del ciclo de peticiones-respuesta de AJAX.

Este framework provee dos librerías de componentes: el *core ajax* y la *UI*. La primera de ellas es la encargada de adicionar comportamiento AJAX a las páginas ya existentes sin necesidad de escribir código JavaScript o tener que reemplazar algún componente. La librería de UI provee una amplia variedad de componentes para crear de un modo rápido complejas y ricas vistas en aplicaciones JSF. También es posible definir un componente propio por la capacidad de extensión que posee Rich Faces. Actualmente cuenta con una amplia gama de componentes que cada día crece.

Otro aspecto que hace al framework en estudio interesante es su poder para generar dinámicamente recursos como: sonido, imágenes, hojas electrónicas de cálculo como Excel, gráficos, entre otros muchos elementos que le dan más dinamismo a la web.

Fue desarrollado basado en una arquitectura abierta para ser compatible con una amplia variedad de ambientes. Para trabajar con Rich Faces necesita:

- Java
- JavaServer Faces
- Un servidor de aplicaciones para Java o un contenedor de servlet
- Un navegador del lado del cliente.

Para funcionar necesita de una versión del SDK de Java superior o igual a la 1.4 y de alguna de las siguientes implementaciones de JSF:

- Sun JSF 1.1 RI - 1.2
- MyFaces 1.1.1 - 1.2
- Facelets JSF 1.1.1 - 1.2
- Seam 1.2. - 2.0

Como antes se había mencionado esta tecnología necesita de un contenedor de Servet. Actualmente en el mercado existe una buena cantidad de ellos. Algunos de los que soportan Rich Faces son:

- Apache Tomcat 4.1 - 6.0

- IBM WebSphere 5.1 - 6.0
- BEA WebLogic 8.1 - 9.0
- Oracle AS/OC4J 10.1.3
- Sun Application Server 8 (J2EE 1.4)
- JBoss 3.2 - 4.2.x

Del mismo modo que por el lado del servidor requiere de una aplicación en el lado del cliente necesita de un navegador web. En estos días son muchos los navegadores de moda aunque no todos soportan muchos de los nuevos elementos que se están incluyendo en las páginas web de estos tiempos. Los navegadores que el personal de soporte de este framework acredita como aptos para trabajar con esta tecnología son:

- Internet Explorer 6.0 - 7.0
- Firefox 1.5 - 2.0
- Opera 8.5 - 9.0
- Netscape 7.0
- Safari 2.0

Los anteriores son los requerimientos para trabajar con esta tecnología de AJAX. Es el momento de estudiar los conceptos fundamentales por los que esta se rige.

Este framework es implementado como una librería de componentes para darle soporte AJAX a las páginas web. Es posible definir los eventos en la página que invocarán una petición AJAX así como las zonas de la página que deben ser sincronizadas con el árbol de componentes JSF luego de que la petición cambie los datos en el servidor de acuerdo al evento que se disparó en el cliente.

La siguiente figura muestra su funcionamiento:

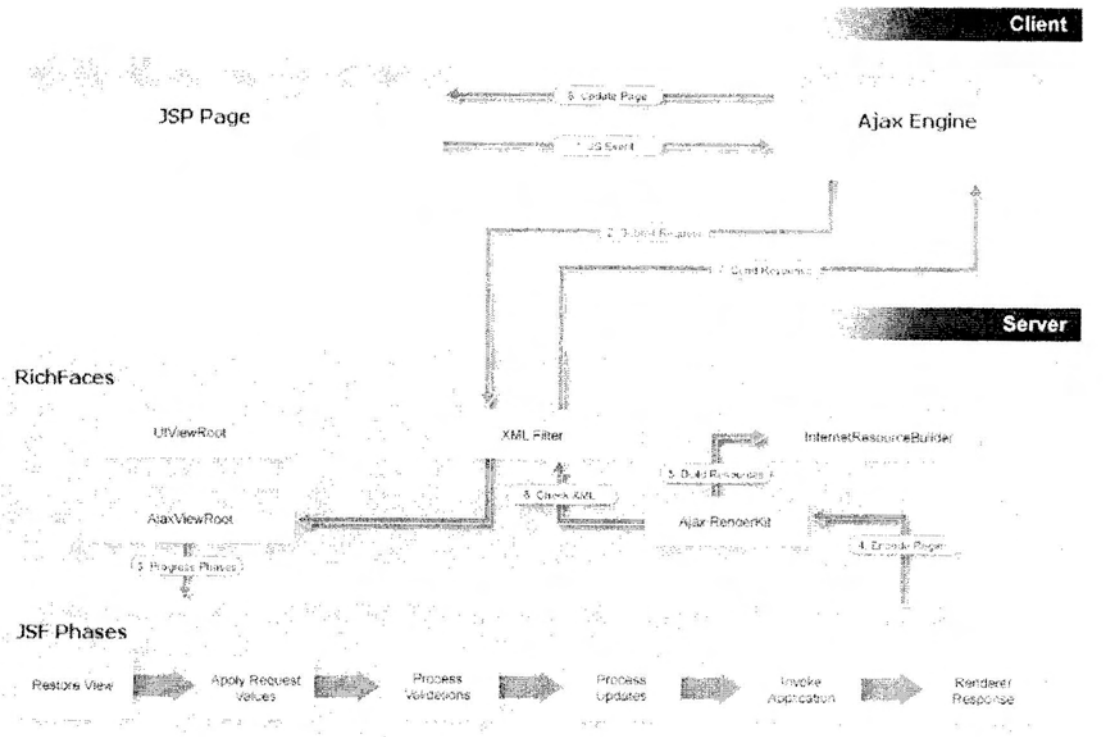


Figura 3.3.1: Funcionamiento de una petición AJAX con Rich Faces (24)

Elementos arquitectónicos fundamentales

Son cinco los principales componentes de la arquitectura de Rich Faces. A continuación se abordan estos módulos en más detalle.

Ajax Filter

Lo primero que necesita para trabajar es registrar un filtro en el web.xml de su aplicación. Este filtro es el encargado de reconocer una multitud de tipos de peticiones. Cuando llega la petición para un recurso el filtro de Rich Faces busca en la cache de recursos por el mismo, si lo encuentra allí lo envía al cliente. Si ocurre el caso contrario el filtro busca entre los recursos registrados por el ResourceBuilder. Si el recurso está registrado, el filtro envía una petición al ResourceBuilder para que cree el recurso.

AJAX Action Components

Aquí es donde radican los componentes de acción del framework tales como: AjaxCommandButton, AjaxCommandLink, AjaxPoll and AjaxSupport, etc. Los mismos se pueden utilizar para enviar peticiones AJAX desde el lado del cliente.

AJAX Containers

El contenedor de AJAX es sencillamente una interfaz que describe una porción de la página JSF que debe ser decodificada durante una petición asíncrona. Las clases AjaxViewRoot y AjaxRegion implementan dicha interfaz.

JavaScript Engine

Este componente se ejecuta del lado del cliente. Es el responsable de conocer que áreas de la página JSF deben ser actualizadas basadas en la información que contiene la respuesta a la petición. No es recomendable utilizar este código directamente, dado que se encuentra disponible automáticamente.

Optimización de peticiones AJAX

Los atributos AJAX son comunes en componentes como: <a4j: support>, <a4j: commandButton>, <a4j: jsFunction>, <a4j: poll>, <a4j: push> entre otros. Estos atributos ayudan a Rich Faces a exponer sus características y la mayoría de ellos poseen valores por defecto, o sea, se puede comenzar a trabajar con el framework sin conocer su uso. Sin embargo, su uso permite refinar el rendimiento de AJAX. Uno de los atributos fundamentales es el "*reRender*". Este atributo permite indicar el área o las aéreas de la página a ser actualizadas en la interacción JSF. El valor del atributo es el identificador del componente JSF a ser actualizado o una lista de ellos. A continuación un ejemplo de uso:

```
<a4j: commandButton value="update" reRender="infoBlock"/>

    ...
<h: panelGrid id="infoBlock">

    ...
</h: panelGrid>
```

El fragmento de código anterior indica al botón que debe actualizar el control con identificador infoBlock. Como es de suponer el componente o los componentes en cuestión deben ser localizados para actualizarlos dentro del árbol de componentes. Este proceso implica una serie de pasos que es deseable optimizar. El valor de este atributo puede ser expresado de forma que indique más información sobre el componente y ayude a su búsqueda de un modo más eficiente, por ejemplo: reRender=": infoBlockl: sv: infoBlock2". En este caso se indica que el primer identificador pertenece a

la vista principal y en el segundo caso se indica que esta dentro de la subvista "sv". También mediante el lenguaje de expresiones de puede ordenar la actualización de propiedades de las clases.

El atributo "eventsQueue" define el nombre de la cola empleada para ordenar las peticiones AJAX, las cuales por defecto no se encolan. Cuando se disparan dos eventos simultáneamente las peticiones llegan al servidor igual de parejas y JSF no garantiza que se atiendan exactamente por el orden en que llegan, haciendo el comportamiento para estos casos impredecibles. Definiendo explícitamente este atributo se elimina este tipo de problema. También puede darse el caso que el usuario haga un número elevado de peticiones similares al servidor. En este caso el uso de este atributo evita que se sobresature el mismo con peticiones innecesarias. Solamente las nuevas serán encoladas.

Otro atributo relacionados con el tema es el "requestDelay". Como su nombre indica establece un límite de tiempo en milisegundo que se espera antes de enviar una nueva petición al servidor o de renovarla por otra similar si existe. Asociado al atributo anterior está "ignoreDupResponses", indicando que se debe ignorar una respuesta si se ha encolado una petición más novedosa para la cual dicha respuesta se corresponde.

Opciones del procesamiento de datos

Rich Faces utiliza un enfoque basado en formularios a la hora de hacer las peticiones. Cuando se produce una petición asíncrona todos los datos del formulario JSF involucrado son enviados al servidor. Este comportamiento puede ser ineficiente en formularios que poseen una gran cantidad de información. Para evitar esto existe el atributo "ajaxSingle", el cual indica si está en *true* que sólo se debe incluir el valor del componente con el que se asocia además de los parámetros incluidos en <f: param> o <a4j: action> si existiera alguno.

Interacción con JavaScript

Aunque Rich Faces fue hecho para dotar de capacidad *AJAX* a una aplicación JSF sin necesidad de escribir código JavaScript, es posible incluir código de este tipo si fuera necesario. Existe una buena cantidad de atributos para lograr esta tarea.

Por ejemplo, el atributo "onsubmit" permite invocar un determinado código JavaScript antes de enviar la petición. El atributo "onclick" realiza tareas similares pero para este caso en componentes donde se posible disparar un evento al dar un clic. Otro indicador importante es "oncomplete", el cual se emplea para ejecutar código propio justo después que la respuesta *AJAX* es devuelta.

Rich Faces permite serializar a JSON no sólo tipos primitivos, también tipos complejos como arreglos y colecciones. Los beans por ejemplo son serializados y se puede referir a ellos a través de *data*.

La cantidad de atributos interesantes es un poco extensa. La mayoría de ellos se conoce y aprende sobre la marcha. Los anteriores son sólo algunos útiles a la hora de lograr un buen rendimiento en las aplicaciones.

3.4 Facelets

Facelets es un sistema de gestor de plantillas para aplicaciones que utilizan JavaServer Faces, lo que sería Tiles para Struts (25). Facelets es un proyecto de JavaServer Faces que maneja el viewhandler de una aplicación JSF. Debido a que es un proyecto para JSF, Facelets no pude ser utilizado en ningún proyecto que no utilice algunas de las implementaciones de la especificación de JavaServer Faces. Facelets permite definir vistas JSF utilizando plantillas en formato HTML, reduciendo de esta forma código innecesario para agregar componentes en la vista y que no necesariamente sea un contenedor web. Facelets es un framework simplificado de presentación, en donde es posible diseñar de forma libre una página web y luego poderle asociarle los componentes JSF necesarios para el diseño. Ofrece mayor libertad a los diseñadores y mejora los informes que tiene JavaServer Faces, entre otras cosas.

Facelets no sólo es un framework de plantillas para JSF. También posibilita la creación de tag para los componentes personalizados utilizados en las aplicaciones JavaServer Faces, lo que posibilita la necesidad de no escribir los JSF para utilizar dichos componentes en las vistas. También con el uso de Facelets se le puede agregar a los componentes JSF propiedades nuevas en dependencia de las necesidades del programador. Esto le permite a los desarrolladores extender las funcionalidades de los componentes JSF de diferentes formas y en el momento que lo necesiten.

Unas de las cosas más importantes que aporta Facelets es que es el responsable y encargado de construir el árbol de componentes de una página JSF. Esto se debe a que en Facelets interviene el viewhandler de la aplicación JSF. Facelets también resuelve los problemas que existían entre las tecnologías JSF y JSP en el desarrollo de aplicación JSF, ya que JSP lo que hace es generar los contenidos que se van a mostrar mediante un servlet y no se encargaba de construir el árbol de componentes JSF. Está labor la tendría que hacer el mismo framework de JSF mediante la etiqueta <f: view> incrustada en las plantillas JavaServer Pages.

Facelets también se integra con otros librerías elaboras para ser integradas en las aplicaciones que utilizan la especificación de JSF como Myfaces Tomahawk, JSTL, entre otros. Integrar Facelets con

aplicaciones JSF no es difícil, sólo hay que modificar el fichero de configuración principal de JSF de la aplicación y especificar el elemento <view-handler>. Esto es posible debido a la interoperabilidad que posee la especificación de JSF a la hora de brindar a los demás framework modificar los elementos que componen la aplicación JSF.

```
<faces-config>
 <application>
  <view-handler>
   com.sun.facelets.FaceletViewHandler
  </view-handler>
 </application>
</faces-config>
```

Por otro lado Facelets no es dependiente de un contenedor JSP, lo que significa que una aplicación que utilice Facelets pueda a empezar a utilizar las nuevas características de JSF 1.2 sin esperar a que el contenedor tenga soporte para JSP 2.1. Facilita el diseño de las páginas mediante el uso de la propiedad jsfc en las etiquetas HTML, la cual es utilizada para decirle al compilador que sustituya dicha etiqueta por el componente especificado en el atributo jsfc. Facelets tampoco necesita de un contenedor de servlets para definir y probar las vistas JSF, lo que abre las puertas a JSF para poder utilizarse fuera de los servlets, incluyendo a Portlets. Facelets es un framework de código abierto (open source) y que utiliza a XHTML como formato para definir las plantillas.

3.5 XHTML

La tecnología XHTML o lenguaje XHTML (Lenguaje eXtensible de Marcado de Hipertextos) (26) es un lenguaje de marcado diseñado para sustituir la tecnología anteriormente vista (HTML) como estándar para las páginas web. Está tecnología es la versión de XML de HTML, por lo que básicamente tiene las mismas funcionalidades, capacidades y cumple las especificaciones más estrictas de XML. El objetivo de XHMTL es lograr una web semántica, donde la información y la forma de presentarla estén separadas. En este sentido XHTML se utilizaría sólo para transmitir la información que contiene un

documento, dejando a JavaScript y a las CSS para su aspecto. Básicamente XHTML tiene las mismas funcionalidades que HTML.

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Técnicamente, HTML es un lenguaje descendiente del lenguaje SGML, mientras que XML, es también hijo del lenguaje SGML.

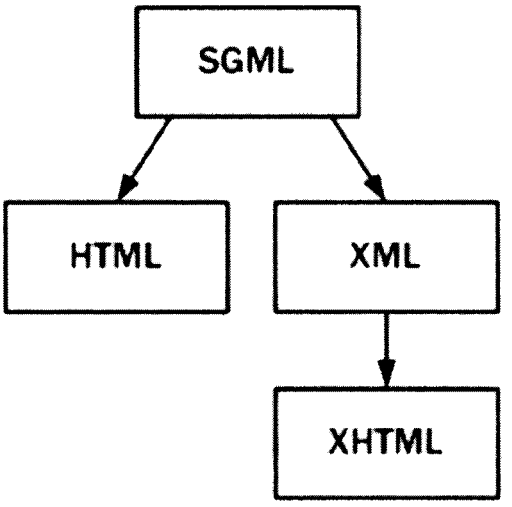


Figura 3.5.1: Esquema de evolución de HTML y XHTML

Los documentos y páginas creados con XHTML son muy similares a las páginas y documentos creados con HTML. Actualmente, entre XHTML 1.0 y HTML 4.0.1, los desarrolladores Web siempre escogen a XHTML para la creación de los documentos electrónicos. En un futuro, no muy lejano, los desarrolladores tendrán un gran desafío en elegir el lenguaje para la creación de sus documentos electrónicos entre HTML 5 y XHTML 1.1 ó XHTML 2.0.

XHTML y las CSS también juegan un papel importe en el diseño de los documentos electrónicos, debido a que los son el mecanismo de separación entre los contenidos definidos mediante XHTML y el aspecto de esos contenidos en la pantalla de los usuarios. De esta manera, los documentos XHTML creados son más flexibles, ya que se pueden adaptar mejor a los diferentes dispositivos, como: pantallas de ordenadores, impresoras, pantallas de móviles y dispositivos de deferentes naturalezas.

Un documento XHTML normalmente está formado por tres elementos generales: HTML, head y body. La etiqueta html representa al documento electrónico, que es la que contiene todo lo referente al documento como cabecera, cuerpo, contenido, imágenes y otras cosas más. La etiqueta head

representa la cabecera del documento. Esta etiqueta se encuentra dentro de la etiqueta HTML, por lo que es hija de dicha etiqueta. Dentro de la etiqueta head podemos encontrar información como el título del documento, idioma, entre otros. La etiqueta body, como su nombre lo indica, representa al cuerpo de un documento XHTML, en cual se encuentra todo el contenido que el usuario observa en la pantalla, como párrafos, imágenes, audio entre otros. La etiqueta body también es descendiente de la etiqueta HTML.

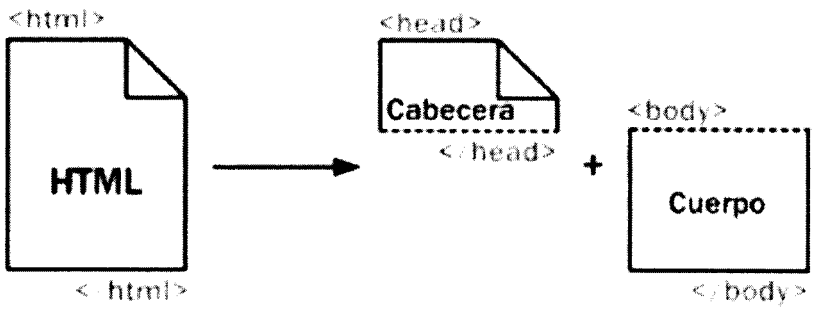


Figura 3.5.2: Esquema de un documento XHTML

XHTML define el término de elemento para referirse la las partes que componen los documentos XHTML. A veces es común referirse a los elementos como etiqueta. Un elemento XHTML está formado por:

1- Una etiqueta de apertura.

2- Cero o más atributos.

3- Puede contener cualquier tipo de contenido.

4- Una etiqueta de cierre.

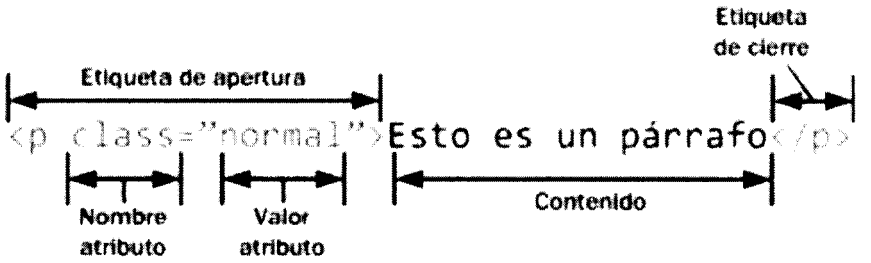


Figura 3.5.3: Partes que componen un elemento XHTML

Una de los principales beneficios que aporta el uso de XHTML en la creación de documentos electrónicos, es la posibilidad de obtener documentos bien formateados, que contienen una buena estructuración de sus contenidos. Con XHTML se puede lograr construir documentos que puedan adoptar diseños radicalmente distintos en diferentes dispositivos, pudiendo incluso escogerse entre varios diseños para el mismo medio. También permite la edición directa del código y mantenimiento de los documentos electrónicos elaborados con el lenguaje XHTML. Por otro lado, es compatible con los nuevos estándares que actualmente está desarrollando el W3C como recomendación para los futuros agentes de usuario y navegadores. Los documentos escritos en XHTML pueden potencialmente presentar mejor rendimiento en las actuales herramientas Web. Permite definir etiquetas personalizadas y no necesita de un contenedor de Web para ser procesado. Es una tecnología que puede ejecutarse en cualquier plataforma (Windows, Linux). Es utilizado por varios Motores de Plantillas (facelets) como formato base para la generación de plantillas. XHTML es una tecnología que pertenece a los estándares Web de la W3C.

3.6 Reglas de optimización del frontend de las aplicaciones Web

En el desarrollo de software, orientado a la web, surgen varias interrogantes que giran sobre el tema ¿rendimiento en los sitios Web? En los inicios de la era de las aplicaciones Web, los desarrolladores buscaron la respuesta de las preguntas sobre el tema en el Backend de las aplicaciones, concentrándose en las opciones de compilación, indexación de las bases de datos y manejo de la memoria para ganar en rendimiento. Estos esfuerzos fueron posibles debido a la abundante documentación y libros existentes que enfatizaban cómo obtener rendimiento en una aplicación en estas áreas. En el mundo de la programación web el rendimiento no se gana optimizando el Backend, sino tratando de optimizar lo más posible el Frontend de las aplicaciones. Las peticiones HTTP a un servidor Web constan de dos partes: una parte es la solicitud del recurso (HTTPrequest) y la otra es la entrega del recurso (HTTPResponse). En este sentido el tiempo total de la respuesta de una página es el tiempo que transcurre desde el momento en que se inicia la petición del recurso menos el tiempo en que se demora en visualizarlo el navegador.

En las peticiones de una página Web, no más del 20% del total del tiempo de la petición se utiliza para transportar el documento HTML correspondiente de la petición desde el servidor Web hasta el navegador del usuario. Si lo que realmente se quiere es ganar rendimiento en una aplicación Web, hay que tratar de disminuir el tiempo total de las respuestas de las páginas. ¿Dónde se encuentra el restante 80 %? Por investigaciones realizadas por un conjunto de desarrolladores envueltos en el

tema, se llegó a la conclusión que el restante 80% se encuentra dentro de los contenidos del documento HTML tales como imágenes, sonido, pdf, CSS, JavaScript y el mismo contenido de texto del documento. Cuando el navegador obtiene el documento HTML empieza a parsearlo y cada vez que se encuentra un recurso como una imagen ó un script, los solicita al servidor para poderlo mostrar en la pantalla del usuario. En este proceso el navegador no sigue parceando el documento hasta que el archivo solicitado no allá sido devuelto por el servidor al navegador. Una vez ya descargado el archivo se muestra en la pantalla del usuario y en navegador continúa parseando las demás líneas y solicitando archivos al servidor en caso que los necesite. Debido a esto, para poder ganar en rendimiento en una aplicación Web hay que concentrarse en el restante 80 % del total de la respuesta de la página.

La principal idea que existe para ganar en rendimiento en las aplicaciones web, es disminuir la cantidad de peticiones HTTPRequest por parte del navegador al servidor Web. Es decir, cuando se disminuye la cantidad de peticiones HTTPRequest por parte del navegador web al servidor, el navegador web interpreta más rápido los documentos HTML y entonces estos se muestran más rápido en la pantalla del usuario. Con esta forma se disminuye el tiempo total de las respuestas de las páginas. Entonces, la interrogante es:¿Cómo se puede reducir?

Para lograr reducir el tiempo total de las respuestas de las páginas en las aplicaciones Web, hay que aplicar las 14 reglas de Steve Souders planteadas en el libro "High Performance Web Sites". El autor explica la importancia de optimizar el Frontend para buscar rendimiento en la aplicación Web. Aplicando dichas reglas podemos lograr reducir los tiempos de respuesta de las páginas. Algunas de dichas reglas son expuestas a continuación.

1- Reducir el número de peticiones
2- Cachear elementos del documento
3- Comprimir elementos del documento
4- Colocar los CSS en la cabecera de la página
5- Colocar los JavaScript en la parte baja de la página
6- Colocar los JavaScript y CSS en archivos externos
7- Reducir DNS Lookups
8- Reducir los archivos JavaScript
9- Remover los script duplicados
10- Cachear las peticiones Ajax

Cuando se aplican todas estas reglas en conjunto podemos lograr rendimiento en el Frontend de la aplicación y de esta forma reducir entre un 40 y 50 % el tiempo total de las peticiones de las páginas.

En la actualidad en varias aplicaciones Web que aplican estas reglas para disminuir los tiempos de respuestas de sus páginas, tales como:

- www.gmail.com
- www.yahoo.com
- www.msn.com
- www.wikipedia.org

Steve Souders en su libro expone como ejemplo el sitio de Yahoo.

Cuando se visita la página de yahoo (www.yahoo.com) y se descarga dicha página con Internet Explorer, sucede lo siguiente. La primera vez que se visita la página, debido a que la caché del navegador esta *empty*, el 5% del total del tiempo de la respuesta pertenece a la transmisión del documento HTML desde el servidor al navegador. El navegador comienza a interpretar el documento y comienza a descargar los componentes que se encuentran en el cuerpo de la página como imágenes, CSS y JavaScript. Esto representa el 95 % del total del tiempo de la repuesta (Figura 3.6.1).

Cuando se visita de nuevo la página del Yahoo con Internet Explorer, el documento HTML es solamente el 12 % del total del tiempo de respuesta, mientras que los demás componentes no se tienen que descargar ya que están cacheados en el navegador (Figura 3.6.2).

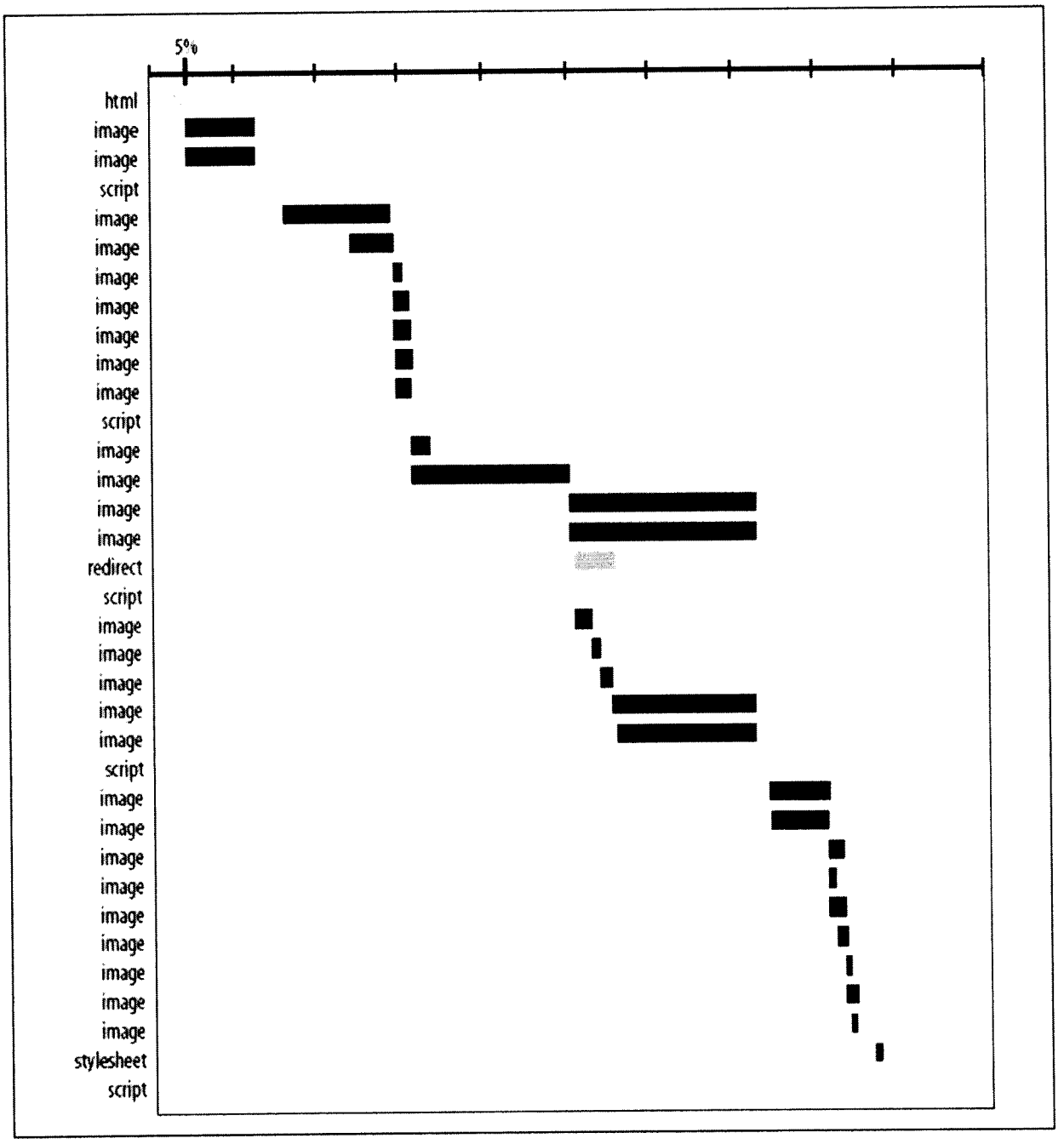


Figura 3.6.1: Descarga de la página principal de YAHOO con la cache empty

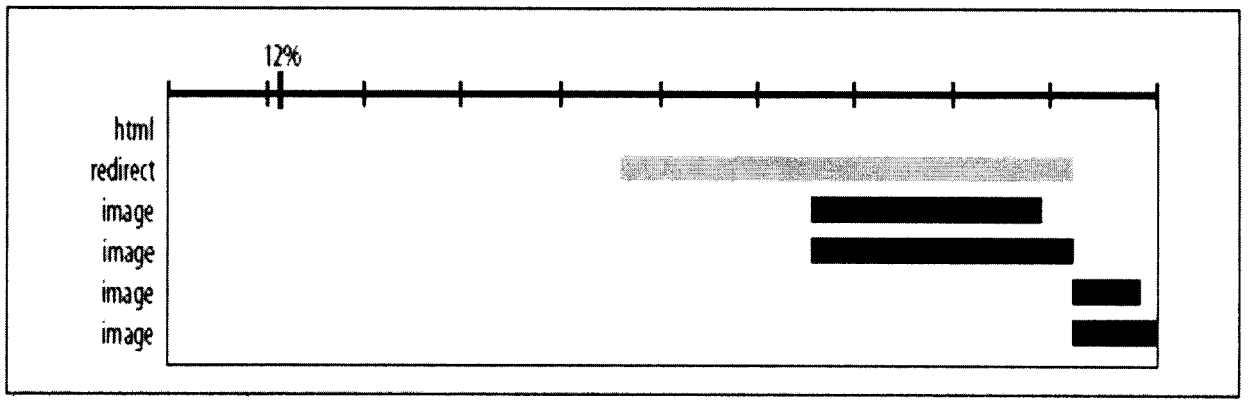


Figura 3.6.2: Descarga de la página principal de YAHOO con la cache activa

3.6.1 Reducir el número de peticiones

La idea es reducir el número de peticiones al servidor. Esto se logra reduciendo la cantidad de componentes que tengan que ser descargados por el navegador. En la mayoría de las aplicaciones web, las páginas se diseñan con un elevado número de imágenes, por lo que si disminuimos la cantidad de imágenes, el tiempo total de la respuesta será menor y así la página se mostrará más rápido en la pantalla del cliente. Para reducir la cantidad de imágenes se recomienda integrar la mayor cantidad de imagen en una sola o realizar un Map de imágenes en caso que sea posible.

Figura 3.6.1.1: Map de imágenes.

En una página que contiene cuatro imágenes, el navegador, al interpretar el documento HTML, tiene que descargar desde el servidor las cuatro imágenes, por lo que efectuaría cuatro peticiones HTTPRequest al servidor para obtener dichos recursos. Si las cuatro imágenes se integran en una sola, como en la figura 3.5.3, el navegador web sólo haría una petición HTTPRequest. Esta técnica también se les puede aplicar a los CSS Sprites. De esta forma se disminuye la cantidad de peticiones al servidor por el navegador y se logra disminuir el tiempo total de la respuesta de las páginas.

3.6.2 Cachear elementos del documento

Otra forma de obtener rendimiento en las aplicaciones Web es cacheando en el navegador algunos elementos que por necesidad no tendrán que ser modificados en el tiempo. La idea es determinar cuáles son estos componentes y cachearlos en el navegador por un tiempo prolongado para que este no tenga que descargarlo de nuevo desde el servidor. Los componentes candidatos para hacerle esta técnica son los CSS, JavaScript, las imágenes de diseño de las páginas. Cuando se visita por primera vez una página contiene, la cual contienen CSS, JavaScript e imágenes, el navegador descarga del servidor los elementos mencionados. Si dichos elementos se cachean en el navegador, la segunda vez que se visite la página, el navegador no tiene la necesidad de descargar los archivos desde el servidor, ya que los toma del cliente. De esta forma se disminuye el número de peticiones HTTP por el navegador al servidor y las páginas se carga más rápido.

3.6.3 Comprimir elementos del documento

La compresión de elementos del documento HTML es otra de las vías para ganar en rendimiento en las aplicaciones Web. Esta técnica no consiste en reducir el número de peticiones, sino en acelerar la transmisión de los recursos pedidos por el navegador al servidor por el canal de comunicación. Cuando el navegador pide un recurso al servidor, este tiene que viajar a través del canal de comunicación. Si dicho recurso es muy grande, se tardará en llegar desde el servidor al navegador. Lo ideal es comprimir lo más que se pueda los recursos en el servidor como CSS, JavaScript y contenido del cuerpo de la página, para que se transmita más rápido por el canal y de esta forma se pueda presentar la página solicitada más rápido en la pantalla del cliente.

Para lograr esta técnica el navegador debe soportar GZIP, que es una de los algoritmos utilizados por el servidor para comprimir los recursos. Cuando el recurso llega al navegador, este detecta que esta comprimido y los descomprime para sí visualizarlo en la pantalla del cliente.

Con esta técnica se recomienda sólo comprimir archivos como CSS, JavaScript y textos. Las imágenes, sonidos y pdf no se comprimen, ya que estos formatos ya están reducidos a lo más que se puede.

3.6.4 Colocar los CSS en la cabecera de la página

La técnica de poner los estilos CSS en la cabecera de los documentos HTML no reduce el número de peticiones, sino que trata que los datos en la pantalla de usuario se muestra más rápido. Generalmente las páginas Web contienen elementos que dependen de los estilos CSS para mostrarse correctamente

en la pantalla del usuario. Los estilos CSS son los que contienen las reglas de estilos para los elementos. Si estos estilos CSS no se cargan antes de que el navegador interprete los elementos que necesitan de esos estilos, dichos elementos no se mostrarán correctamente en la pantalla del usuario. Por lo que pone de manifiesto, la importancia de poner los estilos CSS en la cabecera de página para que este se muestre correctamente.

3.6.5 Colocar los JavaScript en la parte baja de la página

La técnica de poner los JavaScript en la parte baja de la página no reduce el número de peticiones, sino que trata que los datos en la pantalla de usuario se muestren más rápido. Generalmente las páginas no necesitan de los script de JavaScript para que el contenido de la página se muestre. Si se ponen los script en la cabecera de la página, el navegador los interpreta y hasta que no termine, los demás elementos de la página no serán interpretados por el navegador. Esto provoca que las páginas que contienen los script JavaScript en la cabecera se mostraran más lentas. Poniendo los script JavaScript al final de la página, el navegador interpretará los elementos que se tienen que mostrar en la pantalla de usuario y de esta manera se mostrarán dichos elementos en la pantalla de los usuarios y después interpretará los scripts de JavaScript. De esta forma se mostrará en la pantalla del usuario la información que necesiten de forma rápida.

3.6.6 Colocar los JavaScript y CSS en archivos externos

Esta técnica no reduce el número de peticiones al servidor por parte del navegador, sino que trata que las páginas se muestren más rápido en la pantalla del usuario. Los script JavaScript y CSS se agregan al documento de dos maneras: inline o en un archivo externo. Inicialmente se ponen los scripts (JavaScript, CSS) de forma inline, las páginas cargan rápido debido a que no realiza la petición al servidor de dichos archivos. Pero en este caso el contenido del documento HTML se hace más grande, y cada vez que pida este documento, será del mismo tamaño y tardará en llegar al servidor como la primera vez que lo solicite. Mientras que poniendo los scripts (JavaScript, CSS) en archivos externos el contenido de la página disminuirá, pero se realizará la petición al servidor por parte del navegador para descargar dichos archivos. La primera vez que se solicite la página se demorara un instante de tiempo, pero en las demás solicitudes de la página se observará que se ha disminuido el tiempo de respuesta para mostrar dicha página. Esto es posible, ya que se ha aplicado la 2 regla, donde se cachean en el navegador los archivos CSS y JavaScript. De esta manera el contenido de la página disminuye y se muestra más rápido en la pantalla de los usuarios.

3.6.7 Cachear las peticiones AJAX

AJAX es una técnica que permite al navegador realizar peticiones asincrónicas al servidor. Unas de los beneficios que aporta AJAX es la de acelerar las aplicaciones Web. Pero si no se aprovechan al máximo las ventajas de AJAX, no se alcanzarán buenos rendimientos en las aplicaciones Web. El factor para lograr esto son los dos tipos de peticiones que se puede realizar con AJAX: peticiones pasivas y activas.

Las *peticiones pasivas* son aquellas peticiones que siempre se tienen de antemano una parte de las cosas que el usuario va a solicitar. Una vez que el usuario solicite una de estas cosas el navegador no tendrá que pedírselo al servidor debido a que lo tiene descargado en el cliente. Si el usuario solicita algo que no está descargado, el navegador lo pide al servidor para así mostrárselo al usuario. Ejemplo de este tipo de petición es lo que realiza YAHOO. Cuando un usuario de YAHOO solicita su bandeja de entrada, automáticamente se descarga en el navegador los tres primeros correos que se encuentran en la bandeja de entrada. Cuando el cliente selecciona alguno de esos correos, el navegador no tiene que descargarlo desde el servidor, sino que lo obtiene de los datos que tiene guardado en el cliente.

Las *peticiones activas* son aquellas peticiones que se realizan normalmente. Son las peticiones que necesitan datos del servidor cuando se realiza cualquier evento en el navegador que necesite de esos datos. Algo interesante de este tipo de peticiones es que también se pueden convertir en pasivas.

Al reducir el número de peticiones AJAX al servidor, se reduce la cantidad de peticiones por parte del navegador al servidor y así se reduce el tiempo total de respuesta de las páginas.

Conclusiones

En la industria del software de la actualidad, la mayoría de los productos que se elaboran, son aplicaciones orientadas a la Web. La Web en el mundo de hoy se ha convertido en unos de los servicios más usados y consumidos por los usuarios que utilizan internet para compartir y obtener información de toda naturaleza. Estos servicios que brinda la Web a través se ofertan de diversas maneras; como chat, correo electrónico, foros, multimedia, sitios Web, servicios Web, aplicaciones Web entre otros. Cada una de estas maneras de ofertan servicios a través de la Web son software realizados para poder brindarles dichos servicios a los usuarios de internet. Estos software no solo tienen la misión de ofertan estos servicios a los usuarios de internet, sino que también pretenden brindar dichos servicios de una manera eficiente, rápida y amigable para los usuarios de internet, en especial a los de la Web.

En el desarrollo de las aplicaciones orientadas a la Web se busca optimizar dichas aplicaciones, para que otorguen sus servicios a sus usuarios de manera rápida, eficiente, amigable, operativa y que se sean fáciles de mantener. También se busca que su desarrollo se lo más rápido posible y factible para los desarrolladores.

El presente trabajo se centra en hacer una propuesta de arquitectura para mejorar el proceso de desarrollo y la optimización de la capa de presentación del nuevo sistema SIIPOL. Esta propuesta se ha basado en el análisis y la investigación sobre la arquitectura del software y las nuevas tecnologías y prácticas para el desarrollo de aplicaciones empresariales orientadas a la Web enmarcadas en la capa de presentación. También se ha basado en el estudio y análisis de los mejores mecanismos para lograr aplicaciones Web eficientes, óptimas y que brinden mejores servicios.Como resultado se propone una arquitectura para la presentación que hace posible satisfacer todas las necesidades de los desarrolladores de la capa de presentación que desarrollan el nuevo sistema SIIPOL. Esto se logra mediante mecanismos de comunicación con capas adyacentes, de codificación, de control y organización del desarrollo, realización de componentes personalizados y la optimización del frontend del nuevo sistema SIIPOL.

Todo esto conlleva a obtener un sistema más rápido y eficiente al reducir los tiempos de respuesta de las peticiones de los servicios que brinda el SIIPOL a sus clientes. También posibilita obtener un producto robusto y fácil de mantener. Además, permite disminuir el tiempo de desarrollo de dicho sistema, así como disminuir el esfuerzo de los desarrolladores de la capa de presentación.

Por lo que podemos concluir que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente.

Recomendaciones

Como se ha observado los objetivos trazados para este trabajo han sido logrados, sin embargo, la propuesta es sólo la segunda fase de un proyecto que puede ser mucho más ambicioso. Por lo que hacemos las siguientes recomendaciones:

- Profundizar en el estudio de las tecnologías y prácticas mencionadas en la solución.

- Crear un equipo de desarrollo para que se prepare en las tecnologías y prácticas mencionadas, con vista a que puedan ser usadas en una aplicación futura.

- Que se utilicen en los proyectos de la universidad que son orientados a la Web (sitios Web ó aplicaciones Web), las prácticas que están relacionadas con la optimización de frontend de las aplicaciones Web.

Referencias Bibliográficas

1. HTML con Clase. [En línea] 15 de Febrero de 2008. http://html.conclase.net/articulos/historia.

2. INEI. [En línea] 17 de Febrero de 2008. http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5038/indice.HTM.

3. WEBNOVA. [En línea] 19 de Febrero de 2008. http://www.webnova.com.ar/diseno-web-argentina/aplicaciones-web-argentina.php.

4. **S, Christian Van Der Henst.** MOESTROSDELWEB. [En línea] 23 de Febrero de 2008. http://www.maestrosdelweb.com/editorial/web2/.

5. LIBROSWEB. [En línea] 21 de Febrero de 2008. http://www.librosweb.es/ajax.

6. JAVAHISPANO. [En línea] 26 de febrero de 2008. http://www.javahispano.org/contenidos/archivo/55/templates1.pdf.

7. JORDIZAN. [En línea] 02 de Marzo de 2008. http://jordisan.net/blog/2006/que-es-un-framework/.

8. MICROSOFT. [En línea] 03 de Marzo de 2008. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#10.

9. LIBROSWEB. [En línea] 07 de Febrero de 2008. www.librosweb.es/javascript/.

10. LIBROSWEB. [En línea] 09 de Marzo de 2008. http://www.librosweb.es/css/.

11. W3C. [En línea] 10 de Marzo de 2008. http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo(w3c).

12. **Stephenson, Sam.** PROTOTYPEJS. [En línea] 11 de Marzo de 2008. http://www.prototypejs.org.

13. PROACTIVA. [En línea] 13 de Marzo de 2008. http://www.proactiva-calidad.com/java/intro_applet/que_es_un_applet.html.

14. **Artaza, Dr. Diego Lz. de Ipiña Gz. de.** PAGINASPERSONALES. [En línea] 15 de Marzo de 2008. http://paginaspersonales.deusto.es/dipina/cursos/J2EECesine.zip.

15. **Merelo, J.J.** GENUARA. [En línea] 16 de Marzo de 2008. http://geneura.ugr.es/~jmerelo/JSP/.

16. **Microsystems.** DESARROLLOWEB. [En línea] 18 de Marzo de 2008. http://www.desarrolloweb.com/articulos/831.php.

17. SC.EHU.ES. [En línea] 19 de Marzo de 2008. http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/javaBeans/fundamento.htm#Definición%20de%20JavaBean.

18. **Albiol, Francesc Rosés.** FROSES. [En línea] 20 de Marzo de 2008. http://www.froses.com/Assets/Files/Articles/Hibernate_Introduccion_es.zip.

19. *Hibernate Refence.* 2008.

20. *Spring Reference.* 2008.

21. **Piñeiro, Juan Medín.** CSI. [En línea] 21 de Marzo de 2008. http://www.csi.map.es/csi/tecnimap/tecnimap_2006/01T_PDF/hacia%20una%20arquitectura.pdf.

22. desarrolloweb. [En línea] 10 de Abril de 2008. http://www.desarrolloweb.com/articulos/2380.php.

23. *JavaServer faces in Action.* 2007.

24. *Richfaces Reference.* 2008.

25. DANILAT. [En línea] 28 de Marzo de 2008. http://www.danilat.com/weblog/2007/07/06/empezando-con-facelets/.

26. *Introducción a XHTML.* 2008.

Bibliografía

Albiol, Francesc Rosés. FROSES. [En línea] 20 de Marzo de 2008. http://www.froses.com/Assets/Files/Articles/Hibernate_Introduccion_es.zip.

Artaza, Dr. Diego Lz. de Ipiña Gz. de. PAGINASPERSONALES. [En línea] 15 de Marzo de 2008. http://paginaspersonales.deusto.es/dipina/cursos/J2EECesine.zip.

DANILAT. [En línea] 28 de Marzo de 2008. http://www.danilat.com/weblog/2007/07/06/empezando-con-facelets/.

desarrolloweb. [En línea] 10 de Abril de 2008. http://www.desarrolloweb.com/articulos/2380.php.

Hibernate Refence. 2008.

HTML con Clase. [En línea] 15 de Febrero de 2008. http://html.conclase.net/articulos/historia.

INEI. [En línea] 17 de Febrero de 2008. http://www.inei.gob.pe/biblioineipub/bancopub/inf/lib5038/indice.HTM.

Introducción a XHTML. 2008.

JAVAHISPANO. [En línea] 26 de febrero de 2008. http://www.javahispano.org/contenidos/archivo/55/templates1.pdf.

JavaServer faces in Action. 2007.

JORDIZAN. [En línea] 02 de Marzo de 2008. http://jordisan.net/blog/2006/que-es-un-framework/.

LIBROSWEB. [En línea] 07 de Febrero de 2008. www.librosweb.es/javascript/.

LIBROSWEB. [En línea] 09 de Marzo de 2008. http://www.librosweb.es/css/.

LIBROSWEB. [En línea] 21 de Febrero de 2008. http://www.librosweb.es/ajax.

Merelo, J.J. GENUARA. [En línea] 16 de Marzo de 2008. http://geneura.ugr.es/~jmerelo/JSP/.

MICROSOFT. [En línea] 03 de Marzo de 2008. http://www.microsoft.com/spanish/msdn/arquitectura/roadmap_arq/style.asp#10.

Microsystems. DESARROLLOWEB. [En línea] 18 de Marzo de 2008. http://www.desarrolloweb.com/articulos/831.php.

Piñeiro, Juan Medín. CSI. [En línea] 21 de Marzo de 2008. http://www.csi.map.es/csi/tecnimap/tecnimap_2006/01T_PDF/hacia%20una%20arquitectura.pdf.

PROACTIVA. [En línea] 13 de Marzo de 2008. http://www.proactiva-calidad.com/java/intro_applet/que_es_un_applet.html.

Richfaces Reference. 2008.

S, Christian Van Der Henst. MOESTROSDELWEB. [En línea] 23 de Febrero de 2008. http://www.maestrosdelweb.com/editorial/web2/.

SC.EHU.ES. [En línea] 19 de Marzo de 2008. http://www.sc.ehu.es/sbweb/fisica/cursoJava/applets/javaBeans/fundamento.htm#Definición%20de%20JavaBean.

Spring Reference. 2008.

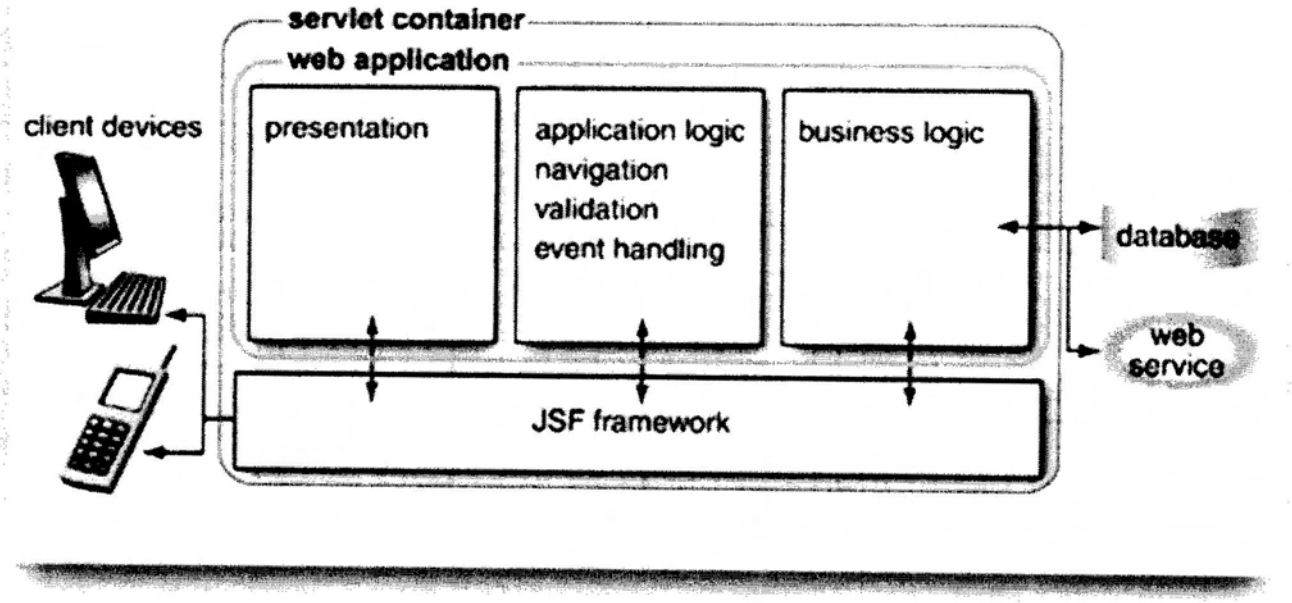
Stephenson, Sam. PROTOTYPEJS. [En línea] 11 de Marzo de 2008. http://www.prototypejs.org.

W3C. [En línea] 10 de Marzo de 2008. http://www.w3c.es/divulgacion/guiasbreves/HojasEstilo(w3c).

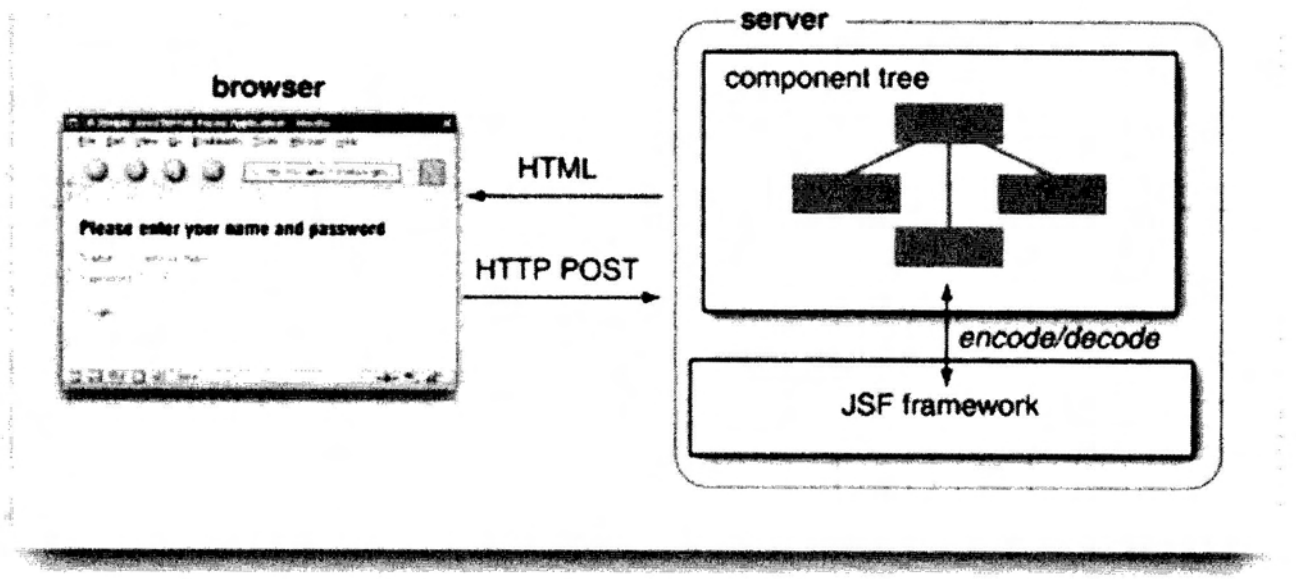
WEBNOVA. [En línea] 19 de Febrero de 2008. http://www.webnova.com.ar/diseno-web-argentina/aplicaciones-web-argentina.php.

Anexos

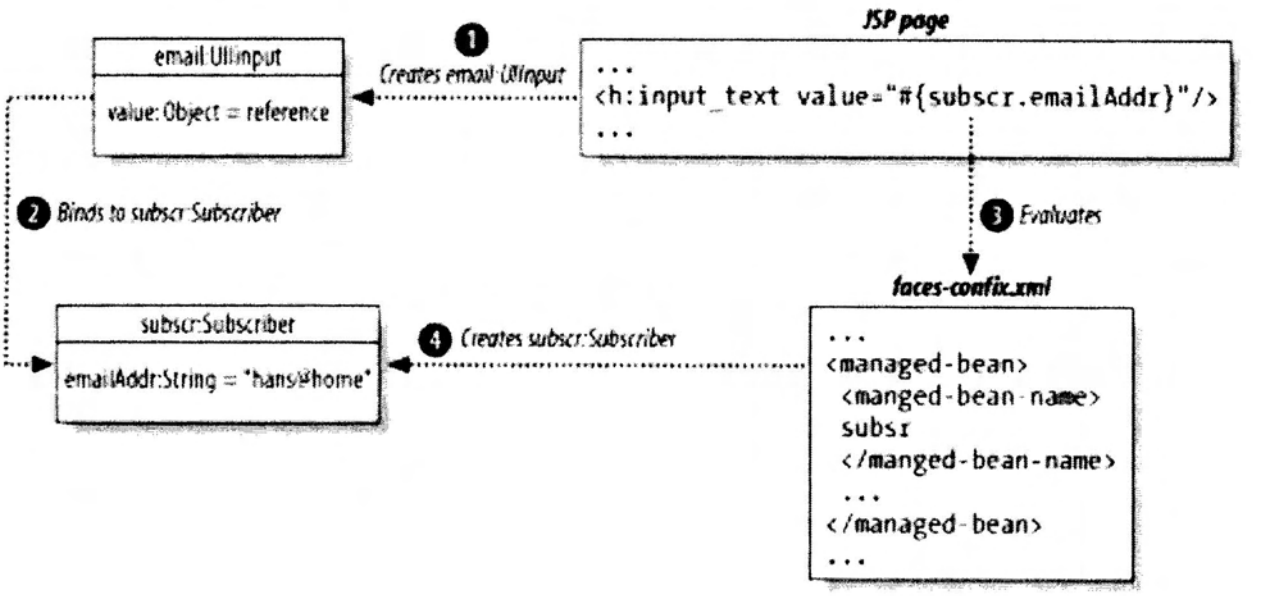
Anexo 1 Arquitectura JavaServer Faces



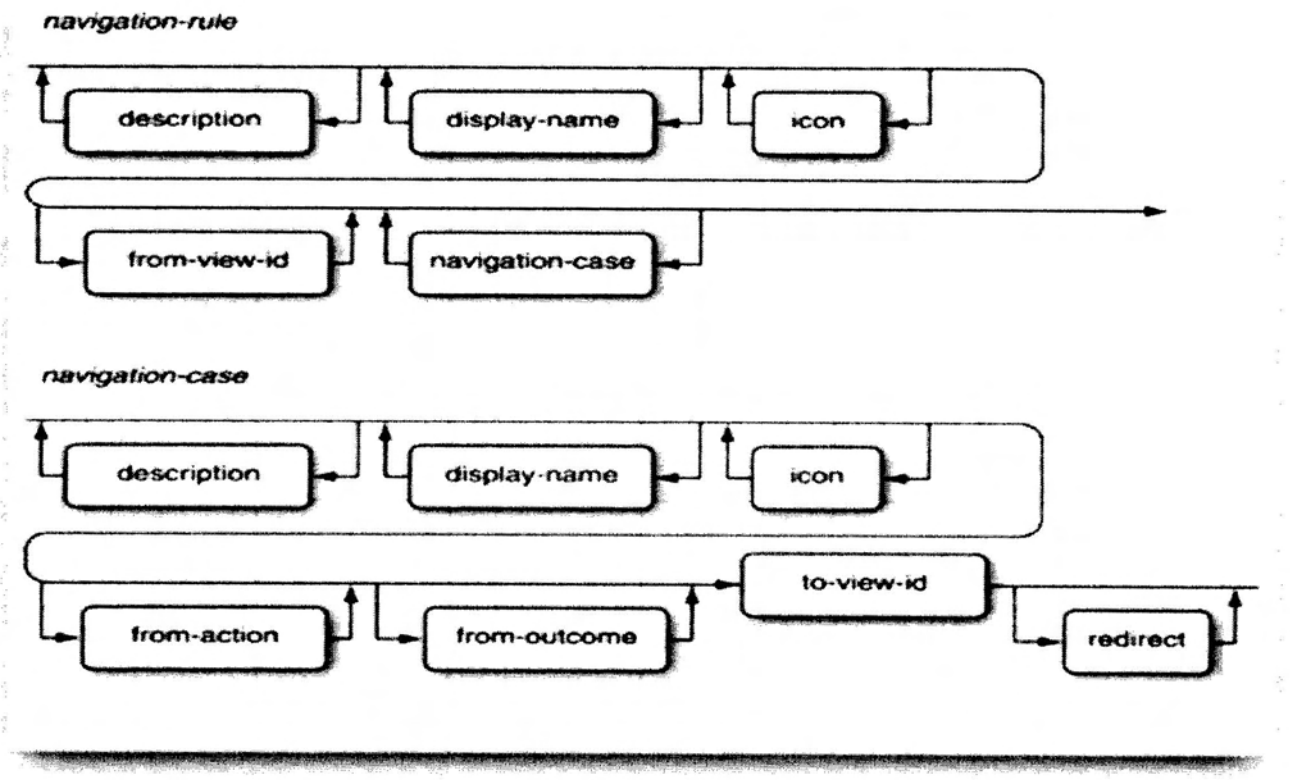
Anexo 4 Codificación y descodificación de las páginas JSF



Anexo 4 Creación de Objetos cuando se procesan las páginas JSP



Anexo 5 Elementos del diagrama de navegación



Anexo 6 Capacidad de AJAX con Rich Faces

```
<f:view>
   <h:form>
      <h:panelGrid columns="2">
         <h:outputText value="Type the Text:" />
         <h:inputText value="#{bean.text}">
            <a4j:support event="onkeyup" reRender="repeater" />
         </h:inputText>
         <h:outputText value="Text in the AJAX Response:" />
         <h:outputText id="repeater" value="#{bean.text}" />
      </h:panelGrid>
   </h:form>
</f:view>
```

Fires AJAX request on this event

a4j:support adds AJAX capability to existing components

Points to component(s) to be re-rendered

Anexo 7 Composición de componentes JSF inputtext

```
 9<ui:composition>
10
11    <h:panelGroup>
12        <h:outputLabel id="#{fieldName}Label" value="#{label}"
13            for="#{fieldName}" /> <br />
14
15        <h:inputText id="#{fieldName}" value="#{entity[fieldName]}"
16            required="#{required}" style="#{style}">
17            <ui:insert />
18        </h:inputText>
19    </h:panelGroup>
20
21</ui:composition>
```

Anexo 8 Composición de componentes JSF selectonemenu

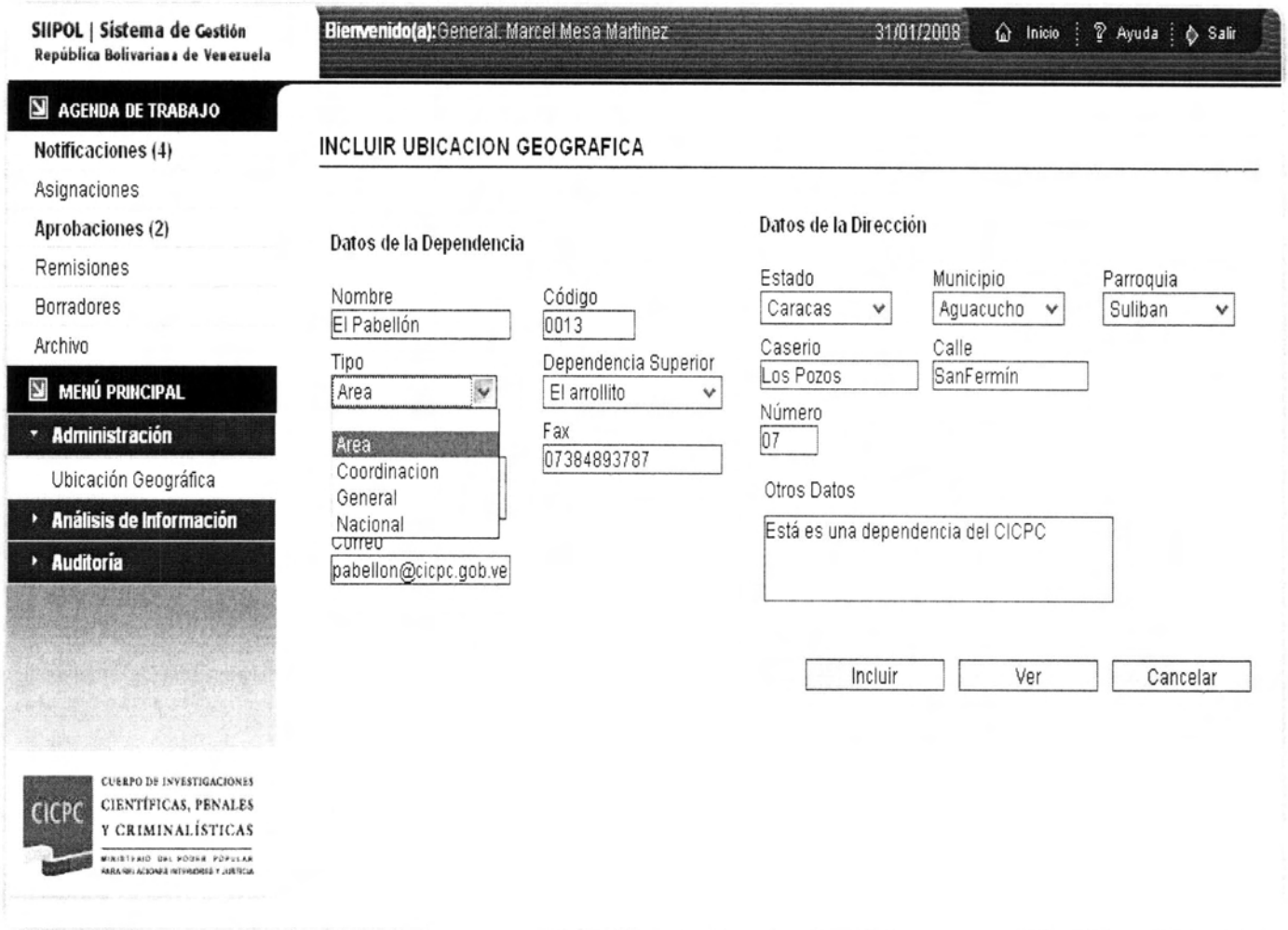
```
 9<ui:composition>
10
11    <h:panelGroup>
12     <h:outputLabel id="#{fieldName}Label" value="#{label}"
13        for="#{fieldName}" /> <br />
14
15     <h:selectOneMenu id="#{fieldName}" value="#{entity[fieldName]}"
16        required="#{required}" style="#{style}">
17        <ui:insert />
18     </h:selectOneMenu>
19     </h:panelGroup>
20
21</ui:composition>
```

Anexo 9 Composición de componentes JSF inputtextarea

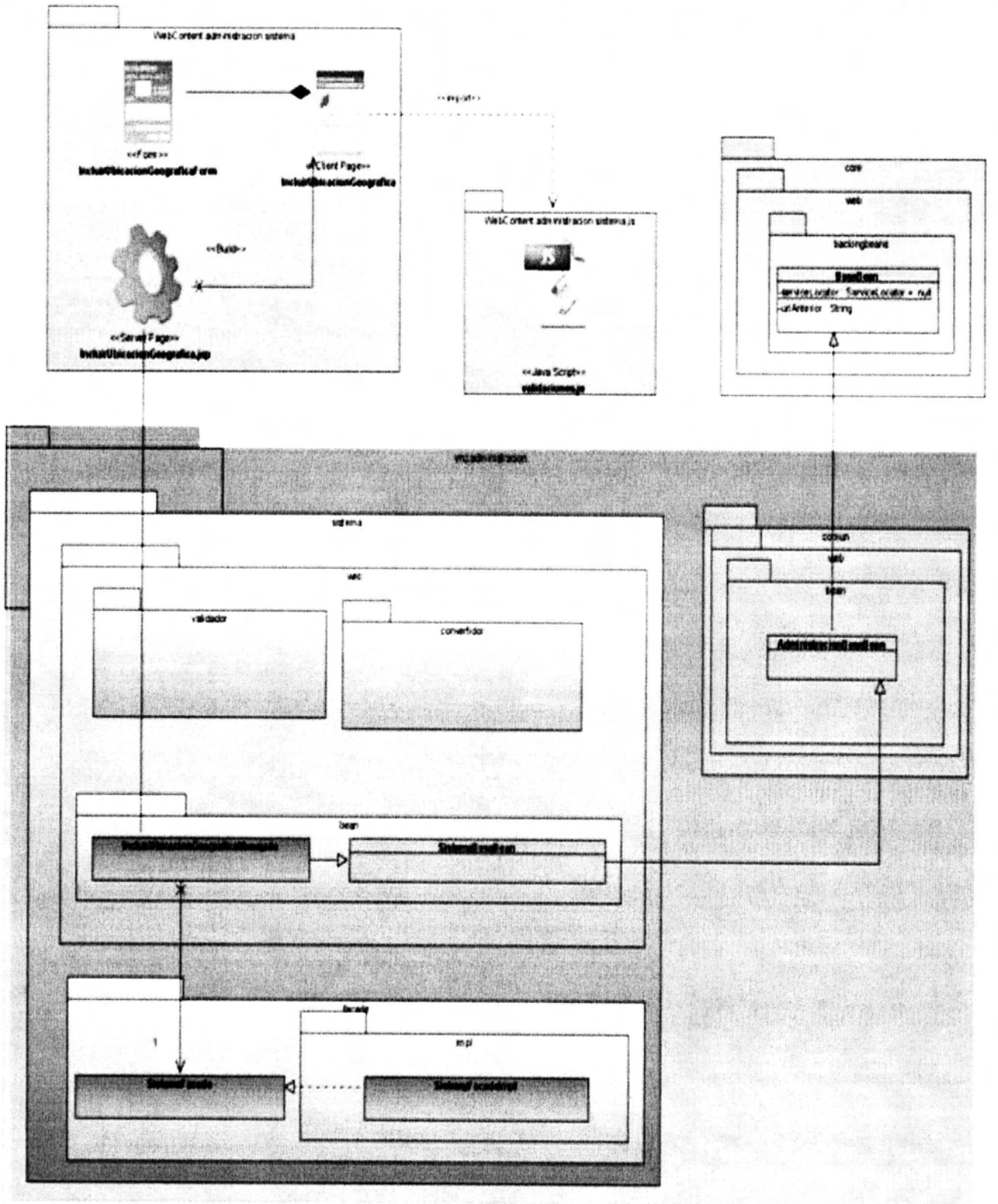
```
 9<ui:composition>
10
11    <h:panelGroup>
12        <h:outputLabel id="#{fieldName}Label" value="#{label}"
13            for="#{fieldName}" /> <br />
14
15        <h:inputTextarea id="#{fieldName}" value="#{entity[fieldName]}"
16            required="#{required}" style="#{style}" styleClass="marg_arriba">
17            <ui:insert />
18        </h:inputTextarea>
19    </h:panelGroup>
20</ui:composition>
```

Anexo 10 Portada del sistema SIIPOL

Anexo 11 Incluir Ubicación Geográfica



Anexo 12 Diagrama de Clase del escenario del Caso de Uso Incluir Ubicación Geográfica



Glosario

AJAX (Asincrónico JavaScript y XML): Técnica de programación que posibilita a las aplicaciones web realizar peticiones asincrónicas al servidor.

APPLET: Componente de una aplicación que se ejecuta en el contexto de otro programa (navegador Web).

CGI (Interfaz Común de Pasarela): Protocolo o interfaz de intercambio de información que se realiza entre el navegador del usuario y un servidor WWW.

CICPC (Cuerpo de Investigaciones Científicas Penales y Criminalística): Institución que se encarga de las investigaciones criminalísticas en la República Bolivariana de Venezuela.

CSS (Hojas de Estilos en cascada): Lenguaje utilizado para crear los estilos de los elementos de los documentos electrónicos.

DOM (Modelo de Objetos del Documento): Es una interface que contiene todos los elementos de la página en un árbol de objetos.

Frameworks (Marco de trabajo): Unión de varios componentes reutilizables que facilitan el desarrollo de software informático.

HTML (Lenguaje de Marcado de Hipertextos): Lenguaje utilizado para crear documentos electrónicos y transmitirlos a través de internet.

HTTP (Protocolo de Transferencia de Hipertextos): Conjunto de reglas para intercambiar archivos (texto, gráficos, imágenes, sonido, video y otros archivos multimedia) en la WWW.

IDE (Ambiente de Desarrollo Integrado): Es como se le llama al ambiente que proporciona al usuario una determinada herramienta de desarrollo.

Java: Lenguaje de programación que sigue el paradigma de Orientado a Objeto.

JavaScript: Lenguaje script utilizado para interactuar con el navegador en la aplicaciones Web.

JSON (Notación de Objetos de JavaScript): Formato ligero de intercambio de datos.

Open Source: Es una tendencia internacional del desarrollo de software que profesa la distribución del código junto a las aplicaciones. Se rigen por licencias tales como GNU/GPL.

Prototype: Librería echa en JavaScript que brinda gran variedad de funcionalidades para acelerar el desarrollo de aplicaciones Web.

RSS (Really Simple Syndication)**:** Sencillo formato de datos que es utilizado para re difundir contenidos a suscriptores de un sitio Web.

Servicios Web: Aplicación simple que realiza un cometido y que puede formar parte de otros servicios para formar un servicio más completo.

Tecnología: Es el conjunto de saberes que permiten fabricar objetos y modificar el medio ambiente, incluyendo plantas y animales, para satisfacer las necesidades y los deseos de nuestra especie.

WWW (World Wide Web)**:** Servicio de internet que posibilita la transmisión de documentos electrónicos.

XHTML (Lenguaje de Marcado de Hipertextos Extensibles:) Lenguaje utilizado para crear documentos electrónicos para transmitirlos a través de internet.

XML (Lenguaje de Marcas Extensibles)**:** Es un lenguaje utilizado para crear otros lenguajes y para crear documentos electrónicos para trasmitirlos a través de internet.

XMLHTTPRequest (Lenguaje de Marcado Extensible/ Protocolo de Transferencia de Hipertextos)**:** Interfaz empleada para realizar peticiones HTTP y HTTPS a servidores Web.