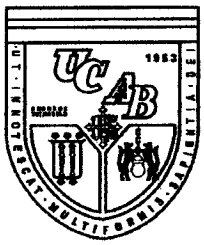

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

PLATAFORMA GENÉRICA CON TECNOLOGÍA WEB PARA EL SOPORTE AUTOMATIZADO DE LOS PROCESOS INTERNOS DE COMPAÑÍAS ASEGURADORAS.

TRABAJO ESPECIAL DE GRADO

presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

como parte de los requisitos para optar al título de

INGENIERO EN INFORMÁTICA

REALIZADO POR **Daniel Bohórquez**
Alejandro Miñarro

PROFESOR GUIA **Lic. William Torrealba**

FECHA **Mayo, 2006**

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SINÓPSIS

El propósito fundamental del presente Trabajo Especial de Grado consistió en desarrollar una "**Plataforma genérica basada en tecnología Web para el soporte automatizado de los procesos internos de compañías aseguradoras**", cumpliendo objetivos como la investigación de la información teórica necesaria y obtención de los conocimientos prácticos sobre los procedimientos actualmente ejecutados en empresas del ramo de manera de poder definir un modelo de datos estándar, capaz de soportar información básica para cualquier compañía del área de seguros y poder refinar procesos como la declaración de siniestros de vehículos y control de órdenes de reparación de un modo automatizado, y así crear una aplicación cuyas funcionalidades permitan apoyar la gestión y administración del negocio.

Los motivos que impulsaron el desarrollo de tal plataforma surgen ante la necesidad de automatizar algunos de los procedimientos más críticos de empresas aseguradoras, tales como el análisis y la declaración de siniestros de vehículos, y así agilizar estos procesos y lograr dar al cliente una respuesta más inmediata.

Se decidió usar la metodología de desarrollo conocida como RUP o Proceso Unificado de Rational (Rational Unified Process, en inglés) pues al ser una metodología iterativa, evolutiva e incremental se pueden obtener productos rápidamente que comprueben el adecuado rumbo del proyecto.

Así pues se obtuvieron resultados como: el conocimiento adquirido, base para la investigación y el desarrollo; la estandarización de procesos y datos, primordial en la búsqueda de un sistema genérico; la estructuración modular del sistema, esencial para lograr la escalabilidad de la aplicación; y lo más importante, una plataforma totalmente funcional capaz de ser adaptada a cualquier empresa aseguradora.

INTRODUCCIÓN

Quizá la razón más importante para el uso de tecnología en la automatización de procesos de grandes compañías, además del alto índice de competitividad, es la intención de ofrecer al cliente respuestas más inmediatas a asuntos concernientes con servicios solicitados. Es por ello que la implementación de una plataforma Web donde el cliente en la comodidad de su hogar, a través de un simple explorador Web, pueda acceder al sistema de la aseguradora, y que la misma pueda establecer comunicación, apoyándose en Redes Inalámbricas, con dispositivos móviles. Esto permitiría optimizar en gran medida los procesos más críticos de empresas de seguros, desde los procedimientos más simples como el registro de un usuario nuevo, pasando por la contratación de una póliza, la declaración de un siniestro e inclusive la revisión del vehículo siniestrado. Todo gracias a la tecnología.

Los capítulos que siguen a continuación pretenden plasmar el proceso de desarrollo de la plataforma planteada en el presente Trabajo de Grado, basados en el siguiente esquema: Capítulo 1: Planteamiento del Problema, se detallan las razones que impulsaron la realización del proyecto; Capítulo 2: Marco Referencial, comprende los conocimientos teóricos básicos necesarios para el correcto entendimiento del contenido; Capítulo 3: Marco Metodológico, este hace referencia a la metodología implementada, las actividades que la conforman y los posibles productos a obtener; Capítulo 4: Desarrollo, donde se detallan cada una de las iteraciones, apoyadas en las fases planteadas en la metodología, que fueron necesarias para completar la aplicación así como los productos obtenidos en cada una de ellas; Capítulo 5: Resultados, aquí se detallan los resultados obtenidos luego de completar la etapa de desarrollo; y por último se presentan las conclusiones, recomendaciones y bibliografía consultada, así como la información complementaria resumida en el ejemplar de apéndices.

1. PLANTEAMIENTO DEL PROBLEMA

Actualmente en Venezuela las compañías aseguradoras cubren un gran número de pólizas de seguro, lo que se traduce en la necesidad de atender altos volúmenes de siniestros relacionados a los mismos.

Debido a la gran cantidad de papeleo que se genera al procesar este tipo de siniestros, resulta muy complejo y tedioso tanto para las aseguradoras como para los asegurados dar inicio a estos procesos. Por otra parte el llevar un orden coherente de la documentación necesaria para hacer estos registros resulta complicado, pues no depende de solo una persona sino de un grupo de personas que se encargan de realizar cada uno de los pasos necesarios para generar la respuesta del siniestro al asegurado.

Todo lo mencionado anteriormente se refleja en una larga espera por una respuesta por parte de cliente que sufre el siniestro, lo que hace que la imagen de la compañía aseguradora sea de ineficiencia, afectando la calidad de servicio.

El manejo de los siniestros de vehículos es un proceso complejo realizado por un equipo interdisciplinario de especialistas quienes se encargan de analizar los datos y variables según cada caso.

En este proceso se evidencia una serie de planteamientos que sintetizan la problemática a tratar:

- Las compañías aseguradoras, deben ofrecer un servicio de calidad a sus asegurados brindando una respuesta eficiente y a tiempo, esto se reduce a ofrecer un servicio de calidad.
- Los siniestros de vehículos son manejados por diferentes departamentos dentro de las aseguradoras, que normalmente no tienen un medio de comunicación interna, por lo que los titulares de la

póliza deben encargarse ellos mismos del traslado del automóvil de un lugar a otro.

- Otro problema debido a la falta de comunicación entre los diferentes departamentos es la redundancia e inconsistencia de los datos de los titulares almacenados en los sistemas, trayendo como consecuencia que el titular deba facilitar información que, aparte de estar almacenada en la póliza, ya fue entregada con anterioridad.
- Un elemento importante en este proceso es la revisión de los daños del vehículo por parte del revisor o perito, quien debe realizar un informe de las averías incluyendo fotografías, descripción, etc. el cual debe ser enviado a otro departamento de la aseguradora y la mayoría de las veces, como no existe una comunicación directa entre los departamentos, el envío no se realiza de forma inmediata.
- Una vez concluido el proceso de análisis del siniestro dentro de la compañía aseguradora, se debe generar una orden de reparación que debe ser enviada al taller designado a realizar la reparación del vehículo afectado. Al igual que en los casos anteriores (como no existe comunicación directa entre los talleres y la aseguradora) el envió de dicha orden tarda un tiempo indefinido.

Por todo lo anteriormente planteado se hace perentoria la necesidad de diseñar y desarrollar una plataforma que ayude a soportar la información de los procesos de las compañías aseguradoras, estableciendo un medio de comunicación entre todos los departamentos involucrados y permitiendo la transmisión eficiente de los datos, necesarios para obtener tiempos de respuesta óptimos a fin de brindar un servio de calidad al asegurado.

1.1 OBJETIVO GENERAL

Desarrollar una plataforma genérica con tecnología Web para el soporte automatizado de los procesos internos de compañías aseguradoras.

1.2 OBJETIVOS ESPECÍFICOS

- Investigar acerca de los distintos tipos de pólizas de seguro.
- Investigar acerca de los distintos procesos internos a nivel de pólizas de las compañías aseguradoras.
- Investigar acerca de la información manejada en los distintos tipos de siniestros y así como los procesos relacionados con los mismos.
- Desarrollar un modelo de datos genérico para cualquier compañía aseguradora, que soporte toda la información recopilada.
- Definir una división funcional de los requerimientos a nivel de módulos.
- Desarrollar un modelo de configuración de módulos e interfaz para la aplicación.
- Desarrollar la arquitectura de la plataforma.
- Desarrollar la aplicación.

1.3 JUSTIFICACIÓN

Ofrecer un servicio de calidad a los clientes debe ser la meta de todas las compañías aseguradoras que operan en Venezuela, para que la imagen de la misma sea de prestigio.

El creciente número de siniestros de vehículos que deben ser procesados diariamente por las aseguradoras las han obligado a agilizar sus procesos de registro de este tipo de siniestros para acelerar sus tiempos de respuesta a los asegurados.

Sin embargo, una de las razones que dio impulso para la realización de este proyecto es la necesidad de que, aunque algunos de estos procesos

han sido mejorados, es evidente que los tiempos de respuesta al cliente continúan siendo extensos.

Para lograr una mejora sustancial y evidente, es necesario contar con mecanismos automatizados que permitan adaptar estos procesos de manera que se establezca una mejor comunicación entre los factores determinantes.

Desde el punto de vista tecnológico, la oportunidad de desarrollar una plataforma que englobe todos los procesos relacionados con los siniestros de vehículos utilizando tecnología Web como base para la elaboración del mismo, y la creación de un producto que no solo resulte novedoso y moderno, sino que también reduzca la complejidad del manejo de la información de un siniestro, ha sido también motivo de impulso para la realización de este Trabajo Especial de Grado.

1.4 LIMITACIONES

- Para la implementación del módulo de siniestros solo se contemplará la parte de siniestros de vehículos.
- En caso de aparecer algún módulo adicional a los reflejados en el alcance durante el levantamiento de información, el mismo no será implementado.

1.5 ALCANCE

- Definición de una plataforma genérica para el soporte de los procesos internos de las compañías aseguradoras.
- Implementación de una aplicación Web sobre la plataforma, que sea modular y configurable y que contendrá los siguientes módulos:

    - Módulo de Pólizas de Seguro: El módulo de pólizas será el encargado de manejar toda la información acerca del asegurado, ya sea una persona, un vehiculo, un inmueble u

otro tipo de bien. En este módulo se podrán realizar las operaciones de registro y consulta de los datos de las pólizas.

- Módulo de Siniestros: Este módulo manejará toda la información relacionada a siniestros de vehículos, desde los datos del asegurado hasta la información del siniestro.
- Módulo de Revisión: Este módulo es el encargado de todo el proceso de revisión del vehículo a la hora de un siniestro. Maneja todo el proceso de toma de fotografías del vehiculo, transmisión de las fotografías y revisión del vehículo.
- Módulo de talleres: El módulo de talleres será el encargado de la gestión de los talleres por parte de la compañía aseguradora. Este módulo se encargará de la administración de la información de los talleres asociados a la aseguradora y manejará todo el proceso de generación de las órdenes de reparación de los vehículos.
- Módulo de Configuración: Será el encargado de la administración de las configuraciones visuales del sistema. Manejará todo el proceso de habilitar y deshabilitar los módulos en el sistema. Permitirá que la interfaz de la aplicación sea configurable.
- Módulo de Reportes: Este módulo manejará la información del sistema a través de reportes informativos, reportes de estadística básica, copias de órdenes de siniestro y otros.

- Se desarrollará un modelo de datos genérico que soporte toda la información recopilada de las compañías aseguradoras.
- Toda la aplicación será desarrollada bajo el patrón MVC, por lo que se hará uso de Frameworks de desarrollo tipo Struts y Spring.
- Para la parte de configuración y manejo visual, se hará con herramientas como Ajax.

2. MARCO REFERENCIAL

Este capítulo contiene información que permite darle al lector los conocimientos básicos necesarios para comprender el contenido del presente Trabajo de Grado.

Aspectos tales como definiciones, conceptos y descripciones, llevarán al usuario a vislumbrar las ideas y objetivos planteados en el documento.

2.1 SEGUROS

La palabra Seguro proviene del latín Securus, que significa libre y exento de todo peligro, daño o riesgo. Es un contrato por el cual una persona natural o jurídica, se obliga a resarcir pérdidas o daños que ocurran en las cosas que corren un riesgo en mar o tierra [1].

El Contrato de Seguro es el documento o póliza por virtud del cual el asegurador se obliga frente al asegurado, mediante la percepción de una prima, a pagar una indemnización, dentro de los pactados, si se produce el evento previsto (siniestro).

2.1.1 Principios de los Seguros.

Son los fundamentos doctrinarios en que se basa la actividad aseguradora y son las normas que rigen las relaciones entre Asegurador y Asegurado. Son los siguientes [2]:

- Principio de Buena Fe: Llamado la "ubérrima fide" o la máxima buena fe que debe sustentar la validez del contrato de seguro, cuando las partes se rigen por actos de absoluta veracidad, a fin de evitar todo intento de dolo o mala intención.
- Principio de Indemnización: Trata de evitar un afán de lucro por parte del asegurado, en lugar de tener un seguro para garantizarle

solamente una protección que le libere de una pérdida o daño.

- Principio de Interés Asegurable: Hace que el seguro proteja lo económico de un bien hasta una suma máxima de pérdida, pero sin exceder el valor real total de dicho bien.
- Principio de Subrogación: Consecuencia del principio de indemnización, que faculta al asegurador (una vez que ha indemnizado una pérdida) a recuperar de terceras personas responsables, en caso de haberlas.
- Principio de Contribución: Según este principio, en caso de que una misma materia asegurada tuviera otros seguros, la pérdida debe ser compartida por los otros aseguradores en proporción a los capitales asegurados, sea cual sea el tipo de seguro.

2.1.2 Prima de Seguro.

La prima o precio del seguro es la contraprestación que ha de satisfacer el tomador del seguro a la entidad aseguradora, para que ésta asuma las eventuales consecuencias económicamente desfavorables del acaecimiento de los riesgos objeto de seguro. Para el asegurador, la prima representa el equivalente dinerario de la garantía de indemnización que otorga. Mediante la percepción de dicho precio, la entidad aseguradora puede constituir el fondo que le permite cumplir su finalidad, es decir, atender en la medida prevista al pago de los siniestros que se produzcan en su masa de asegurados, obteniendo unos diferenciales para asumir los gastos inherentes a toda actividad industrial, y simultáneamente los beneficios como retribución al capital invertido en la empresa [3].

2.1.3 Póliza de Seguro

Es un documento escrito que instrumenta el contrato de seguro, en el cual se reflejan las normas que de forma general, particular o especial, regulan las relaciones contractuales convenidas entre el asegurador y el

asegurado. La póliza puede ser nominativa, a la orden o al portador [3].

2.1.4 Contrato de Seguro

Es un documento o un contrato suscrito entre el asegurador y el tomador del seguro, en el que se recogen las condiciones por las que se va a regir el mismo. En ella figurarán los datos del representante de la compañía de seguros y del tomador [3].

Las características que tiene un contrato de Seguro son:

- Es un contrato aleatorio, ya que las partes ignoran en el momento de su conclusión si se verificará el siniestro.
- Es un contrato de duración.
- Es un contrato consensual del que deriva la obligación del asegurador de entregar un documento probatorio al tomador del seguro.
- Es un contrato de adhesión ya que el asegurador predispone las condiciones generales.

2.1.5 Siniestro

Según la Superintendencia de Seguros de la Republica Bolivariana de Venezuela, un siniestro es el acontecimiento futuro e incierto del cual depende la obligación de indemnizar por parte de la empresa de seguros. Si el siniestro ha continuado después de vencido el contrato, la empresa de seguros responde del valor de la indemnización en los términos del contrato. Pero si se inicia antes de la vigencia del contrato, y continúa después de que los riesgos hayan principiado a correr por cuenta de la empresa de seguros, ésta queda relevada de su obligación de indemnizar [4].

Según el Diccionario MAPFRE de seguros un siniestro es la manifestación concreta del riesgo asegurado que produce unos daños garantizados en la póliza hasta determinada cuantía. En otros términos, es

un acontecimiento que por originar unos daños concretos previstos en la póliza, motiva la aparición del principio indemnizatorio, obligando al asegurador a satisfacer total o parcialmente al asegurado o a sus beneficiarios, el pago del capital garantizado en el contrato [5].

2.1.6 Cláusulas o Condiciones

Son el acuerdo establecido en un convenio. Generalmente, en los contratos de seguro, las cláusulas vienen a modificar, aclarar o dejar sin efecto parte del contenido de sus condiciones generales o particulares.

Las partes que intervienen en un Contrato de Seguro son:

- La Aseguradora: Es la empresa que presta el servicio de aseguramiento, y que asume la obligación del pago de la indemnización cuando se produzca el evento asegurado.
- El Asegurado: Es la persona que en sí mismo, en sus bienes o intereses económicos, está expuesta al riesgo. En la práctica, la figura del asegurado se ve acompañada de otras manifestaciones personales que unas veces son coincidentes y otras, particularmente en algunas modalidades de seguro, gozan de independencia: el suscriptor de la póliza, llamado generalmente tomador del seguro o contratante, cuya peculiaridad radica en su responsabilidad del pago de la prima; el asegurado, cuyas circunstancias personales o de otro tipo originan o pueden motivar el pago de la indemnización; el beneficiario, cuya única vinculación al contrato de seguro es la de ser titular del derecho indemnizatorio.
- El Contratante: Es el cliente de la aseguradora, la persona que suscribe la póliza y que paga por el servicio (pago de la prima).
- El Beneficiario: La persona que recibirá el pago por parte de la compañía de seguros, es el titular de los derechos indemnizatorios.

Puede darse el caso de que dos o más de las últimas figuras, se encuentren en una sola persona [6].

2.2 FRAMEWORK

Los frameworks orientados a objetos son la piedra angular de la moderna ingeniería del *software*. El desarrollo del framework está ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente. Los frameworks son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.

Algunas de las características del framework no son mutables ni tampoco pueden ser alteradas fácilmente. Estos puntos inmutables constituyen el núcleo o *kernel* de un framework, también llamados como los puntos congelados del framework, a diferencia de los puntos calientes, los puntos congelados o inmutables son los pedacitos del código puestos en ejecución ya dentro del framework que llaman a uno o más puntos calientes proporcionados por el ejecutor. El núcleo o *Kernel* será la constante y presentará siempre la parte de cada instancia del framework.

La capacidad de reutilización del código y del diseño de frameworks orientados al objeto permite una productividad mayor y un tiempo de mercado breve en el desarrollo de aplicaciones, en comparación con el desarrollo tradicional de los sistemas de *software*. La configuración flexible de frameworks, permite la reutilización del núcleo *kernel* [7].

2.2.1 FrameWork Struts

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el *patrón de diseño* MVC en la plataforma *J2EE* (*Java* 2, Enterprise Edition). Originalmente fue creado por Craig R. McClanahan

quien se encuentra profundamente involucrado con los grupos de expertos quienes describen las especificaciones de *JSP* y *Servlets* y ha escrito una gran parte de la implementación de Tomcat 4.0. Actualmente el framework *Struts* se desarrolla como parte del proyecto. Jakarta de la Apache *Software* Foundation.

Struts permite reducir el tiempo de desarrollo, su carácter de *software libre* y su compatibilidad con todas las plataformas en donde *Java* Entreprise está disponible, lo convierte en una herramienta altamente disponible [8].

El uso de este ambiente permite la implementación de tecnología *Java* para el desarrollo de aplicaciones Web, por un lado archivos comunes *Java* que permiten definir clases en el servidor con lo cual se orienta el desarrollo a objetos, práctica usada por muchos programadores a nivel mundial. Por otro lado se utilizan archivos *JSP* (*Java*Server Pages) que permiten mostrar a través del explorador Web la interacción entre el usuario y los objetos del sistema. Al momento de su compilación, estos archivos son pre-procesados y convertidos en archivos *Java* que serán interpretados por el servidor cuando son llamados por el usuario. La figura 2.1 ilustra este proceso:

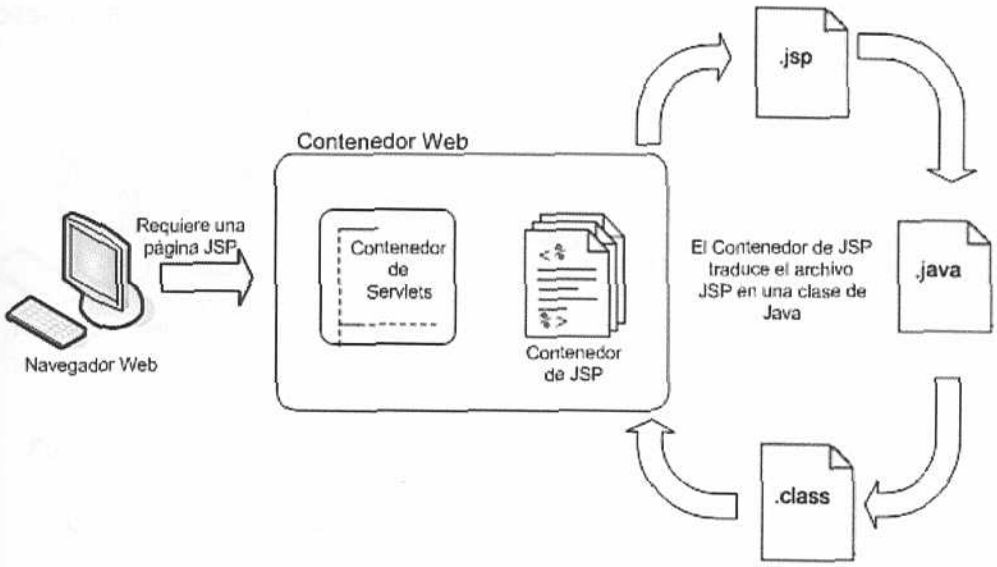

Figura 2.1: Proceso de compilación de un archivo JSP en Java
Fuente: http://www.triviasecurity.net/library/Programming/OReilly%20-%20Jakarta%20Struts.pdf

En general, el uso de tecnología *JSP* se ha convertido en una solución extremadamente popular en la construcción de aplicaciones Web utilizando la plataforma *Java*.

JSP ofrece algunas ventajas sobre las demás tecnologías disponibles:

- Es una especificación, no un producto.
- Las páginas son compiladas no interpretadas, lo cual sugiere mejor rendimiento y un procesamiento más eficiente.
- Soporta acceso completo al lenguaje Java.

Las tempranas especificaciones de *JSP* presentaron 2 acercamientos para la construcción de aplicaciones Web utilizando esta tecnología, que fueron descritos como arquitecturas Modelo 1 y Modelo 2, y que actualmente son ampliamente usadas en toda la red. Ambas arquitecturas difieren en varios puntos clave, siendo la mayor diferencia cómo y por cual componente es manejado el procesamiento de una petición. En la primera arquitectura la misma página *JSP* se encargar de manejar el procesamiento de una petición y también es responsable de mostrar la información al usuario [10]. Este proceso puede ser visto en la figura 2.2.

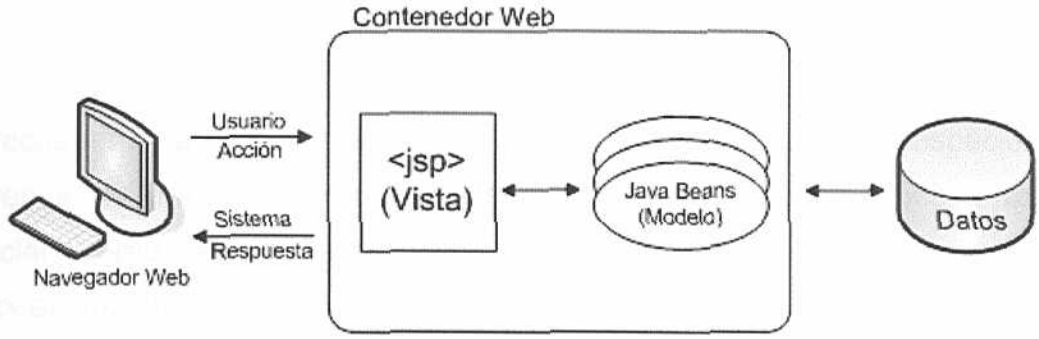

Figura 2.2: Diagrama del patrón de diseño MVC Modelo 1
Fuente: http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.HTML

Como puede ser apreciado en la figura 2.2, no existe ningún *Servlet* o controlador involucrado en el proceso. La petición del cliente es enviada directamente a la página *JSP*, el cual se puede comunicar con otros

JavaBeans, archivos *Java* u otros servicios, pero al final la página *JSP* selecciona la siguiente página a ser mostrada al cliente. La siguiente vista es determinada bien sea basada en el JSP seleccionado o en los parámetros de la petición del cliente.

En comparación directa con el acercamiento del Modelo 1, en la arquitectura del Modelo 2 la petición del cliente es primeramente interceptada por un controlador o *Servlet*, quien maneja el procesamiento inicial de la petición y también determina la próxima página *JSP* a mostrar. Este acercamiento es mostrado en la figura 2.3:

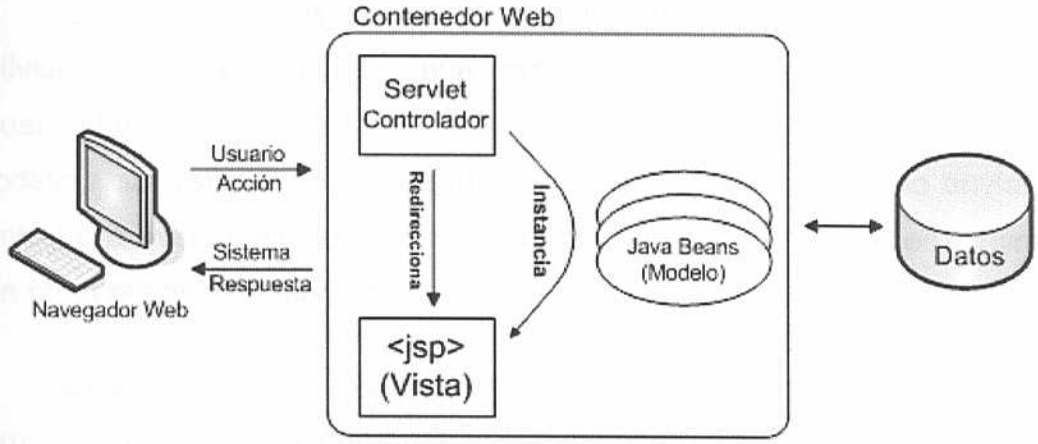

Figura 2.3: Diagrama del patrón de diseño MVC Modelo 2
Fuente: http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.HTML

Como puede ser apreciado, un cliente nunca envía una petición directamente a una página *JSP*. El controlador actúa como especie de director de tránsito, lo que permite al controlador realizar procesamiento inicial como autenticación y/o autorización, etc. Una vez que el procesamiento ha finalizado, el controlador direcciona la petición a la página *JSP* apropiada. Exactamente de qué manera cada página es determinada puede variar ampliamente en cada aplicación.

La principal diferencia entre ambos modelos es la presencia de un *Servlet* o controlador que provee un único punto de acceso y también permite

mayor reusabilidad y extensibilidad que el Modelo 1. Con la arquitectura del Modelo 2 existe también una clara separación de la lógica del negocio, presentación y procesamiento de peticiones, separación a la cual se hace referencia como patrón Modelo-Vista-Controlador (MVC). El uso de esta arquitectura también simplifica el mantenimiento de la aplicación y permite obtener aplicaciones significativamente más extensibles que las desarrolladas bajo la arquitectura del Modelo 1.

El componente Modelo comprende accesos a Bases de Datos o sistemas que funcionan independientemente de la aplicación Web.

Los componentes de control son los encargados de coordinar las actividades de la aplicación, que van desde la recepción de datos del usuario, las verificaciones de forma y la selección de un componente del modelo a ser llamado. Por su parte los componentes del modelo envían al control sus eventuales resultados y/o errores de manera de poder continuar con otros pasos de la aplicación.

Esta separación simplifica enormemente la escritura tanto de vistas como de componentes del modelo: Las páginas *JSP* no tienen que incluir manejo de errores, mientras que los elementos del control simplemente deciden sobre el paso siguiente a seguir [9].

Las Características del Framework son [10]:

- Configuración del control centralizada.
- Interrelaciones entre Acciones y página u otras acciones se especifican por tablas *XML* en lugar de codificarlas en los programas o páginas.
- Componentes de aplicación, que son el mecanismo para compartir información bidireccionalmente entre el usuario de la aplicación y las acciones del modelo.

- *Struts* contiene herramientas para validación de campos de plantillas. bajo varios esquemas que van desde validaciones locales en la página (en *JavaScript*) hasta las validaciones de fondo hechas a nivel de las acciones.
- *Struts* permite que el desarrollador se concentre en el diseño de aplicaciones complejas como una serie simple de componentes del Modelo y de la Vista intercomunicados por un Control centralizado. Diseñando de esta manera se debe obtener una aplicación más consistente y más fácil de mantener.

2.3 PATRÓN DE DESARROLLO MVC

Modelo Vista Controlador (MVC) es un patrón de diseño de *software* que separa el modelo de datos de una aplicación, la *interfaz de usuario*, y la lógica de control en tres componentes distintos de forma que las modificaciones al componente de la vista pueden ser hechas con un mínimo impacto en el componente del modelo de datos. Esto es útil ya que los modelos típicamente tienen cierto grado de estabilidad (dependiendo de la estabilidad del dominio del problema que esta siendo modelado), donde el código de la *interfaz de usuario* el cual usualmente sufre de frecuentes y a veces dramáticos cambios (dependiendo de problemas de *usabilidad*, la necesidad de soportar clases crecientes de usuarios o simplemente la necesidad de mantener la aplicación viéndose como nueva). Separar la vista del modelo hace que este sea más robusto, debido a que el desarrollador esta menos propenso a romperlo mientras trabaja de nuevo en la vista [10].

El patrón fue descrito por primera vez en 1979 por Trygve Reenskaug, quién trabajaba en Smalltalk en los laboratorios de investigación de la Xerox.

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control generalmente es el siguiente:

- El usuario interactúa con la interfaz de alguna manera (ej. presionando un botón, enlace).
- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario.
- El controlador accede al modelo, posiblemente actualizando los datos enviados por el usuario.
- El controlador delega a los objetos de la vista la tarea de desplegar la *interfaz de usuario.*
- La vista usa el modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo, aunque en algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista.
- La interfaz espera por nuevas interacciones de usuario para iniciar nuevamente el ciclo.

Las aplicaciones Web complejas continúan siendo más difíciles de diseñar que las aplicaciones tradicionales de escritorio, el patrón MVC se presenta como una solución para ayudar a disminuir dicha complejidad.

Las Caracteristicas del modelo son [10]:

Modelo: dependiendo en el tipo de arquitectura de la aplicación, la porción del modelo en el patrón de diseño puede variar y tener formas diferentes. En una aplicación donde la parte Web interactúa directamente con un almacén de datos, las clases del modelo pueden ser simplemente un grupo de objetos regulares. En aplicaciones empresariales más complejas donde la parte se comunica con servidores especializados, la porción del modelo puede corresponder a objetos empresariales.

Vista: en aplicaciones Web son consideradas como vistas las páginas *HTML*, usadas para servir contenido estático y aquellos *JSP*, *ASP*, *PHP*, que pueden ser usados para mostrar contenido estático y dinámico. La mayor

parte del contenido dinámico es generado en la parte Web, sin embargo algunas aplicaciones optan por el uso de lenguajes ejecutados del lado del cliente en la misma vista, como *JavaScript*, lo cual no influye o infringe el concepto original del patrón MVC.

Controlador: en una aplicación Web, cumple con las labores de interceptar las peticiones del cliente, traducir estas en una operación específica del negocio a realizar e invocar una operación específica o enviársela a otro controlador y por último ayuda a seleccionar y devuelve la próxima vista a ser mostrada al cliente.

2.4 ARQUITECTURAS DE DESARROLLO

Define el planteamiento tecnológico que se establece para organizar el buen desarrollo de la plataforma.

2.4.1 Modelo Cliente Servidor

El paradigma cliente/servidor es uno de los más extendidos dentro de los servicios a través de red. La idea básica y general que hay detrás de este modelo es que hay alguien que ofrece algo (el servidor) y alguien que quiere algo (el cliente) [14].

El cliente realiza peticiones al servidor, mientras que el servidor se dedica simplemente a responderle. De por sí, un servidor no hace nada; necesita que un cliente le demande algo [11].

En esta arquitectura cada uno de los usuarios, produce una demanda de información a cualquiera de las computadoras que la proporcionan, las cuales responden a la demanda del cliente que la produjo.

Los clientes y los servidores pueden estar conectados a una red local o una red amplia, como la que se puede implementar en una empresa o a una red mundial como lo es Internet.

Bajo este esquema cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

Estas solicitudes pueden ser peticiones de datos de una base de datos, de información contenida en archivos o los archivos en sí mismos o peticiones de imprimir datos en una impresora asociada. Una máquina cliente unida a un servidor puede ser a su vez servidor de otro cliente y el servidor puede ser un cliente de otro servidor en la red [11].

Las características principales del modelo son:

- El Cliente y el Servidor pueden actuar como una sola entidad y también pueden actuar como entidades separadas, realizando actividades o tareas independientes.
- Las funciones de Cliente y Servidor pueden estar en plataformas separadas, o en la misma plataforma.
- Un Servidor atiende a múltiples Clientes en forma concurrente.
- Cada plataforma puede ser escalable independientemente. 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 formatos de datos y de los protocolos.

- Un sistema de Servidores realiza múltiples funciones al mismo tiempo que presenta una imagen de un solo 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.

2.4.2 Modelo Web

Se podría describir una aplicación Web como un entorno del sistema que cumple con ciertas funcionalidades al igual que un programa común, pero cuya ejecución se realiza a través de un navegador o *browser* del lado del cliente, sin la necesidad de instalar un *software* intermedio, lo cual ofrece un alto índice de practicidad y simplicidad para el usuario y el desarrollador [15].

Sus inicios forman parte de la evolución de la Web. Originalmente se ofrecían al usuario contenidos estáticos, y su funcionalidad era meramente informativa. Sin embargo, y con el pasar de los años, dicha funcionalidad fue cambiando conforme iban cambiando las necesidades de los usuarios, como acceso remoto a aplicaciones, cambios constantes en los requerimientos, repetidas actualizaciones, etc. Una vez que aparecieron los lenguajes de codificación y ejecución interpretados por el Servidor Web (*PHP*, *ASP*, *JSP*, etc.) la Internet sufrió un cambio vertiginoso e importante, y es a partir de entonces cuando el desarrollo de estas aplicaciones usando estas tecnologías comenzó crecer de manera exponencial hasta hacer casi desparecer a los comunes modelos Cliente-Servidor.

Entre las ventajas del modelo se pueden mencionar [15]:

- Se considera como una de las razones más comunes para la migración a una Aplicación Web, la necesidad de tener acceso vía

Internet al *software* desde cualquier punto, bien sea remoto o localmente en la misma red. Esta funcionalidad le da al usuario la facilidad de disponer de la aplicación donde sea que éste se encuentre.

- Así mismo, los costos por razones de actualización del *software* resultan mucho menor que las aplicaciones Cliente-Servidor, pues el *software* como tal reside en un solo lugar, el servidor, y estando en esta ubicación los cambios y modificaciones son realizadas de manera única sin afectar al resto de la aplicación.
- La alta disponibilidad y simplicidad forman parte también de las ventajas de este tipo de aplicación.

Al igual que las aplicaciones Cliente-Servidor, las aplicaciones Web también necesitan de un lenguaje o código de desarrollo que las haga funcionar. La diferencia radical es que se hace necesario la existencia y selección de 2 tipos de estos, los que se ejecutan en el servidor en donde se sirve la aplicación Web, y otro que se ejecuta en el navegador o *browser* del cliente donde se muestra la información.

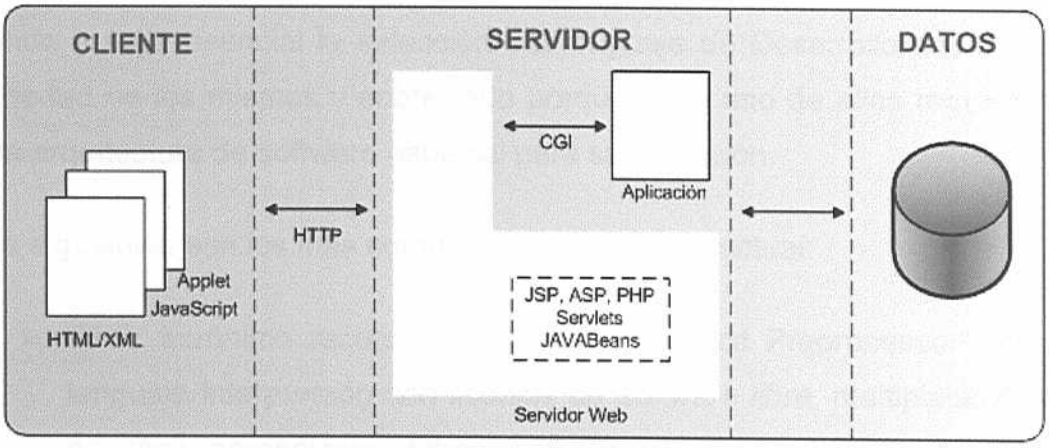

Figura 2.4: Diagrama de una aplicación Web
Fuente: http://www.triviasecurity.net/library/Programming/OReilly%20-%20Jakarta%20Struts.pdf

Con respecto a los lenguajes en el lado del cliente se tienen [16]:

- ***HTML***: es un lenguaje definido por un conjunto de reglas para estructurar y dar formato a un documento electrónico y texto, para presentarlos en forma de hipertexto, que es el formato estándar de las páginas Web. Utiliza etiquetas o marcas, que consisten en breves instrucciones de comienzo y final, mediante las cuales se determinan la forma en la que debe aparecer en el navegador el texto, así como también las imágenes y los demás elementos [17].
- ***JavaScript***: Posee una sintaxis semejante al Lenguaje *Java*. Se ejecuta en el navegador del cliente y actualmente resulta el más usado a nivel mundial. Posee una estructura bastante sencilla, dado su similitud con *Java*, y una gran capacidad de procesamiento.
- ***CSS***: Las Hojas de Estilo en Cascada, (*Cascade Style Sheets*, por sus siglas en inglés) son utilizadas para definir de manera formal y estructurada la presentación de un documento *HTML*. Permite tener un control centralizado de la presentación de un sitio Web completo con lo que se agiliza de forma considerable la actualización del mismo.

Por otra parte, en la parte de los lenguajes del lado del servidor es donde resulta esencial la selección del lenguaje de Desarrollo debido a la variedad de los mismos y sobre todo porque cada uno de ellos requiere de una arquitectura de software especial para su ejecución.

Los siguientes son los más comunes en el mercado actual:

- ***PHP***: acrónimo recursivo de "***PHP***: Hypertext Preprocessor", es un lenguaje interpretado con licencia de *software libre*, multiplataforma y de fácil aprendizaje. Ofrece muchas ventajas antes sus rivales competidores como lo son la capacidad de conexión con la mayoría de los manejadores de Base de Datos, además de ser una alternativa de fácil acceso por su carácter de *software libre* [18].

- ***ASP / ASP*.NET:** acrónimo de "**A**ctive **S**erver **P**ages", es un lenguaje creado por Microsoft, de ejecución en servidores IIS (Internet Information Services) de Windows. En el caso de *ASP*, ofrece funcionalidades básicas de programación, de igual facilidad de aprendizaje, incluso para una persona profana de la programación, por su similitud con el lenguaje Visual Basic. La nueva versión, *ASP*.NET, aumenta el nivel de complejidad basando su desarrollo en lenguajes como C#, también de Microsoft, lo cual proporciona mayores funcionalidades.
- ***JSP*:** acrónimo de "**J**ava **S**erver **P**ages", es un lenguaje desarrollado por Sun Microsystems, cuya estructura es una variante del lenguaje *Java*. La principal ventaja de *JSP* frente a otros lenguajes es que permite integrarse con clases *Java* (.class) dando la posibilidad de separar en niveles las aplicaciones Web, almacenando en clases *Java* las partes que consumen más recursos así como las que requieren más seguridad, y dejando la parte encargada de formatear el documento *HTML* en el archivo *JSP*. Es multiplataforma, siendo requisitos indispensables para su ejecución la existencia en el sistema del Servidor Apache y la última versión del paquete de desarrollo de *Java* (JDK), todos estos de licencia gratuita.

2.4.3 AJAX [12] y [13]

A diferencia de lo que muchos pueden pensar (un equipo de fútbol holandés o un detergente de triple acción), el significado de AJAX es un acrónimo de **Asynchronous *JavaScript* And *XML*** (es.wikipedia.org/wiki/AJAX) cuyo significado en español sería ***JavaScript* y *XML* de manera Asíncrona.** En realidad AJAX no es una tecnología, sino más bien un concepto o incluso la unión de varias tecnologías ya existentes que juntas pueden lograr cosas realmente impresionantes como lo son

GoogleMaps, Gmail, BackBase o algunas otras aplicaciones muy conocidas. Ajax incorpora:

- Presentación basada en estándares mediante la utilización de XHTML y *CSS*. Donde *XHTML* que por su acrónimo inglés es eXtensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a *HTML* como estándar para las páginas Web. *XHTML* es la versión *XML* de *HTML*, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas de *XML*. Y *CSS*.
- Intercambio y manipulación de datos utilizando XML y XSLT.
- Obtención de data asíncrona usando XMLHttpRequest.
- Y JavaScript como nexo de unión de todo ello.

Básicamente, la principal virtud de AJAX está en la potencia que se le puede extraer al trabajo asíncrono de peticiones al servidor. Normalmente se hace uso de un modelo de interacción sincrónica basada en clic-petición-presentación. Con AJAX la interacción pasa a ser asíncrona. Cada vez que se hace clic no necesariamente se establece una conexión con el servidor. Esto ofrece una gran ventaja ya que proporciona a los usuarios la posibilidad de realizar complicadas interacciones sin tener que esperar una respuesta del servidor, porque toda la información necesaria está descargada en el navegador.

El modelo clásico de aplicaciones Web (como anteriormente se mencionó) funciona de esta forma: La mayoría de las acciones del usuario en la interfaz disparan un requerimiento *HTTP* al servidor Web. El servidor efectúa un proceso (recopila información, procesa números, hablando con varios sistemas propietarios), y le devuelve una página *HTML* al cliente. Este es un modelo adaptado del uso original de la Web como un medio hipertextual, pero como es bien sabido, lo que hace a la Web buena para el

hipertexto, no la hace necesariamente buena para las aplicaciones de *software*.

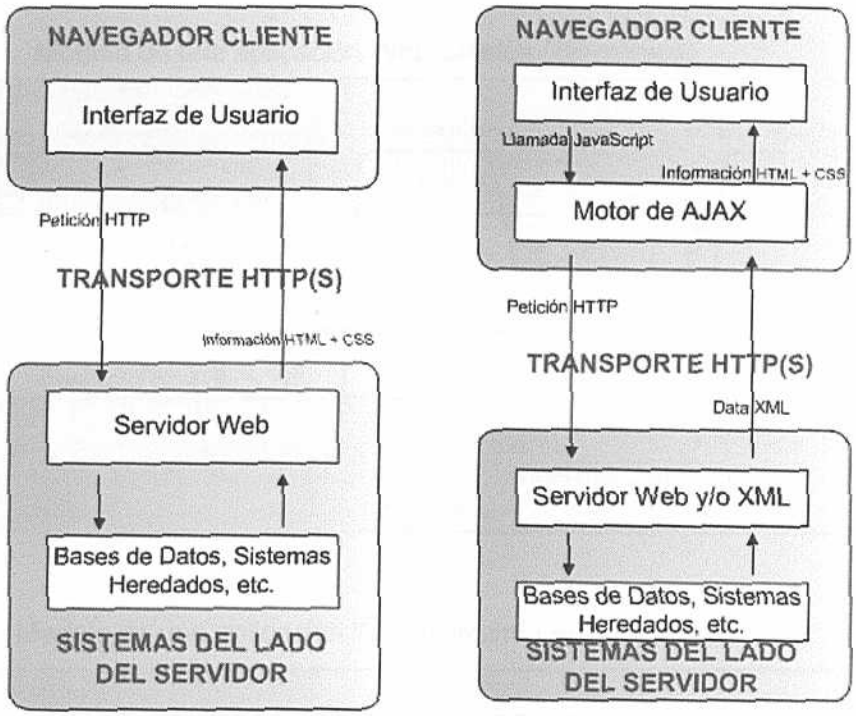

Figura 2.5: Modelos de una aplicación Web
Fuente: http://adaptivepath.com/publications/essays/archives/000385.PHP

Este acercamiento tiene mucho sentido a nivel técnico, pero no lo tiene para una gran experiencia de usuario. Mientras el servidor se encuentra realizando las tareas de procesamiento, el usuario no tiene otra opción más que esperar. Y en cada paso que éste realiza, espera aún más. Lo cual, desde el punto de vista lógico, carece de sentido pues una vez que la interfaz gráfica está cargada las interacciones del usuario no deberían detenerse cada vez que la aplicación necesita algo del servidor.

La utilización de la técnica de AJAX permite establecer un intermediario entre el usuario y el servidor, agregando una capa adicional a la aplicación y permitiendo que la interacción con el usuario sea de manera asíncrona (independientemente de la comunicación con el Servidor) ofreciendo una Comunicación Visual más fluida e inmediata en cuanto a

velocidad de respuesta se refiere dándole al usuario mayor comodidad y confianza pues en cada momento el usuario está enterado de lo que sucede en la aplicación.

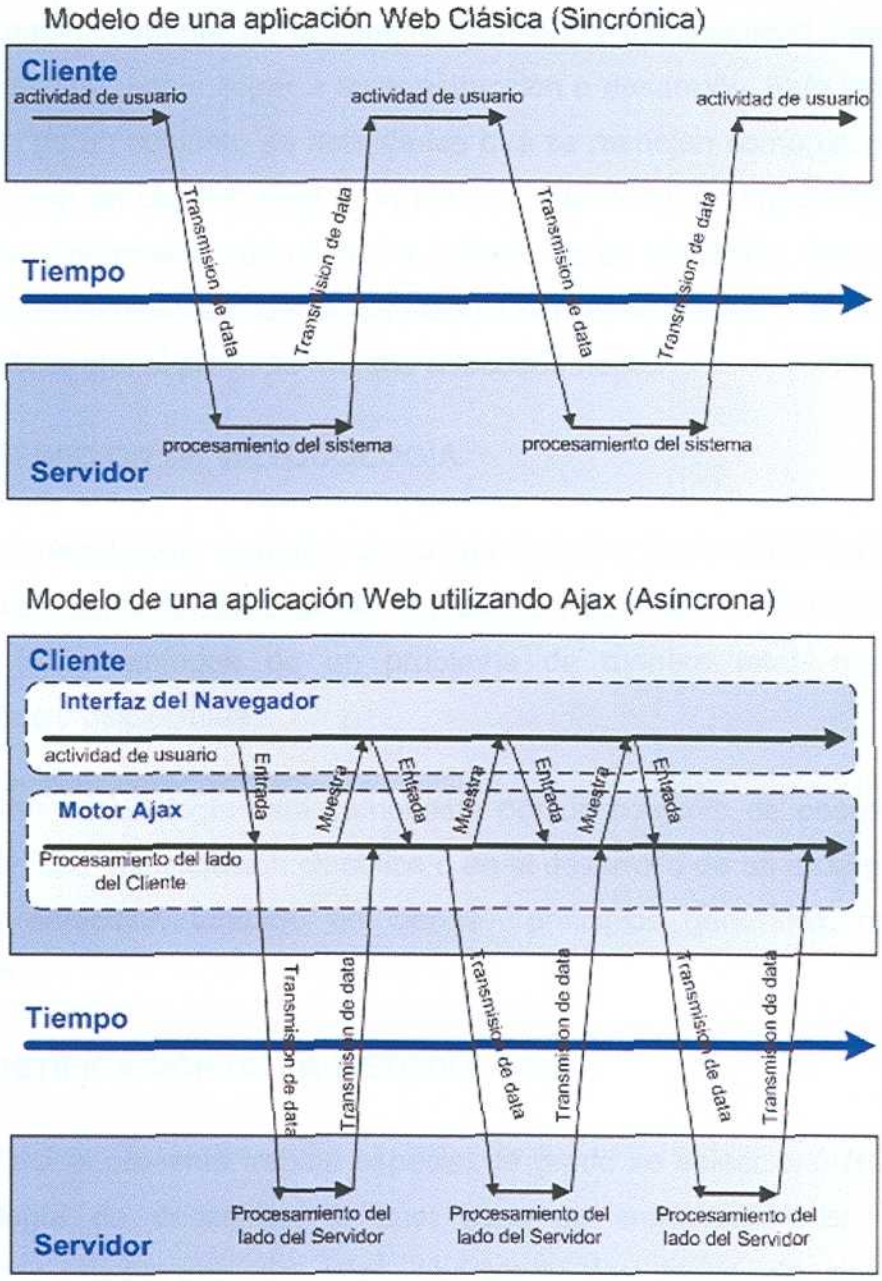

Figura 2.6: Patrón de interacción de una aplicación Web tradicional y de una aplicación AJAX
Fuente: http://adaptivepath.com/publications/essays/archives/000385.PHP

3. MARCO METODOLÓGICO

En el desarrollo de una aplicación se sigue un proceso en el cual se avanza paulatinamente en la comprensión de la funcionalidad requerida y cómo realizarla, hasta llegar a su construcción o desarrollo. Esto requiere la ejecución de un conjunto de actividades que se manejan como un proyecto, es decir, con un objetivo final y un plazo. Y como tal, es importante contar con puntos intermedios de control a lo largo de su ejecución, denominados hitos, que se establecen cuando se elabora el plan de trabajo y sirven de faro para verificar que el proyecto marcha adecuadamente.

3.1 DEFINICIÓN DE METODOLOGÍA

La metodología se define como una ciencia o teoría sobre los métodos para el conocimiento científico de la realidad y para la transformación de la misma. Es el enfoque de un problema de manera total, organizada, sistemática y disciplinada.

Una metodología está compuesta por un conjunto de pasos que se siguen en una investigación científica o en el desarrollo de un sistema. Es un enfoque particular, fundado en ciertos principios generales, de orden filosófico.

3.2 JUSTIFICACIÓN DE LA METODOLOGÍA

Para el presente trabajo especial de grado se seleccionó *RUP* como metodología de desarrollo, la cual basa su implantación en diversas iteraciones, cada una de ellas abarcando las fases de concepción, elaboración, construcción y transición bajo las cuales funciona este modelo.

La decisión de seleccionar *RUP* como guía estuvo basada en la necesidad de utilizar un esquema de desarrollo *evolutivo* e *iterativo*, pues de

esta manera se obtendrían productos tangibles conforme con la ejecución del proyecto.

3.3 METODOLOGÍA *RUP* (PROCESO UNIFICADO DE RATIONAL)

En el modelo en cascada, el proceso de desarrollo avanza en forma secuencial a través de cinco actividades fundamentales: captura de requisitos, análisis, diseño, implementación y pruebas. Éste plantea que cada actividad debe completarse antes de proceder a la siguiente, por lo cual ellas mismas se convierten en referentes para el avance del proyecto en el tiempo y reciben la denominación de fases. Así pues, un proyecto se planifica colocando como hitos la finalización de las distintas fases, donde normalmente se entregan uno o varios productos asociados al desarrollo del sistema [20].

RUP rompe la secuencialidad de las actividades fundamentales del modelo en cascada al plantear un desarrollo incremental e *iterativo*, en el cual no es necesario agotar completamente una actividad para iniciar la siguiente. En lugar de ello, se avanza a través de la construcción de prototipos, cada uno de los cuales exige la ejecución parcial de las actividades fundamentales. Puede verse entonces el desarrollo incremental como una serie de iteraciones, cada una de las cuales se realiza siguiendo el modelo en cascada.

Esta estrategia conlleva que no pueda seguirse utilizando la terminación de las actividades fundamentales para establecer los hitos del proyecto, pues esto sucede hacia el final de su ejecución. Se hace necesario entonces establecer nuevos criterios para definir los puntos de control del proyecto, criterios que estarán determinados por los productos obtenidos en las sucesivas iteraciones [22].

Por esta razón *RUP* organiza las actividades de desarrollo siguiendo dos criterios ortogonales ilustrados en la Figura 3.1. En el eje vertical, se describen las actividades fundamentales o en términos de *RUP* se denominan componentes, los cuales establecen cómo avanzar en la conceptualización y construcción del sistema. El nivel de intensidad del trabajo en cada componente se representa mediante la amplitud de la gráfica asociada y se corresponden con la estructura estática del proceso de desarrollo, pues definen qué acciones se deben realizar. En el eje horizontal, se describen los criterios para la planeación y control en el tiempo, los cuales representan la dinámica del proceso de desarrollo pues establecen cuándo se deber realizar las acciones definidas por los componentes [22].

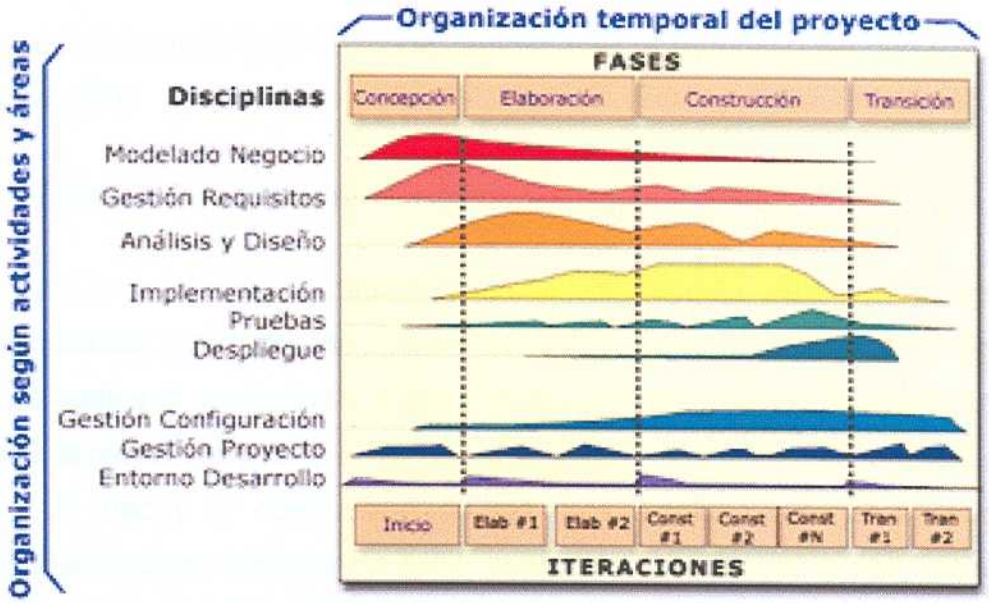

Figura 3.1: Organización del Proceso Unificado
Fuente: ftp://jano.unicauca.edu.co/cursos/Especializacion/ApliServicios/docs/UML-RUP-doc.pdf

3.3.1 Fases de la Metodología *RUP*

Cada fase se realiza en iteraciones sucesivas e incrementales que permiten un proceso de mejoras sucesivas sobre el producto. Cada una concluye en un punto clave, en el cual deben ser tomadas decisiones críticas en el tiempo [19].

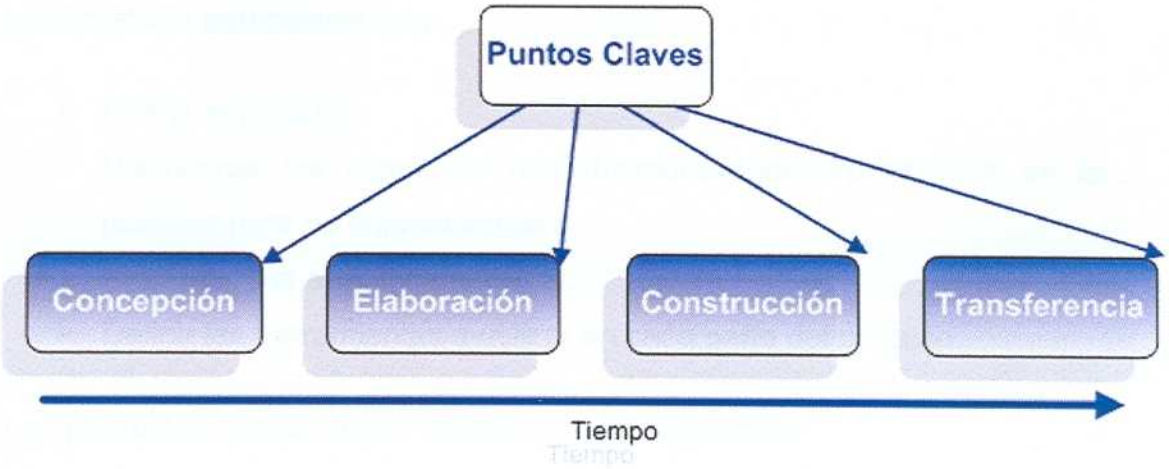

Figura 3.2: Fases y puntos claves en el proceso
Fuente: http://www-128.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

A continuación se describe brevemente el propósito de cada una de las fases [20]:

a) Fase de Concepción o Gestación.

Su propósito es delimitar el alcance del proyecto y elaborar el estudio de factibilidad. Para ello, se deben identificar todas las entidades externas con las cuales el producto va a interactuar (actores) y definir la naturaleza de estas interacciones a alto nivel. Esto implica la identificación de todos los casos de uso y la descripción de los más significativos. El estudio de factibilidad incluye factores de éxito, evaluación de riesgos y estimación de los recursos requeridos, y la planeación de las fases estableciendo las fechas de los hitos principales.

Para cumplir con lo anterior se debe avanzar sustancialmente en los componentes de Modelado de la Organización y Captura de Requisitos.

El hito es la definición de los objetivos, alcance del sistema y el estudio de factibilidad.

Los objetivos particulares son:

- Definir el producto.
- Discriminar los casos de uso (funcionalidades) prioritarios de los posibles pero no imprescindibles.
- Proponer una arquitectura inicial.
- Definir las herramientas a utilizar en cada parte del proceso.

Los productos que se deben obtener son los siguientes:

- Documento de Visión: plantea una introducción que describe el problema, una lista de requerimientos funcionales y no funcionales, riesgos y restricciones.
- Documento SRS (System Requirement Specification): da una descripción detallada de los actores del sistema, se enumeran y describen los casos de uso y se indica que caso de uso corresponde a que módulo del sistema. Adicionalmente se describen detalladamente los requerimientos de cada uno de los casos de uso.
- Diagrama de la arquitectura inicial: Este Diagrama muestra la interacción entre los módulos del sistema planteados.
- Diagramas de casos de uso: Se deben generar los primeros diagramas de caso de uso a un nivel muy general, mostrando la relación entre actores y casos de uso.

b) Fase de Preparación o Elaboración.

Se busca analizar el dominio del problema, establecer una sólida base arquitectónica, desarrollar el plan del proyecto y eliminar los elementos de más alto riesgo.

Puede afirmarse que la fase de preparación es la más crítica de las cuatro fases. Para la mayoría de los proyectos, este momento también

corresponde a la transición entre una operación trasladable, ligera y de bajo riesgo, a una operación de alto costo, gran riesgo y mucha inercia.

En la fase de preparación, se construye un prototipo ejecutable de la arquitectura en una o más iteraciones, dependiendo del alcance, tamaño, riesgo y novedad del proyecto. Este esfuerzo debe dirigirse al menos a los casos de uso críticos identificados en la fase de gestación, los cuales conllevan típicamente los mayores riesgos técnicos. Si bien el objetivo es siempre un prototipo incremental con componentes de calidad de producción, no se excluye el desarrollo de uno o más prototipos exploratorios y de descarte, para mitigar riesgos específicos tales como decisiones de diseño versus requisitos, estudiar la factibilidad de un componente.

Todo esto implica recorrer en varias iteraciones prácticamente todos los componentes del proceso, dejando poco por hacer en el Modelado de la Organización, Captura de Requisitos y Análisis, y habiendo avanzado lo suficiente en las actividades de Diseño, Implementación y Pruebas como para definir la arquitectura.

El hito de esta fase es la definición de la arquitectura del sistema, junto con una revisión detallada de sus objetivos y alcance, y la resolución de los mayores riesgos.

Los objetivos principales incluyen:

- Asegurar que los requerimientos y definiciones obtenidos en la etapa de concepción sean sólidos y se hayan contemplado y mitigado todos los riesgos que podrían afectar el desarrollo del sistema.
- Definir los posibles escenarios de instalación y trabajo, verificando las necesidades de equipamiento de hardware e infraestructura.

- Analizar y profundizar en cada uno de los casos de uso obtenidos en la etapa de concepción para elaborar un prototipo funcional que permita verificar el alcance del desarrollo de software.
- Revisar que los requerimientos de software se correspondan con la estructura actual de trabajo y documentar las propuestas de reorganización.
- Afinar el diseño de las arquitecturas a fin de verificar que sea sólida y cumpla con los requerimientos del sistema.
- Refinar el esquema de desarrollo seleccionando herramientas y metodologías particulares para la etapa siguiente (construcción).

Los productos que se deben obtener son:

- Formularios de casos de uso: se realiza un formulario sobre cada caso de uso donde se especifica una breve descripción, qué actores lo usan, el flujo de eventos para llevar a cabo el caso de uso y casos alternos de flujo.
- Diagrama de actividades: se realiza un diagrama de actividades por cada caso de uso, el cual muestra de manera gráfica el flujo de eventos del mismo. Adicionalmente permite detectar posibles errores en los formularios anteriores.
- Modelo de Dominio: Este modelo muestra el conjunto de clases que van a representar el sistema y la relación que hay entre ellas.
- Diagrama Sistémico: Este diagrama muestra a un nivel más detallado la arquitectura del sistema con los actores y módulos del sistema.
- Diagramas de Casos de Uso refinados: En él se especifican más detalles sobre los diagramas de casos de uso, tales como herencia de actores e inclusión de unos casos de uso dentro de otros.
- Refinar el Modelo de Dominio.
- Crear el Modelo de Base de Datos.

c) Fase de Construcción.

El objetivo es desarrollar todos los componentes y funcionalidades restantes del sistema, e integrarlos en el producto, así como probar completamente todas las características. La fase de construcción es, en cierto sentido, un proceso de manufactura donde el énfasis está puesto en la gestión de los recursos. En este sentido, el foco de atención de la administración sufre una transición desde el desarrollo de propiedad intelectual en las fases de concepción y preparación, hacia el desarrollo de productos tangibles en las fases de construcción y transición.

En proyectos grandes o de gran complejidad se puede adoptar una estrategia de desarrollo incremental con construcciones paralelas. Estas actividades pueden acelerar de manera significativa la disponibilidad de entregas de productos, pero pueden así mismo incrementar la complejidad de la gestión de recursos y la sincronización de los flujos de trabajo. Hay una alta correlación entre una arquitectura robusta y un plan entendible. En otras palabras, una de las cualidades críticas de la arquitectura es su facilidad de construcción. Esta es la razón por la cual se enfatiza tanto durante la fase de preparación en el desarrollo balanceado de la arquitectura y el plan.

El hito de esta fase es la obtención de un producto operacional listo para ponerlo en manos de sus usuarios finales.

Objetivos a cumplir en esta etapa:

- Obtener un sistema de calidad en un tiempo acotado.
- Completar para cada módulo a desarrollar las etapas de análisis, diseño, desarrollo y pruebas.
- Trabajar en paralelo en el desarrollo de subsistemas y módulos que pueden ser elaborados de forma independiente.

- Trabajar de forma iterativa e incremental en el desarrollo, documentando y completando las definiciones de los casos de uso, diseño y pruebas.

d) **Fase de Transición.**

El objetivo principal de esta etapa es transferir el producto a la comunidad de usuarios. Una vez el producto se entrega al usuario final, surgen usualmente asuntos que requieren desarrollar nuevas entregas, corregir algunos problemas o terminar algunas características que habían quedado pospuestas.

La fase de transición se inicia cuando una línea de base del producto está suficientemente madura para ser implantada en el dominio del usuario. Esto normalmente requiere que se haya completado un subconjunto utilizable del sistema con un nivel de calidad aceptable y que la documentación de usuario esté disponible, de manera que la transferencia al usuario produzca resultados positivos para todas las partes.

La fase de transición se enfoca en las actividades que se requieren para poner la aplicación en manos de los usuarios. Típicamente, se tienen varias iteraciones que incluyen entregas beta y entregas de disponibilidad general, así como entregas con errores corregidos y nuevas características. Se dirige un esfuerzo considerable hacia los usuarios en el desarrollo de su documentación, entrenamiento, soporte en el uso inicial del producto. Sin embargo, en este punto del ciclo de vida, la retroalimentación de los usuarios debe centrarse prioritariamente en la sintonización, configuración, instalación y usabilidad del producto.

Objetivos y tareas involucradas en esta fase:

- Instalación del software en el entorno final de trabajo, realizando estas de manera progresiva.
- Conversión e importación de datos anteriores al nuevo sistema.
- Ajuste del software.
- Medición de rendimiento de la herramienta.

Los productos que se deben obtener son los siguientes:

- Documentación detallada a ser entregada a los usuarios finales.
- Documentación técnica del sistema por módulos.
- Realizar las pruebas modulares al sistema.
- Realizar pruebas globales del sistema.

4. DESARROLLO

Tal y como se planteó en el capítulo anterior, la metodología RUP escogida propone un desarrollo incremental, evolutivo e iterativo, en el cual se desarrollan las actividades fundamentales en cada fase a través de la creación de prototipos.

Las iteraciones secuenciales en cada fase permiten un proceso de mejoras sucesivas en cada producto alcanzado, siendo las fases definidas por la metodología las siguientes:

- Concepción o Gestación.
- Elaboración o Preparación.
- Construcción o Desarrollo.
- Transición o Transferencia.

A lo largo de la ejecución del proyecto se lograron completar un total de 9 iteraciones, las cuales son descritas a continuación, especificando los hitos o alcances al final de cada fase.

1.1 ITERACIÓN 1

Al ser la primera iteración, el levantamiento de toda la información necesaria para comenzar con la elaboración del proyecto es lo primordial y esencial pues es la base del conocimiento y entendimiento por parte de los ejecutores de la propuesta de los requerimientos y las necesidades existentes.

Por tal razón el objetivo de esta iteración fue el enfoque de la fase de **concepción**, a través de la cual se obtuvieron los siguientes resultados:

- Delimitación del alcance del proyecto.

- Creación del diagrama a alto nivel de la arquitectura del sistema (Fig. 4.1).
- Definición de los posibles módulos y la interacción entre ellos.
- Identificación de las entidades externas (actores).
- Planeación de las fechas de las fases principales. (Fig. 4.2)
- Generación del documento de Visión donde se describen de manera detallada los puntos anteriores. (Apéndice A)

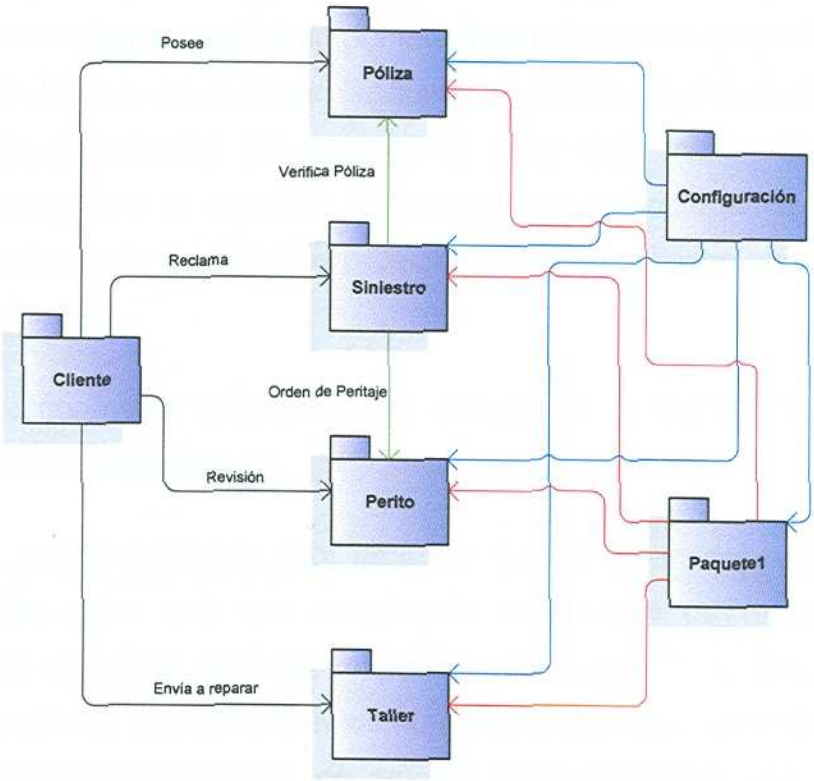

Figura 4.1: Diagrama a alto nivel de la arquitectura del sistema.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

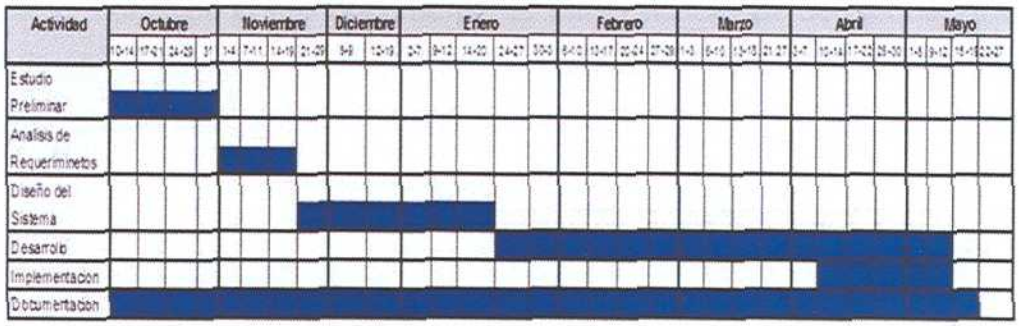

Figura 4.2: Planeación de las fechas de las fases principales.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

1.2 ITERACIÓN 2

El siguiente ciclo consistió en la depuración de los objetivos alcanzados en el paso anterior. A este nivel se hizo necesario el uso de una herramienta de investigación como la entrevista para acopiar los requisitos más importantes del sistema de la mano de especialistas y definir los posibles requerimientos funcionales y no funcionales.

La entrevista se realizó en primera instancia con un grupo de expertos en el área de seguros quienes facilitaron al equipo de desarrollo material didáctico y de consulta de manera de crear una mejor visión de los procedimientos que se ejecutan actualmente en las compañías de seguro donde laboran, y de los cambios propuestos a ser aplicados en el sistema a desarrollar.

El esquema o minuta de las entrevistas y reuniones puede ser examinado en el apéndice G.

Se enfocó entonces esta iteración en las fases de **concepción y elaboración,** pues una vez completado el proceso de recopilación de información aportada por los expertos, se procedió a desarrollar un documento donde se establecen los nuevos requisitos del sistema, refinando los requerimientos existentes, de manera de proporcionar una visión más clara del alcance del proyecto así como del producto a desarrollar. El extracto, conocido como Documento de Especificaciones de Requerimientos del Sistema (System Requeriment Specification, SRS, por sus siglas en inglés), puede ser examinado en el apéndice B.

1.3 ITERACIÓN 3

El siguiente paso fue discutir los objetivos y nuevos requerimientos surgidos a partir de la entrevista con los expertos en donde se lograron definir ciertos puntos claves para el desarrollo del proyecto.

Los especialistas recomendaron estructurar el sistema de manera modular con el fin de adaptarlo a las exigencias de cada empresa, dividiéndolo en capas que posteriormente puedan ser modificas según el crecimiento de la misma. Igualmente, sugirieron aumentar el número de actores que interactúan dinámicamente con el sistema, dando como resultado la definición de los usuarios o actores y módulos de la aplicación.

1.3.1 Usuarios o Actores del Sistema.

Uno de los resultados de la entrevista fue el hecho de determinar los actores que de manera estándar interactúan con el sistema, los cuales se describen a continuación:

- **Perito o Revisor:**

Su función es realizar la inspección de los vehículos, tomar las fotografías al momento de contratar una nueva póliza y a la hora de realizar la revisión de un automóvil por la declaración de un siniestro.

Su interacción con el sistema es vía inalámbrica a través de un dispositivo PDA, con el cual realizará la toma de fotografías y el registro de la información sobre los siniestros.

Basándose en su experiencia, determina el tipo de siniestro declarado, de manera de tomar una decisión en cuanto al análisis del cumplimiento de las reglas o condiciones de la póliza y de esta manera tomar una decisión en cuanto a la cobertura o no del suceso.

Existen dos modalidades de perito o revisor:

- El perito o revisor que se encuentra en el centro de servicio.
- El perito o revisor que visita a los clientes para hacer las respectivas inspecciones, ya sea para una nueva póliza o para un siniestro.

- **Analista de Sistema:**

El rol principal de este usuario consiste en tomar las decisiones referentes a la aprobación o no de los siniestros declarados.

Uno de los objetivos del sistema es disminuir el tiempo de procesamiento y análisis de los siniestros, simplificando su trabajo a estrictamente leer un reporte generado por el sistema en el cual aparece toda la información detallada y necesaria para tomar la decisión final.

El sistema, apoyado en la determinación de condiciones de la póliza versus las características del siniestro, genera dichos reportes en los cuales la última palabra la tiene el analista para decidir el resultado del mismo.

Tiene la posibilidad de generar y consultar las órdenes de reparación que serán enviadas a los talleres y adicionalmente consultar el estado de las pólizas.

- **Usuario Básico:**

Es el encargado de realizar la inserción de la data al momento de generar o renovar una nueva póliza para un cliente.

Tiene la posibilidad de declarar siniestros, así como consultar el estado en que se encuentran los mismos para brindarle dicha información a los clientes que así lo soliciten.

Otra de las actividades que realiza es consultar los datos de cualquier póliza y la lista de los talleres disponibles a los cuales los vehículos siniestrados pueden ser enviados.

- **Usuario Manager:**

Tiene acceso a toda la información que maneja el sistema. Puede tanto consultar como modificar la data referente a pólizas, siniestros y reportes.

Puede también consultar los talleres vinculados a la empresa y los proveedores. Adicionalmente, realiza tanto las tareas del usuario básico como las tareas del analista de sistema.

Tiene la posibilidad de generar reportes en base a la información que él desee.

- **Administrador del Sistema:**

Es el encargado de realizar toda la carga de datos del sistema, entre los cuales se encuentran los siguientes:

- Tipos de Seguro.
- Seguros.
- Pólizas.
- Condiciones de las Pólizas.
- Asignación de talleres.
- Asignación de proveedores.
- Creación de tipos de siniestros.
- Configuración de la interfaz gráfica
- Generación de Reportes.
- Definición de Perfiles de Usuario.
- Creación de Usuarios.

- **Cliente:**

Resulta el actor más relevante del sistema. Interactúa con el mismo a través de la declaración de siniestros, pudiendo hacerlo directamente a través de Internet, llamando al Servicio Telefónico de la empresa o visitando el Centro de Atención al Cliente.

Asimismo, puede consultar el estado de un requerimiento y el resultado del mismo.

1.3.2 Módulos del Sistema

Igualmente se lograron identificar los posibles módulos en los que se encontrará organizada la aplicación:

- **Módulo de Pólizas:** El módulo de pólizas será el encargado de manejar toda la información acerca del asegurado, ya sea una persona, un vehiculo, un inmueble u otro tipo de bien. En este módulo se podrán realizar las operaciones de registro y consulta de los datos de los clientes, sus pólizas y siniestros.
- **Módulo de Siniestros:** Este módulo manejará toda la información relacionada a siniestros de vehículos, la declaración, consulta de estado, carga de data del asegurado y de los siniestros.
- **Módulo de Revisión**: se encarga de todo el proceso de revisión del vehículo a la hora de un siniestro. Maneja el proceso de carga de data del vehiculo y transmisión de las fotografías. Vale la pena recalcar que este módulo existe ya que en el alcance se contempla solo los siniestros de vehículos.
- **Módulo de Talleres**: Es el encargado de la gestión de los talleres por parte de la compañía aseguradora. Administra la información de los talleres asociados a la aseguradora y maneja todo el proceso de generación de las órdenes de reparación de los vehículos. Al igual que el

módulo anterior, es considerado debido al enfoque del presente trabajo al área de siniestros de automóviles.

- **Módulo de Configuración**: Administra las configuraciones visuales del sistema. Maneja el proceso de creación y asignación de Perfiles de Usuario. Permite que la interfaz de la aplicación sea configurable.
- **Módulo de Reportes**: Permite la consulta de reportes preestablecidos o la creación de nuevos reportes de manera dinámica.

1.3.3 Arquitectura del Sistema

El sistema se planteó para ser desarrollado bajo una arquitectura Web donde la aplicación puede ser accesada a través de un explorador, hoy día presente en todos los sistemas operativos, y que estará almacenada en los servidores de la empresa de manera que cumpla con el diseño mostrado en la figura 4.3.

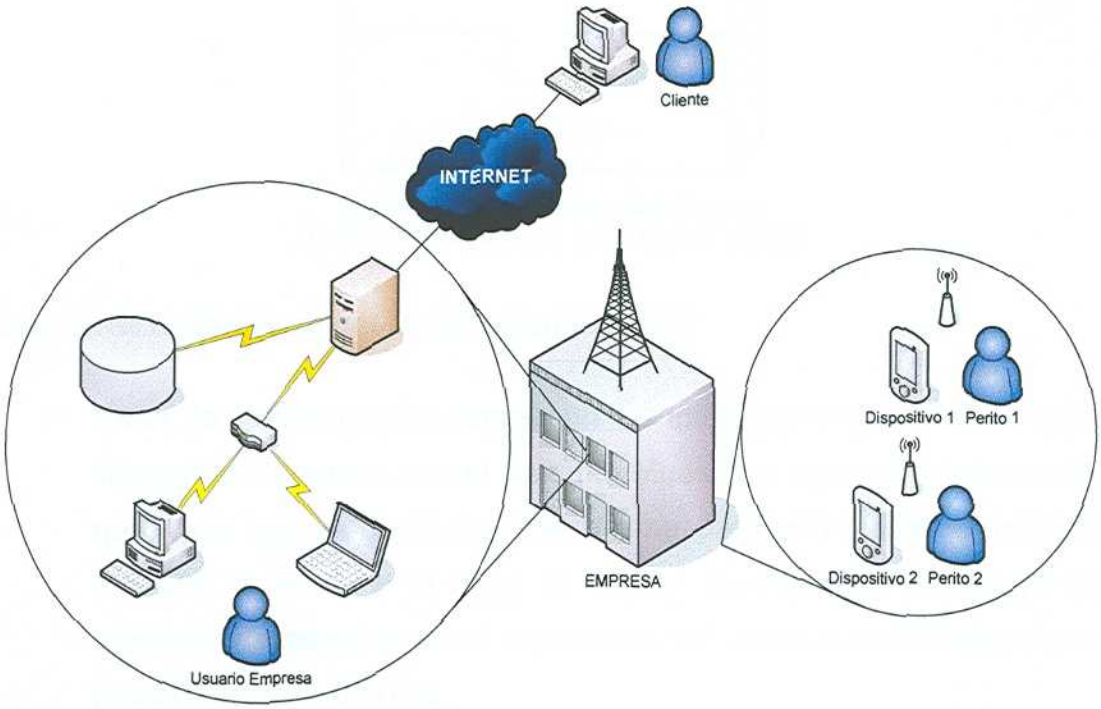

Figura 4.3: Arquitectura de la Aplicación
Diseño: Bohórquez, D. y Miñarro, A. (2006)

1.3.4 Arquitectura de Desarrollo

Se decidió trabajar con el *framework* Struts, que mantiene un *patrón de diseño* MVC modelo 2, es *código abierto* y su implementación en cualquier sistema puede llegar a ser bastante rápida y sencilla.

Este patrón permite implementar una arquitectura organizada en tres capas (figura 4.4), lo que hace que el desarrollo se pueda llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido lo que ofrece escalabilidad a la aplicación.

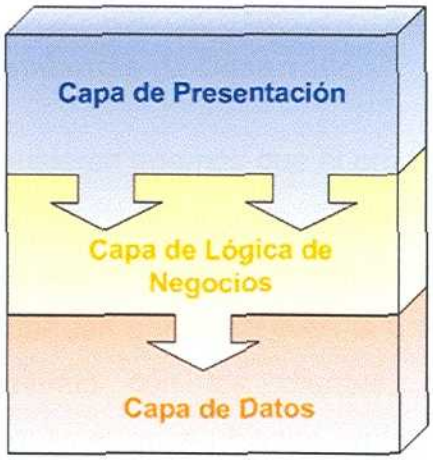

Figura 4.4: Arquitectura del Sistema
Diseño: Bohórquez, D. y Miñarro, A. (2006)

Las capas definidas se describen a continuación:

- **Capa de Presentación:** representa las interfaces visuales con las que interactúa el usuario con el sistema. Muestra y captura la información que, una vez validada por ella misma para comprobar el correcto formato, la envía a la capa de lógica del negocio. Constituye los archivos visualizados en el explorador del usuario cuyas extensiones pueden ser JSP o *HTML*.
- **Capa Lógica del Negocio:** recibe las solicitudes del usuario y ejecuta las acciones definidas para un procedimiento en particular, luego

comunica los resultados. Es aquí donde se establecen todas las reglas de la empresa que deben cumplirse. Se comunica con las capas de Presentación, para recibir las solicitudes y presentar los resultados, y con la Capa de Datos para almacenar o recuperar información del manejador de Base de Datos. Constituye los archivos conocidos como *servlets*.

- **Capa de Datos:** residen los datos. Representan los procesos de almacenamiento y recuperación de la información. Constituyen las funcionalidades ofrecidas por el manejador de Base de Datos como *Procedimientos Almacenados* y *Disparadores*.

1.3.5 Ambiente y herramientas de Desarrollo

Los expertos propusieron además que el proceso de revisión por parte del perito esté incluido dentro de las funcionalidades del sistema, para lo cual se hace necesario contar con un dispositivo inalámbrico capaz de establecer comunicación e interactuar con el mismo pues la presencia de éste actor, al estar en contacto contínuo con los clientes, se encontrará en las inmediaciones del centro de revisión ubicado en las afueras de la empresa, donde no existe acceso directo a un equipo de computación o similar.

Por las razones anteriores, se decidió utilizar entonces el ayudante personal digital (*PDA*, por sus siglas en inglés) modelo Zire 72, desarrollado por Palm®, cuyas características permiten realizar todo el proceso de revisión a través del mismo aparato, ya que además de poder acceder al sistema y realizar la carga de información correspondiente, es posible tomar las fotografías de los vehículos, paso esencial en el proceso de revisión, y enviarlas directamente al sistema para su posterior análisis.

El ámbito de desarrollo está enfocado en un ambiente código abierto donde las herramientas utilizadas (en su mayoría) son de libre licenciamiento. Los componentes de software utilizados en el desarrollo del

sistema se listan a continuación:

Plataforma Java, Edición Empresarial (J2EE)

Es un entorno de desarrollo para la construcción de aplicaciones y componentes mediante el uso del lenguaje de programación Java. Comprende un conjunto de especificaciones y funcionalidades orientadas al desarrollo de aplicaciones empresariales. Se utilizó como plataforma base para la programación del sistema.

Apache Tomcat 5.5

Funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Se le considera un servidor de aplicaciones. Permite la ejecución por completo de la aplicación.

Eclipse SDK IDE 3.1

Es un ambiente integrado de desarrollo (IDE) para la edición, compilación y depuración de programas basados en tecnología Java. Su uso como analizador sintáctico y semántico facilita el proceso de codificación en muchos aspectos.

MyEclipse 4.0

Al igual que eclipse, es definido como un ambiente integrado de desarrollo pero embebido dentro de Eclipse como un *plugin*, lo cual permite darle a Eclipse mayores y mejores propiedades para el desarrollo de aplicaciones, incluyendo el soporte para la implementación frameworks como Struts, Hibernate, Spring, entre otros.

MySQL 5.0

Es un manejador de base de datos desarrollado bajo licencia de código abierto, con capacidad de administrar altos volúmenes de data. La actual versión utilizada en el desarrollo del presente trabajo ofrece funcionalidades especiales como procedimientos, disparadores y vistas, disponibles hasta hace poco únicamente en sistemas de gran renombre como Oracle, DB2, entre otros, que pueden llegar a ser muy complejos y requerir arquitecturas muy completas.

CVS NT

Así mismo, se decidió utilizar un sistema de control de versiones (CVS, por sus siglas en inglés) el cual permite gestionar las versiones de todos los ítems de configuración que forman la línea base del producto o una configuración del mismo. Este tipo de sistemas facilitarán la administración de las distintas versiones de cada producto desarrollado junto a los cambios que surgirán conforme el desarrollo del sistema. La aplicación seleccionada para este rol es CVS NT ya que, además de ser una de las más populares entre la comunidad de desarrolladores a nivel mundial, posee licencia de código abierto.

1.3.6 Nuevos Requerimientos

Al ser un nuevo ciclo, como resulta obvio surgieron nuevos requerimientos con respecto a la arquitectura del sistema, que fueron adaptados al diagrama mostrado en la figura 4.1 y que pueden ser observados en la gráfica 4.5.

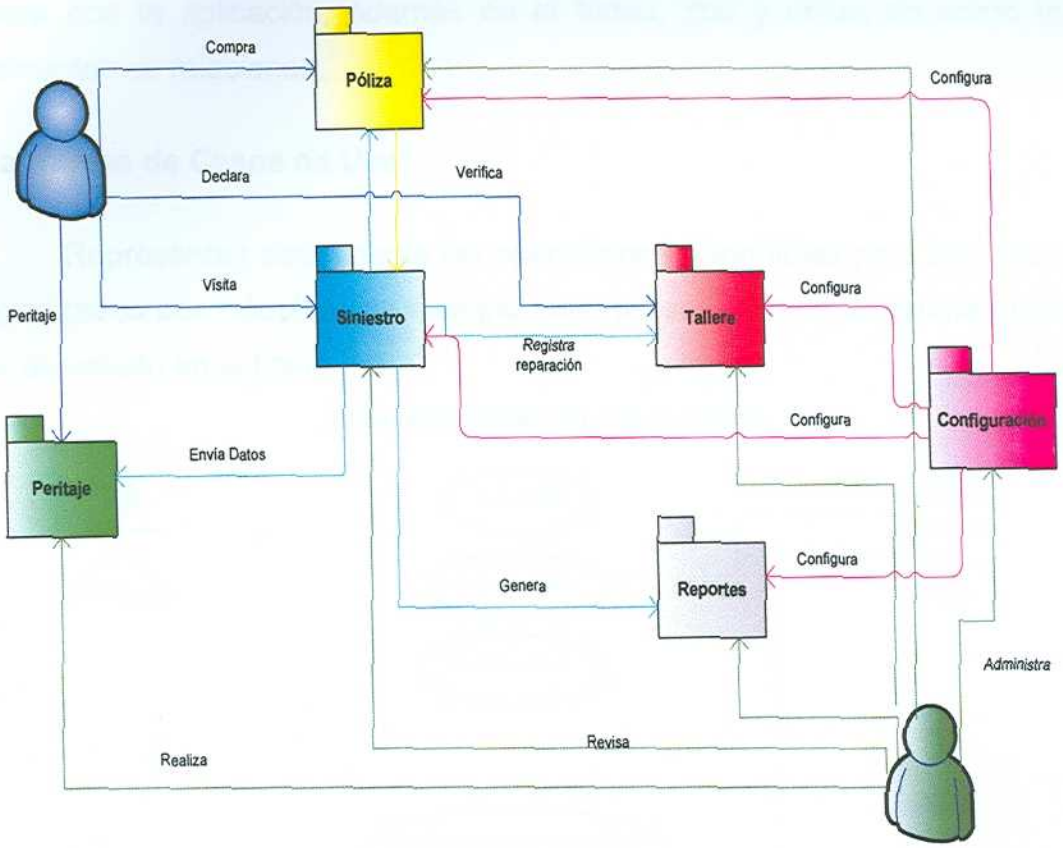

Figura 4.5: Diagrama definitivo del sistema.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

1.4 ITERACIÓN 4

Una vez definida la arquitectura, se procedió a determinar los requerimientos funcionales y no funcionales (apoyados en los diagramas de casos de uso), generar una descripción sólida de cada una de las actividades, describir la relación entre los objetos del sistema y definir un modelo de datos en el cual se basará la aplicación, con lo cual se orientó esta iteración en la fase de **elaboración** propiamente.

1.4.1 Casos de Uso

Ya precisados los actores que tendrán interacción con el sistema, se realizó una representación detallada de la manera en que cada uno de ellos

opera con la aplicación, además de la forma, tipo y orden en como los elementos se relacionan.

Diagramas de Casos de Uso:

Representan visualmente las operaciones disponibles para cada actor organizados por módulos. Un ejemplo claro de este tipo de diagrama puede ser apreciado en la figura 4.6.

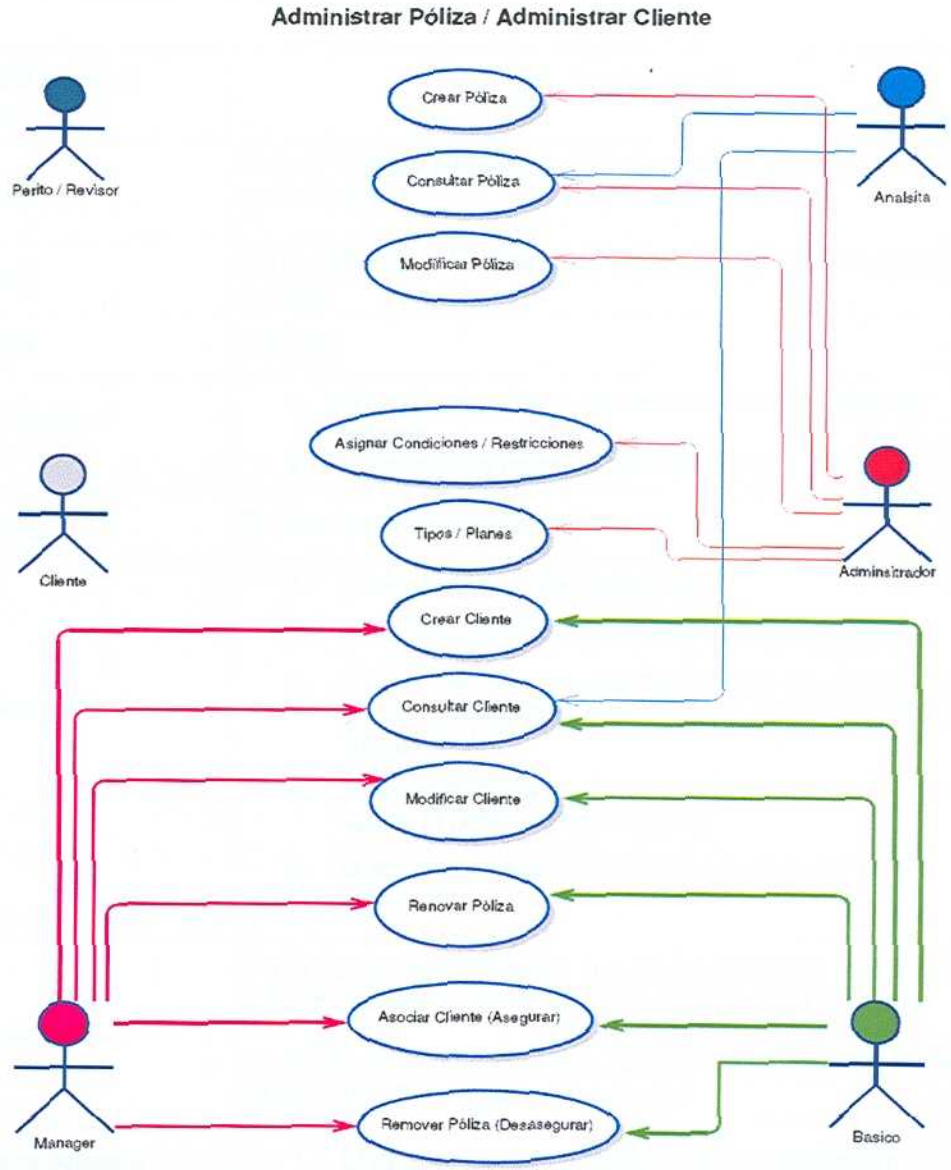

Figura 4.6: Diagrama de Casos de Uso para los módulos Administrar Póliza y Administrar Cliente, de los actores que componen el sistema.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

Los diagramas para los casos de uso restante pueden ser examinados en el apéndice C.

Formulario de Casos de Uso:

Además de los diagramas anteriormente definidos, se describieron los casos de uso de manera más detallada a través de formularios, como puede ser apreciado en la Tabla 4.1.

#De caso de uso/ nombre	***E0 – Iniciar Sesión***
Descripción	El actor ingresa a la aplicación mediante un nombre de usuario y contraseña.
Actor(es)	Perito o Revisor, Administrador, Manager, Básico, Cliente y Analista
Prioridad	Esencial.
Pre-condición / Presunciones	1. El actor debe haber sido creado como usuario del sistema. 2. El actor debe tener un perfil asignado.
Disparador	El actor ingresa a la aplicación.
Flujo de eventos	1. El actor ingresa a la aplicación 2. El sistema muestra la pantalla de inicio de sesión. 3. El actor introduce su nombre de usuario y contraseña en los campos correspondientes y hace clíc en el botón "Iniciar Sesión". 4. El sistema valida los datos y el tipo de usuario para cargar el perfil correspondiente. 5. El sistema muestra la pantalla principal del perfil seleccionado.
Flujos alternos	4.a Si no es posible realizar la consulta por algún motivo, el sistema mostrará un mensaje que así lo indique (E0M1). 4.b Si los datos no son válidos, el sistema muestra un mensaje de error (E0M2) y vuelve a solicitarlos.
Post-condiciones	1. El usuario ingresa exitosamente a la aplicación.

Tabla 4. 1: Formulario de Casos de Uso para Iniciar Sesión
Diseño: Bohórquez, D. y Miñarro, A. (2006)

Estos formularios permiten tener una clara idea de la secuencia de pasos necesaria para completar el caso de uso. Adicionalmente es posible conocer los actores que hacen uso de él, las condiciones necesarias del sistema antes de su ejecución (presunciones) y los resultados posteriores a su ejecución (Post-condiciones). También se indican los posibles caminos o flujos alternos que podrían tomarse en algún paso de la secuencia establecida en el flujo de eventos principal.

Cabe destacar que estos formularios se identifican de acuerdo el criterio de importancia del caso de uso según el cual puede ser Esencial (el identificador llevará la letra E antepuesta al número del caso) o Importante (el identificador tendrá la letra I anterior al número del caso).

Los casos de uso esenciales sugieren la necesidad de su existencia para completar el sistema. Mientras que los importantes son funcionalidades adicionales a los requerimientos básicos de la aplicación y que no juegan un papel determinante en la culminación del sistema.

Los formularios correspondientes a los demás casos de uso pueden ser revisados en el apéndice D.

1.4.2 Diagramas de Actividades

Individualmente para cada caso de uso especificado, se definieron una serie de actividades, acciones y/o eventos que permiten visualizar los pasos a seguir para completarlo. Un claro ejemplo de este tipo de diagramas puede ser observado en la figura 4.7.

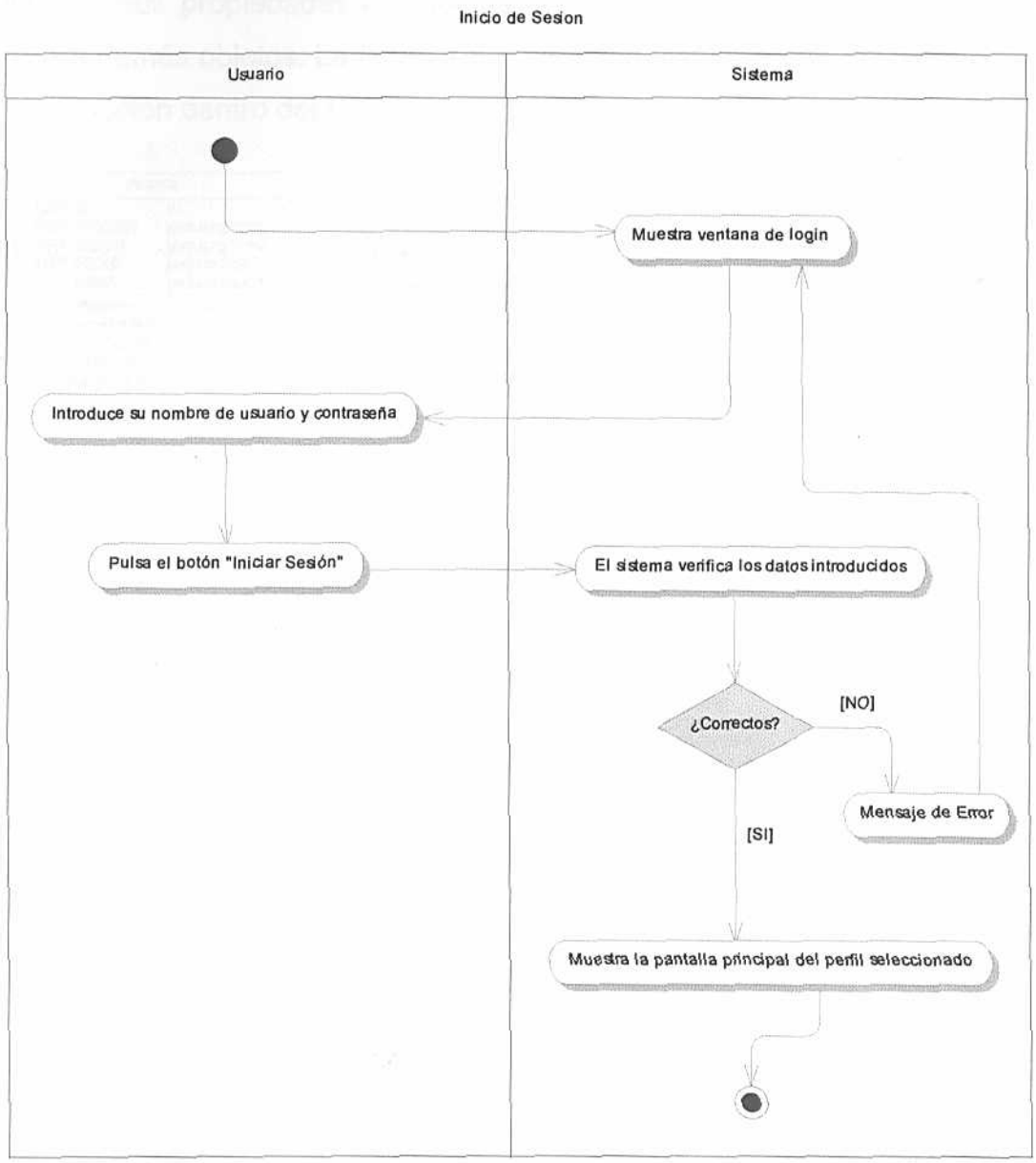

Figura 4.7: Diagrama de Actividades para el Caso de Uso Iniciar Sesión.
Diseño: Bohórquez, D. y Miñarro, A. (2006).

Asimismo, los diagramas correspondientes a cada caso de uso pueden ser encontrado en el apéndice E.

1.4.3 Modelo de Dominio

Reúne las clases de objetos y sus asociaciones. En este diagrama se representa la estructura y el comportamiento de cada uno de los objetos del

sistema, sus propiedades o atributos, su comportamiento y sus relaciones con los demás objetos. La figura 4.8 representa una visión de los objetos que tienen acción dentro del sistema.

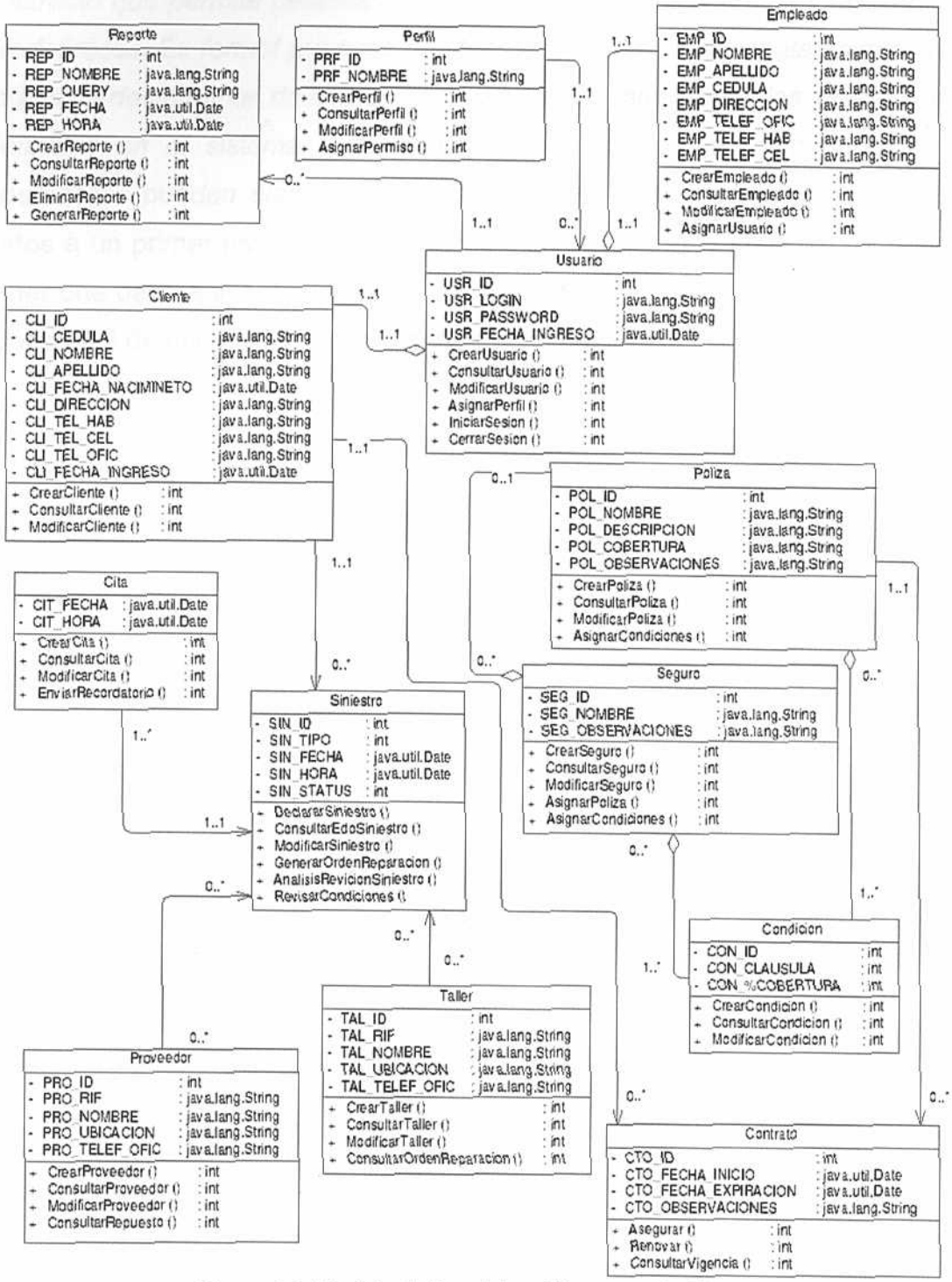

Figura 4.8: Modelo de Dominio o Diagrama de Clases
Diseño: Bohórquez, D. y Miñarro, A. (2006).

1.4.4 Modelo de Datos

Según [Ullman1999]: "*Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. Es formal pues los objetos del sistema se manipulan siguiendo reglas perfectamente definidas y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar.*". De tal manera, la representación de estos datos a un primer nivel se realizó en esta iteración con la finalidad de poder tener una base antes de comenzar la etapa de desarrollo o construcción. La gráfica 4.9 demuestra el primer diseño del modelo de datos.

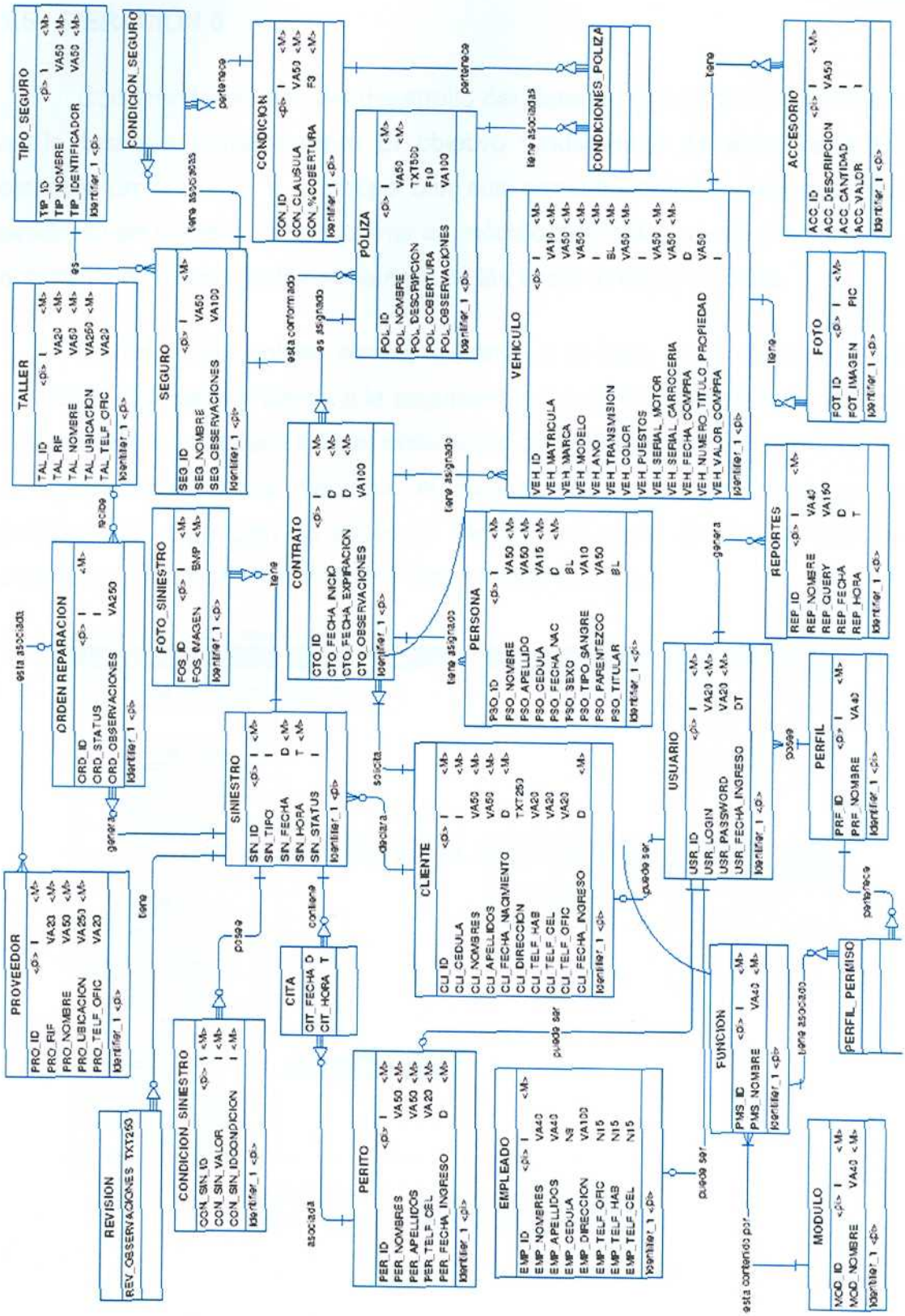

Figura 4.9: Modelo de Datos inicial.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

1.5 ITERACIÓN 5

Comprende la base del desarrollo del sistema, orientada directamente en la fase de **construcción**. El objetivo fundamental de esta etapa es elaborar un producto o prototipo con sus correspondientes componentes, separado de manera independiente en módulos, completamente funcionales, que irán siendo agregados al sistema en las posteriores iteraciones.

En este ciclo inicial correspondiente a la fase de construcción se concedió especial relevancia a la seguridad del sistema y la forma en que los usuarios tendrán acceso a éste, obteniendo como primer producto el módulo de gestión de perfiles mediante el cual el usuario administrador podrá gestionar la asignación de acceso a tareas para cada uno de los perfiles creados. La figura 4.10 ofrece una idea de éste módulo:

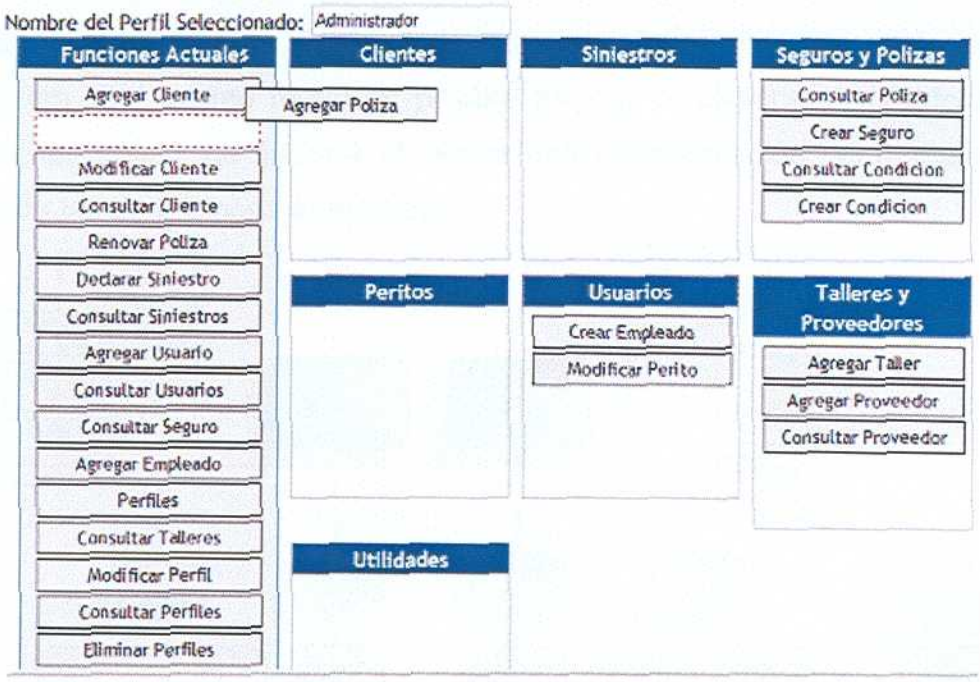

Figura 4.10: Módulo de Gestión de Perfiles.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

Adicionalmente, los procesos de validación de usuarios, registro de clientes, empleados y peritos, también fueron completados en esta iteración, así como la gestión de los talleres y proveedores.

1.6 ITERACIÓN 6

Comenzó una nueva iteración cuando se analizaron los documentos obtenidos a partir de la reunión con los expertos sobre las distintas pólizas, seguros y condiciones existentes en el mercado.

Se definen seguros que identifican el contrato directo que existe entre el cliente y la empresa según lo que se va a asegurar, bien sea personas, vehículos, y demás bienes. A este nivel ya es posible especificar condiciones o restricciones según el seguro contratado. Adicionalmente es posible definir por cada seguro diferentes pólizas que normalmente las compañías de seguro ofrecen con la finalidad de darle al cliente distintas opciones de cobertura. Del mismo modo es posible asignar condiciones y restricciones para cada póliza. La figura 4.11 ofrece una representación del modelo que engloba las definiciones anteriores.

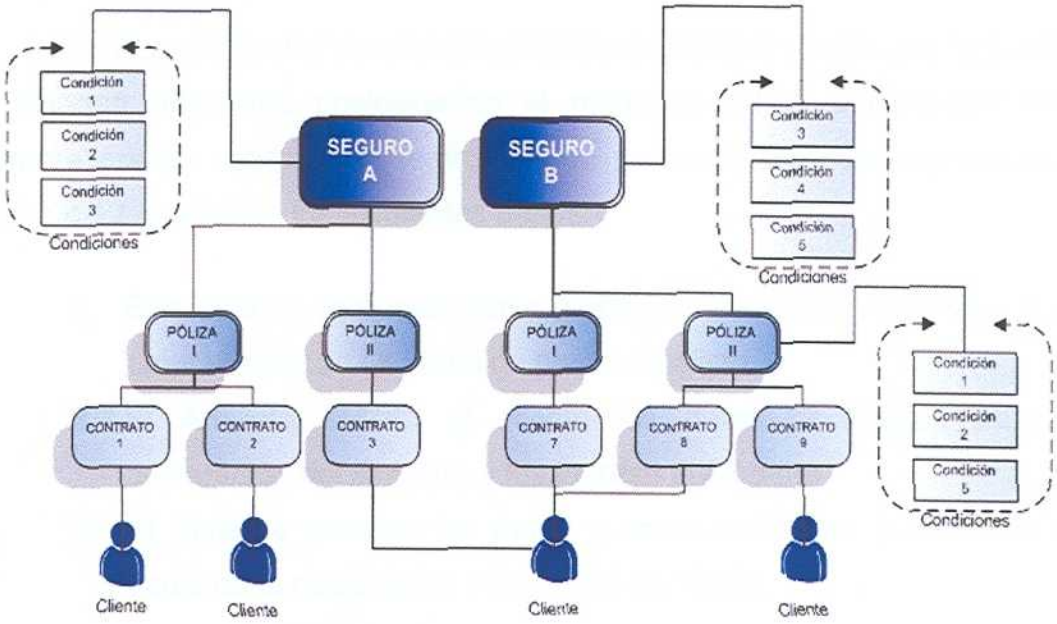

Figura 4.11: Modelo de Seguros, Pólizas y Condiciones.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

Los documentos ofrecidos sugieren la existencia de condiciones o restricciones que aplican tanto a los seguros como a las pólizas, y que tendrán especial relevancia una vez declarado el siniestro del vehículo, pues la aprobación o no de la orden de reparación y la cantidad aprobada, depende básicamente del cumplimiento o no de las condiciones asociadas a la póliza del cliente.

1.7 ITERACIÓN 7

Una nueva iteración se inició cuando surgió la necesidad de consultar nuevamente con el grupo de expertos para concretar ideas referentes al proceso de declaración de siniestros de vehículos y a la participación del perito y el cliente en el mismo.

La reunión tuvo lugar en las instalaciones de la empresa de seguros que brindó el apoyo en el proceso de investigación donde, luego de discutir los puntos acordados, se logró definir el proceso óptimo de declaración de siniestros y las acciones y actividades a realizar por el cliente, el perito y el analista dentro del mismo proceso.

La idea primordial es ofrecerle total comodidad al cliente, por lo cual se le brindan diferentes posibilidades al momento de la declaración. Una primera opción consiste en realizar dicho proceso como se sugiere en la figura 4.12, siguiendo los siguientes pasos:

1. El cliente se comunica con el servicio de atención telefónica de la empresa donde es atendido por un Analista.
2. El Analista solicita al cliente los datos necesarios para la declaración del siniestro y los ingresa al Sistema.
3. El Sistema procesa los datos y envía al Cliente y al Perito los datos de la declaración y de la cita generada.

4. El Cliente y el Perito asisten al Centro de Revisión para realizar la evaluación correspondiente del vehículo.
5. El Perito envía los datos de la revisión al sistema.

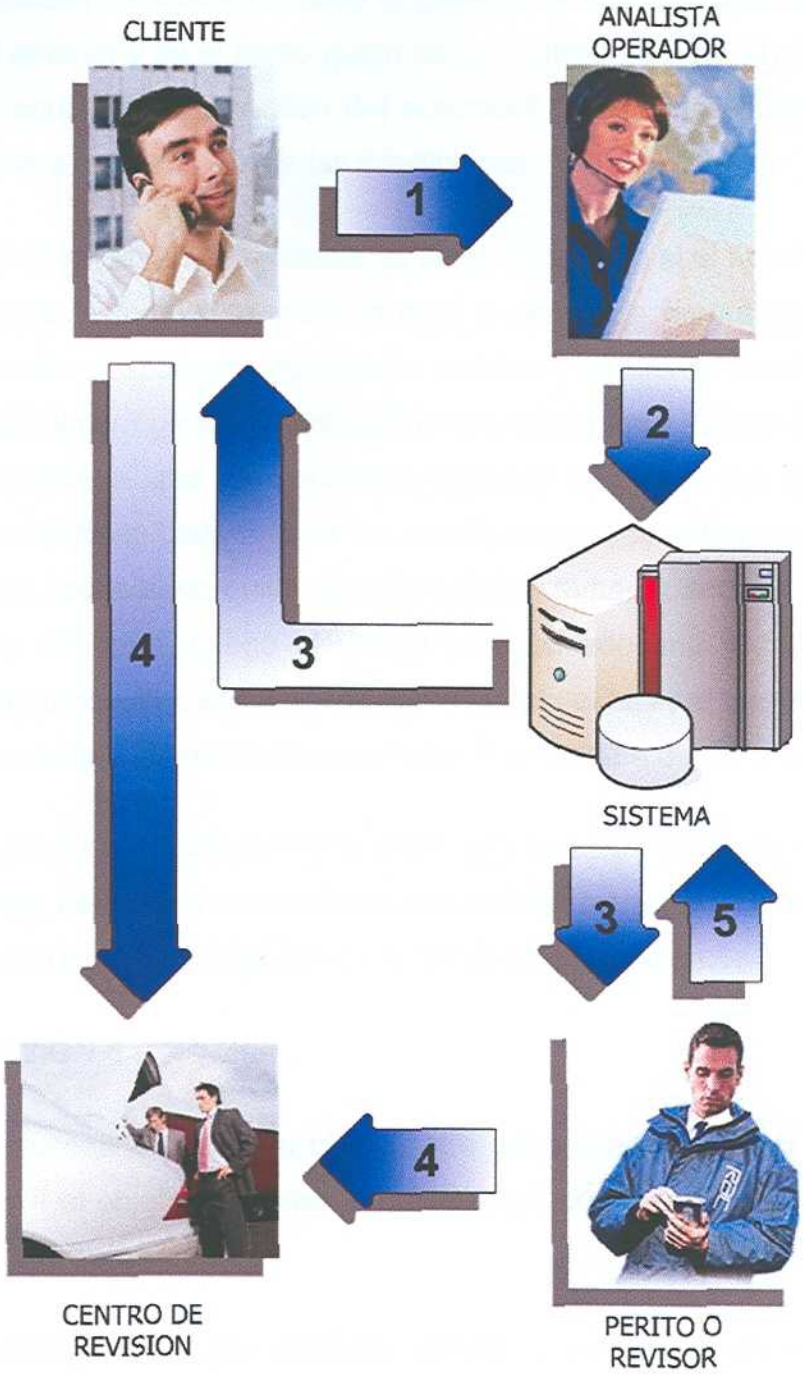

Figura 4.12: Diagrama de Proceso de Declaración de Siniestro. Diseño: Bohórquez, D. y Miñarro, A. (2006)

Otras dos posibles alternativas para el cliente son detalladas en el apéndice F.

El proceso de análisis incluye la revisión de las cláusulas definidas en la iteración anterior y es el perito quien tiene la última palabra al momento de decidir, de acuerdo a la revisión del automóvil, el tipo de siniestro y si el mismo requiere o no análisis de las condiciones.

Basándose en su experiencia, el perito determina si el tipo de siniestro es de Respuesta Inmediata, para lo cual la orden de reparación se emite inmediatamente después de concluir la revisión y el cliente puede acudir al taller indicado y corregir los daños sufridos por el vehículo. Si por el contrario, el perito considera que es necesario conocer aún más los detalles del siniestro, se verifican cada una de las condiciones de la póliza y se remite el siniestro a un especialista quien se encargará de tomar la decisión final sobre el resultado del caso. Luego el cliente es notificado una vez conocido el resultado del proceso y, si así aplica, se le señala el taller al cual debe asistir para la reparación y el monto aprobado por la compañía de seguros.

Los productos obtenidos al final de esta iteración incluyen los módulos de Siniestros, dentro del cual estarán disponibles las tareas de declaración, consulta y aprobación o negación de la orden de reparación.

1.8 ITERACIÓN 8

Fue necesario un nuevo ciclo dentro del proceso evolutivo cuando se buscó cumplir el objetivo de crear un modelo de configuración de módulos e interfaz para la aplicación.

La configuración por módulos ofrece al sistema características de escalabilidad y crecimiento de tal manera que, cuando la empresa así lo requiera, la integración con nuevas funcionalidades y utilidades no será

mayor problema. La figura 4.13 expone un claro ejemplo de la organización por módulos definida en la aplicación.

Figura 4.13: Organización por Módulos.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

La configuración de la interfaz permite al usuario total libertad al momento de darle al sistema el estilo visual de la empresa, mediante la definición de colores, formatos e imágenes.

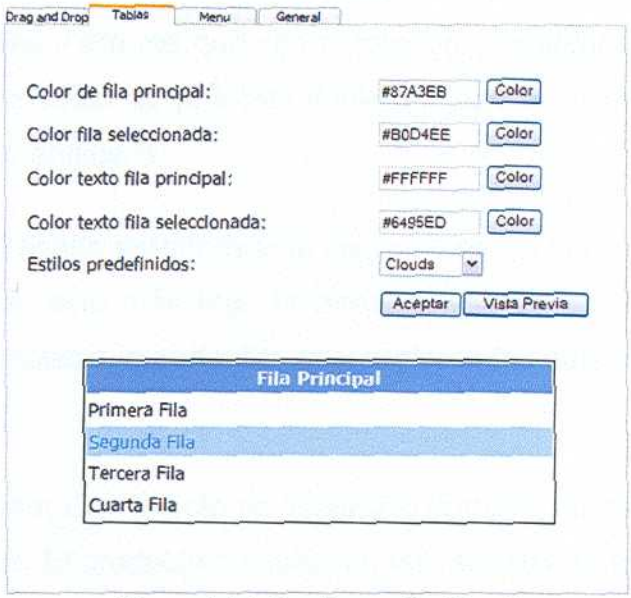

Figura 4.14: Configuración Visual.
Diseño: Bohórquez, D. y Miñarro, A. (2006)

1.9 ITERACIÓN 9

Finalmente surge la última iteración en el desarrollo del presente Trabajo Especial de Grado cuando se procedió a transferir el sistema ya completado al grupo de expertos consultado, de manera de obtener su evaluación final, tras una serie de pruebas y demostraciones, con lo cual se logra certificar y garantizar que los objetivos planteados fueron alcanzados con éxito y que parte de los requerimientos ofrecidos por el panel entrevistado se canalizaron correctamente, dando como resultado un sistema íntegro cuya implementación e implantación dentro las empresas consultadas tan solo depende del cumplimiento de la arquitectura que se presenta en este documento y de las recomendaciones realizadas con el fin de mejorar los servicios ofrecidos al cliente.

Se orienta entonces este ciclo en la etapa de **transición** donde el producto final es entregado a la comunidad de usuarios, en el caso del presente Trabajo, los expertos en el área.

Al ser una metodología iterativa y evolutiva, RUP ofrece la posibilidad de realizar pruebas constantes sobre cada uno de los productos obtenidos, corrigiendo detalles y errores que van surgiendo a medida que se avanza en el desarrollo. El modelo de pruebas implementado en cada fase puede ser apreciado en el apéndice H.

Por lo expresado anteriormente las pruebas no fueron consideradas al final de esta fase, sino más bien la transferencia de la aplicación permitió demostrar dos hipótesis importantes que sugieren los objetivos a alcanzar en esta fase [22]:

- **Verificación:** El producto se construyó correctamente.
- **Validación:** El producto cumple con los requerimientos planteados.

En este punto el desarrollo ya se ha completado junto con las pruebas y la implementación del sistema, a partir de lo cual es posible visualizar el ciclo de vida del presente proyecto a partir de una sencilla gráfica (Figura 4.15) donde se relacionan las iteraciones realizadas y las fases involucradas en cada una de ellas.

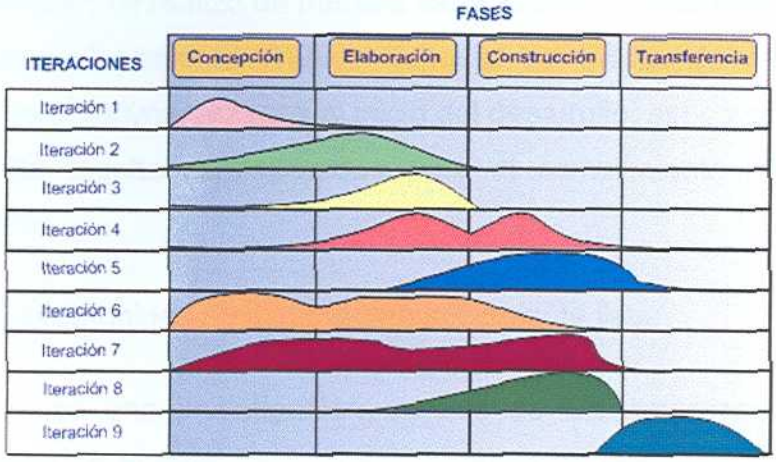

Figura 4.15: Ciclo de Vida del Proyecto
Diseño: Bohórquez, D. y Miñarro, A. (2006)

5. RESULTADOS

Luego de haber completado todas las iteraciones mencionadas en la etapa de desarrollo, pasando en cada una de ellas por las respectivas fases de concepción, elaboración, construcción y transferencia que componen a la metodología RUP, se realizó de manera satisfactoria el desarrollo del sistema planteado para el presente Trabajo Especial de Grado; de modo que se alcanzaron los objetivos trazados al inicio del desarrollo, así como también se obtuvieron los resultados esperados para el cumplimiento de todos los requerimientos.

Los resultados obtenidos se resumen en la siguiente lista:

- Se realizó una investigación en dos de las empresas con mayor mercado en el país como lo son Seguros La Previsora y Seguros La Seguridad con el objetivo de determinar cuales son los tipos de póliza que se manejan en la actualidad. La investigación, además de permitir formar un conocimiento sólido en el área de seguros que fue sustentado y plasmado en el marco referencial del presente documento, proporcionó una vía para obtener recomendaciones, planteando que el sistema pudiese manejar de forma configurable los tipos de seguro, los tipos de póliza, condiciones o cláusulas y junto a ellos los contratos que se le podían ofrecer a los diversos tipos de cliente. El resultado final fue establecer el manejo de los seguros de una forma escalar, es decir en un primer nivel se encuentran los seguros con sus respectivos tipos ya sean de automóviles, salud u otro, bajo estos seguros se encuentran las pólizas de seguro que serian los diversos planes que pueden contratar los clientes y por ultimo se encuentra el contrato que es la relación entre la compañía y sus clientes (Figura 4.10).
- Se indagó sobre los procedimientos y procesos internos relacionados

con las pólizas en las compañías aseguradoras para poder comprender el flujo de la información, logrando entender como se manejan en la actualidad la parte operativa correspondiente a contratos de seguros y la forma en que toda esta información es utilizada por los diversos entes dentro de la misma empresa. Procedimientos como asegurar, desasegurar, renovar y anular pólizas o contratos de seguro fueron analizados y estudiados para poder implementarlos en el sistema.

- Se realizó una investigación exhaustiva en el área de siniestros, y más específicamente de siniestros de automóviles, que como fue mencionado en el alcance del presente trabajo, resulta ser el proceso principal y de mayor énfasis por lo tanto conforma el centro del sistema. Se clasificaron los siniestro en dos tipos, los siniestros de respuesta inmediata que son aquellos que no requieren repuestos ni revisión por parte de los analistas de la aseguradora y los siniestro que requieren un análisis o evaluación del suceso con respecto a las condiciones y cláusulas del contrato del asegurado, además que es posible que entren repuestos que necesitarían una búsqueda en las compañías proveedoras. En cuanto a las actividades relacionadas a este proceso se determinaron la declaración del siniestro por parte del cliente, que se realizaría a través de Internet, mediante una llamada telefónica al centro de atención al cliente o visitando directamente el centro de servicio. La revisión del vehículo por parte del perito o revisor de la empresa, que se realizaría mediante el uso de un dispositivo móvil y por último la toma de fotografías por parte del revisor y el análisis del siniestro por parte de un analista si así se requiriese.
- Luego de estudiar y analizar la información manejada por las empresas aseguradoras se estableció un conjunto estándar de datos que sirvió de piedra angular en el diseño e implementación de un modelo de datos genérico capaz de ser adaptado al cualquier

compañía del ramo, pues el mismo agrupa toda la información necesaria básica a ser procesada por la empresa como Clientes, Contratos, Seguros, Pólizas, Condiciones y Siniestros.

- Se logró definir una estructura funcional de los requerimientos organizada en módulos, de manera que el sistema se puede adaptar de forma sencilla según los requisitos de la empresa. Resulta fácil de actualizar y mantener, y posee una gran flexibilidad a la hora de agregar nuevos componentes a la aplicación.
- Se desarrolló un modelo de configuración de módulos e interfaz que permite manejar toda la permisología y los perfiles de acceso al sistema de una forma amigable, es decir, que la compañía aseguradora podría definir los módulos que desea utilizar y a su vez controlar los perfiles que le serán asignados a los usuarios del sistema. En cuanto a la interfaz gráfica del sistema también se desarrollo de forma configurable de acuerdo a los requerimientos en cuanto a estándares de interfaz que maneje la empresa, de esta forma se puede manejar todo lo referente a interfaz gráfica de la forma en que se desee.
- Se implementó una arquitectura por capas que es la base de la aplicación, apoyada en el patrón de diseño Modelo, Vista y Controlador (MVC). El componente Modelo maneja todo lo referente a los datos, está formada por un manejador de bases de datos que realiza todo el manejo de persistencia y recibe las transacciones desde la capa de negocio. El controlador por otra parte, forma la capa de la lógica del negocio donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso, es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para completar las transacciones de almacenamiento o recuperación de datos vinculadas al manejador de base de datos. Por

último, el componente de la vista constituye la capa de presentación, a través de la cual el usuario interactúa con la aplicación. Esta capa se comunica únicamente con la capa de negocio.

- Vale la pena mencionar la correcta y exitosa integración con el conjunto de tecnologías que engloba la técnica de desarrollo AJAX con el framework *Struts*, lo que permitió darle a la aplicación un ambiente enriquecido y mejorado.
- Para cubrir con el objetivo general del presente Trabajo Especial de Grado, se diseñó, desarrolló e implantó una plataforma Web genérica escalable, robusta y flexible con la cual se manejan de forma automatizada todos los procesos de una compañía aseguradora. Fue implementada bajo un patrón de diseño MVC modelo 2 y una arquitectura de tres capas. Al ser enfocada como aplicación Web, es independiente del sistema operativo que utilice la empresa.

CONCLUSIONES

La cantidad de tiempo y material de oficina que invierten hoy en día muchas empresas, incluyendo las compañías aseguradoras consultadas, para el control de sus procesos internos resulta cuantiosa en términos de pérdida monetaria. Sin embargo, es posible reducir gastos si se toman medidas como la redefinición de estos procesos para mantener un control más acertado sobre el mismo. No obstante para algunas empresas esta solución puede implicar gastos inclusive aún mayores.

La solución más viable a los problemas antes mencionados constituye la automatización de los procedimientos y procesos internos, enfocada a la minimización en los tiempos de respuesta al cliente, y a la disminución considerable del material de oficina empleado comúnmente para llevar el control de las actividades de la compañía. Solución propuesta en el presente Trabajo Especial de Grado.

En conversaciones con representantes de diversas compañías aseguradoras y con personal de la superintendencia de seguros, se logró concluir que todos estos procesos ortodoxos que se manejan en la actualidad pueden ser evitados o mejorados con los cambios que esta plataforma ofrece. Se demostró que los beneficiados al realizar estos cambios o las mejoras en todos los procesos de las compañías de seguro no son solo los clientes, sino también las mismas compañías quienes obtienen aumentos en la utilidad neta de la empresa en cuanto a tiempo ahorrado por sus empleados y dinero invertido en papelería, por mencionar algunos de los beneficios.

La gestión de perfiles de usuario y seguridad de datos en la plataforma, representan una característica fundamental en el desarrollo de aplicaciones, debido a que todas las actividades que los usuarios realizan están relacionadas con dichos perfiles, lo que permite controlar el acceso a

funcionalidades del sistema y establecer políticas de seguridad permitiendo consistencia e integridad de la información almacenada.

La plataforma implementada brinda a las compañías aseguradoras la posibilidad de manejar los distintos tipos de seguro, tipos de póliza y por supuesto los contratos en la forma en siempre los han manejado sin tener que hacer modificaciones sobre el sistema, ya que al ser configurable y adaptable a cualquier empresa, permite gestionar todos estos procesos de múltiples y mejores maneras, sin afectar sus procedimientos previos ni la forma en la que los manejaban. Por otra parte, el hecho de esta automatización trae múltiples beneficios, entre los que se mencionan:

- Mejora en la calidad del trabajo de los operadores y analistas.
- Reducción de costos, pues se racionaliza el trabajo
- Disminución en tiempo y dinero dedicado al mantenimiento.
- Se minimizan los tiempos de procesamiento de información
- Y además se brinda una gran flexibilidad para adaptarse a nuevos productos.

En cuanto a la metodología utilizada, RUP permitió de forma natural guiar el proceso de desarrollo del sistema a través ciertas iteraciones o ciclos, con la finalidad de proporcionarle un carácter evolutivo e incremental de manera que los productos obtenidos, conforme avanza el desarrollo, puedan ser evaluados por el cliente e inclusive por los usuarios finales, lo cual se traduce en una mayor comprensión de sus requerimientos y una labor mucho más acertada.

En relación a la implementación del conjunto de tecnologías englobadas bajo la técnica de AJAX, el mejoramiento y enriquecimiento de la aplicación Web es fácilmente notable lo que proporciona casi las mismas características y funcionalidades que las aplicaciones comunes de escritorio. Además de los beneficios de las aplicaciones del Modelo Web, el uso de esta

técnica reduce los tiempos de espera y dejan ver al usuario continuamente con lo que está trabajando y reaccionar a cualquier cambio, error, o actualización que la interfaz les notifique.

RECOMENDACIONES

Tomando en cuenta las conclusiones obtenidas, se presentan a continuación una serie de recomendaciones para un futuro incremento de las funcionalidades de la plataforma así como para posibles mejoras, de forma tal que estas puedan ser consideradas en futuros proyectos relacionados con el presente Trabajo Especial de Grado.

En primer lugar, para aumentar la funcionalidad del sistema, es recomendable crear los módulos encargados de administrar los restantes tipos de siniestros tal y como lo hace el módulo de siniestros de vehículos, para que de esta forma se puedan controlar todos los procesos bajo una misma aplicación. Estos deben ser creados bajo estrictas medidas de seguridad que mantengan la integridad de la información y basados en los estándares de programación utilizados durante todo el desarrollo.

Con el fin de aumentar el alcance de la aplicación desarrollada, podría agregarse un sistema experto que se encargue del proceso de análisis de los siniestros. La plataforma realiza una simple comparación de condiciones y le da al analista un resumen referente al incumplimiento o no de las condiciones que posee el contrato relacionado al siniestro. En reuniones con los especialistas de la materia se concluyó que el proceso de análisis mejoraría sustancialmente utilizando esta nueva característica que controlase el procesamiento de las condiciones, ya que se pueden obtener conclusiones y resolver problemas de forma más rápida que con los operadores y analistas. De esta forma se eliminaría la subjetividad que siempre esta presente en el proceso.

Para un mejor desempeño de los procesos referentes a siniestros de vehículos se recomienda establecer una comunicación a través de servicios Web con los sistemas de los proveedores de repuestos, para que de esta forma el sistema se comunique con los sistemas de los diversos proveedores

enviándoles a estos la solicitud del repuesto que se está buscando y la respuesta fuese enviada por la misma vía indicando la cantidad que tienen en inventario y el costo de los mismos. Y luego estos precios serían comparados automáticamente por el sistema para buscar el que más le conviene a la empresa. Esto ahorraría mucho tiempo que invierten las compañías aseguradoras en buscar vía telefónica proveedor por proveedor dónde consiguen el repuesto y luego realizan una comparación manual con todos los resultados obtenidos buscando el mejor precio. De igual forma, la misma técnica puede ser empleada con los módulos restantes tales como Siniestros de HCM con su respectiva conexión con los proveedores de servicios médicos.

Sería recomendable para la empresa proporcionar al dispositivo móvil utilizado por el perito o revisor, un servicio fijo de conexión dedicada de tal manera que la visita por parte del cliente al Centro de Servicio no sea totalmente necesaria ya que una vez declarado el siniestro, el Perito podría trasladarse directamente a la ubicación del cliente y realizar la revisión en la comodidad de su hogar u oficina, lo que se convertiría en una ventaja muy positiva para la empresa.

BIBLIOGRAFÍA

- [1] Trabajo de Investigación. *Seguros y Pólizas.*
  http://www.segurostp.8k.com/concepto_de_seguros.html
  Visitado: 7 de Febrero 2006
- [2] LOMA, Information that Works. *Seguros y sus principios.*
  http://www.loma.org/SP-PFSL-280.asp
  Visitado: 7 de Febrero 2006
- [3] Superintendencia de Seguros. *Diccionario de Seguros.*
  http://www.sudeseg.gov.ve/glos.php
  Visitado: 22 de Febrero 2006
- [4] DE LA CAMPA, Olga. *Léxico de Seguros.*
  Tercera Edición, 1991. Caracas, Venezuela
- [5] MATRÁN CASTELO, Julio; LOZANO GUARDIOLA, Antonio. *Diccionario MAPFRE de Seguros.*
  Editorial Mapfre, S.A. 1992. Madrid, España.
- [6] CUETO & LÓPEZ, Peritos de Seguros. *Diccionario del Seguro.*
  http://www.cueto-lopez.com/palabra.php
  Visitado: Martes 28 de Febrero 2006.
- [7] ÓSMOSIS LATINA. *Estructura (Framework) "MVC" ("Model - View - Controller").*
  http://javaweb.osmosislatina.com/curso/mvc.htm
  Visitado: 3 de Abril de 2006
- [8] ALTADILL, Pello Xavier. *Struts, implementación del patrón MVC en aplicaciones Web.*
  http://www.pello.info/?q=node/view/1#struts/struts.zip
- [9] SESHADRI, Govind. *Understanding JavaServer Pages Model 2 architecture.*
  http://www.javaworld.com/javaworld/jw-12-1999/jw-12-ssj-jspmvc.html
  Visitado: 10 Abril de 2006

- [10] O`REILLY. *Jakarta Struts. Mastering Java Web applications with courage*
  http://www.triviasecurity.net/library/Programming/OReilly%20-%20Jakarta%20Struts.pdf
  Visitado: 12 de Abril de 2006
- [11] VEGAS, Jesús. *Creación de un Portal Web Docente* ¨
  http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node20.html
  Visitado: 21 de Abril de 2006
- [12] GARRET, Jesse James. *Ajax: A New Approach to Web Applications.*
  http://adaptivepath.com/publications/essays/archives/000385.php
  Visitado: 2 de Marzo de 2006
- [13] Ajax Patters. *A Primer on the Ajax Phenomenon*
  http://ajaxpatterns.org/Whats_Ajax
  Visitado: 2 de Marzo de 2006
- [14] *Cliente Servidor*
  http://es.wikipedia.org/wiki/Cliente-servidor
  Visitado: 6 de Marzo de 2006
- [15] *Aplicación Web*
  http://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web
  Visitado: 10 de Marzo de 2006
- [16] Lenguajes Web
  http://www.desarrolloweb.com/articulos/709.php
  Visitado: 10 de Marzo de 2006
- [17] ALVAREZ, Miguel Ángel. *Lenguaje HTML*
  http://www.desarrolloweb.com/articulos/711.php
  Visitado: 10 de Marzo de 2006
- [18] VIAL S, Jean Michael. *Programación en PHP*
  http://www.aulambra.com/ver.asp?id=146
  Visitado: 10 de Marzo de 2006
- [19] BOEHM, Barry. *Anchoring the Software Process, IEEE Software* 4, July 1996, pp. 73-82.

- [20] IBM. *Rational Unified Process: Best Practices for Software Development Teams*. 1999.
  http://www-128.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf
  Visitado: 14 de Marzo de 2006
- [21] *Ciclo de Vida del Software*
  http://alarcos.inf-cr.uclm.es/doc/ISOFTWAREI/Tema03.pdf
  Visitado: 14 de Marzo de 2006
- [22] Desarrollo de Sistemas Informáticos Usando UML y RUP
  ftp://jano.unicauca.edu.co/cursos/Especializacion/ApliServicios/docs/UML-RUP-doc.pdf
  Visitado: 20 de Marzo de 2006
- [23] Universidad Rey Juan Carlos. Pruebas de Software.
  http://kybele.escet.urjc.es/Documentos/ISI/Pruebas%20de%20Software.pdf
  Visitado: 25 de Mayo de 2006
- [24] LARMAN, Craig. *UML y Patrones*.
  2º Edición. Prentice Hall. 2003.

GLOSARIO DE TÉRMINOS

A

- **ASP:** Active Server Pages, por sus siglas en ingles, son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el Web.

C

- **Ciclo de vida de un sistema:** Desde que se inicia la necesidad de creación de un sistema informático, hasta que se decide que esta obsoleto, los pasos posibles.
- **Cliente – Servidor:** Se le suele llamar así a la arquitectura a dos capas, es decir, una capa servidor que contendrá los datos y los programas gestores asociados, y capas clientes que se dirigirían al anterior para obtener la información.
- **Código Abierto:** Es el término por el que se conoce al software distribuido y desarrollado en forma libre. Este término empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (free software).
- **CSS:** Cascade Style Sheets, por sus siglas en inglés, son utilizadas para definir de manera formal y estructurada la presentación de un documento HTML. Permite tener un control centralizado de la presentación de un sitio Web completo con lo que se agiliza de forma considerable la actualización del mismo.

D

- **DHTML:** Dinamic HTML, son aplicaciones que contienen objetos y eventos y se procesan en el lado del cliente dentro del navegador Web. Designa el conjunto de técnicas que permiten crear sitios Web interactivos utilizando una combinación de lenguaje HTML estático.

H

- **Hardware:** Conjunto de componentes materiales de un sistema informático. Cada una de las partes físicas que forman un equipo informático, incluidos sus periféricos.
- **HTML:** HyperText Markup Language por sus siglas en ingles, es el lenguaje de marcado de Hipertexto. Es el lenguaje estándar para describir el contenido y la apariencia de las páginas en el www.
- **HTTP:** Es el protocolo de Internet que permite que los exploradores del WWW recuperen información de los servidores. Es un protocolo de aplicación con la sencillez y velocidad necesaria para sistemas de información distribuidos, colaborativos y de diferentes medios.

I

- **Interfaz de usuario:** Sistema de interacción entre la computadora y el usuario, caracterizado por la utilización de iconos y elementos gráficos en su concepción. Es un paso más allá de los interfaces basados en caracteres, que sólo incluían líneas de texto para introducir comandos y conocer las respuestas del sistema.

J

- **J2EE:** (Java 2 Enterprise Edition) define un estándar para el desarrollo de aplicaciones empresariales multicapa diseñado por Sun Microsystems. J2EE simplifica las aplicaciones empresariales basándolas en componentes modulares y estandarizados, proveyendo un completo conjunto de servicios a estos componentes, y manejando muchos de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja.
- **Java:** Lenguaje de programación orientado a objetos desarrollado por Sun Microsystems para la elaboración de aplicaciones exportables a la red y capaces de operar sobre cualquier plataforma a través, normalmente, de visualizadores WWW. El programa Java se descarga

desde el servidor Web y lo interpreta un programa que se ejecuta en el equipo que contiene el explorador de Web.

- **JavaBeans:** Los Enterprise JavaBeans (también conocidos por sus siglas EJB) son uno de las API que forman parte del estándar de construcción de aplicaciones empresariales J2EE de Sun Microsystems. Los EJBs proporcionan un modelo de componentes distribuido estándar para el lado del servidor. El objetivo de los Enterprise beans es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial (concurrencia, transacciones, persistencia, seguridad) para centrarse en el desarrollo de la lógica de negocio en sí. El hecho de estar basado en componentes nos permite que éstos sean flexibles y sobre todo reutilizables. Existen tres tipos: EJBs de Entidad, EJBs de Sesión y EJBs dirigidos por Mensajes.
- **JavaScript:** JavaScript no es un lenguaje de programación propiamente dicho. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto. Nunca podrás hacer un programa con JavaScript, tan sólo podrás mejorar tu página Web con algunas cosas sencillas
- **JSP:** Java Server Page por sus siglas en ingles, se refiere a un tipo especial de páginas HTML, en las cuales se insertan pequeños programas que corren sobre Internet (comúnmente denominados scripts), se procesan en línea para finalmente desplegar un resultado final al usuario en forma de HTML. Por lo general dichos programas hacen consultas a bases de datos y dependiendo del resultado que se despliegue será la información que se muestre a cada usuario de manera individual. Los archivos de este tipo llevan la extensión ".jsp".

K

- **Kernel:** Parte fundamental de un programa, por lo general de un sistema operativo, que reside en memoria todo el tiempo y que provee los servicios básicos. Es la parte del sistema operativo que está más cerca

de la máquina y puede activar el hardware directamente o unirse a otra capa de software que maneja el hardware.

M

- **MVC:** (Model-View-Controller) es un patrón de diseño que define la organización, independiente del Model (Objetos de Negocio), la View (interfaz con el usuario u otro sistema) y el Controller (controlador del workflow de la aplicación: "si estoy aquí y me piden esto entonces hacer tal cosa, si sale bien mostrar esto y sino lo aquello otro"),

O

- **Objeto:** Término base de la Programación Orientada a Objetos. En pura teoría podríamos pensar en lo que el hombre conoce, una mesa, un mechero, son cosas que nuestra inteligencia sabe comprender y explicar porque tienen unas características comunes, a pesar de que las diferencien muchas otras, pues hay formas y modelos muy dispares.
- **Orientado a objeto:** Es un sistema de programación, procedente de la evolución de la programación estructurada, cuyos pilares son la reutilización del código y la facilidad (teórica) de simplificarle. Su base es lo que se denomina Objeto y que es el fundamento de los lenguajes de este tipo

P

- **Patrón de diseño:** Son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces.
- **PHP:** Es un lenguaje de scripting embebido en HTML. Mucha de su sintaxis es tomada de C, Java y Perl con un par de características adicionales únicas y específicas de PHP. El propósito del lenguaje es permitir que los desarrolladores Web escriban páginas generadas dinámicamente con rapidez.

R

- **RUP:** El Proceso Unificado Racional, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

S

- **Servlet:** Aplicación sin interfaz gráfica que se ejecuta en un servidor de Internet, procesando información HTML previamente recogida por un navegador
- **Software:** El término inglés original define el concepto por oposición a hardware: blando-duro, en referencia a la intangibilidad de los programas y corporeidad de la máquina. Software es un término genérico que designa al conjunto de programas de distinto tipo (sistema operativo y aplicaciones diversas) que hacen posible operar con el ordenador.
- **Struts:** es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts. Permite reducir el tiempo de desarrollo, su carácter de "*software libre*" y su compatibilidad con todas las plataformas, en donde Java Entreprise está disponible, lo convierte en una herramienta altamente disponible.

X

- **XHTML:** Acrónimo inglés de eXtensible Hyper Text Markup Language (lenguaje extensible de marcado de hipertexto), es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas Web. XHTML es la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas, de XML. Su objetivo es avanzar en el proyecto del World Wide Web Consortium de lograr una Web semántica, donde la información, y la

forma de presentarla estén claramente separadas. En este sentido, XHTML serviría únicamente para transmitir la información que contiene un documento, dejando para hojas de estilo (como las hojas de estilo en cascada) y JavaScript su aspecto y diseño en distintos medios

- **XMLHttpRequest:** Es un conjunto de APIs que puede ser usado por lenguajes de programación a nivel cliente o servidor para transferir información codificada como XML o texto plano usando HTTP. La mayor ventaja de XMLHTPP es la habilidad de actualizar en tiempo real una página Web sin necesidad de cargarla nuevamente.
- **XSLT:** Extensible Stylesheet Language Transformation por sus siglas en ingles, es un lenguaje de transformación basado en hojas de estilo. Permite dar al XML un formato de salida comprensible para los humanos.