UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
CENTRO DE ENSEÑANZA ASISTIDA POR COMPUTADOR - CENEAC
CARACAS - VENEZUELA

DESARROLLO DE UNA APLICACIÓN WEB ORIENTADA A LA ADMINISTRACIÓN DE DOCUMENTOS Y DECLARACIÓN DE SINIESTROS EN EL ÁREA DE SEGUROS.

Trabajo Especial de Grado
Presentado ante la Ilustre
Universidad Central de Venezuela
por la Bachiller:
Milady Rosmar Sánchez Plaza. CI.- 16.332.846
para optar al Título de
Licenciada en Computación

Tutora:

Profa. Yusneyi Yasmira Carballo Barrera.

Caracas, mayo de 2011

Acta del Veredicto

Quien suscribe, miembros del Jurado designado por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado presentado por la Bachiller: **Milady Rosmar Sánchez Plaza CI. 16.332.846**, con el título **"Desarrollo de una aplicación Web orientada a la administración de documentos y declaración de siniestros en el área de seguros",** a los fines de optar al título de Licenciada en Computación, dejan constancia de lo siguiente:

Leído como fue, dicho trabajo por cada uno de los miembros del jurado, se fijó el día 26 de mayo de 2011 a las 4:30 p.m. para que su autor lo defendiera en forma pública, se hizo en el salón PBIII de la Escuela de Computación de la Facultad de Ciencias de la Universidad Central de Venezuela, mediante una presentación oral de su contenido, luego de lo cual respondieron las preguntas formuladas. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo con 19 puntos

En fe de lo cual se levanta la presente Acta, en Caracas a los veinte seis del mes de mayo del año 2011.

Profa. Vanessa Leguizamo

Prof. Antonio Silva

Profa. Yusneyi Carballo Barrera

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación

Titulo: Desarrollo de una Aplicación Orientada a la Administración de Documentos y Declaración de Siniestros en el Área de Seguros.

Autor: Br. Milady Rosmar Sánchez Plaza.
Unidad de Investigación: Aplicaciones con Tecnología Internet Optimización de Procesos en el área de Seguros.
Tutora: Profa. Yusneyi Yasmira Carballo Barrera.
Fecha de Presentación: 24 y 26 de mayo de 2011

Resumen

El presente Trabajo Especial de Grado tiene como objetivo general, desarrollar una aplicación Web para gestionar la declaración de siniestro y la administración efectiva de documentos asociados, tomando en cuenta los parámetros y requerimientos inherentes al sector de seguros, de la empresa de Seguros Carabobo. Dicha aplicación permite automatizar parte del proceso de declaración de siniestros que se lleva a cabo en dicho departamento. Para el caso de estudio se desarrolló un sistema en el que intervienen los clientes del seguro y parte del personal que en él labora. La idea del sistema es permitir que un cliente pueda informar vía Web un incidente con su vehículo, dicha información es gestionada por parte del personal del seguro quien se encarga de generar una respuesta asociada a la información dada por el cliente a través del sistema.

Para el desarrollo del sistema fueron utilizadas algunas tecnologías de desarrollo Web como: JavaScript, Ajax, jQuery, Apache, PHP, MySql y el Framework CakePHP. El sistema fue implementado siguiendo el patrón de diseño MVC (Modelo- Vista- Controlador) y guiado con la metodología de trabajo XP (programación extrema).

El desarrollo de este sistema permite ofrecer a sus clientes mejoras en el servicio de declaración de siniestro y la gestión de documentos, generando beneficios internos como la automatización de los procesos y mejoras en la calidad de servicio a los clientes.

Palabras Claves: Declaración de Siniestros, Gestión de Documentos y Aplicación Web en el área de Seguros.

Agradecimientos y Dedicatorias

Creo que la vida es lo más complicado en el mundo para cualquier persona, está llena de pruebas y retos, en la que los fuertes superan todas estas pruebas y retos, pero a cambio los débiles se dejan vencer por estas circunstancias. El éxito está en los que nunca se rinden y a pesar de todos los obstáculos que se nos pueden presentar en el camino, hay que derrumbarlos para lograr nuestros objetivos y llegar a una de las metas, porque una meta se cumple pero siempre debemos colocarnos otra, no quedarnos estancados en lo que ya hicimos debemos buscar siempre más para tener un crecimiento tanto personal como profesional.

Antes que nada le agradezco a Dios por darme la vida y hacerme la persona que hoy soy, ya que sin él no sería quién soy, debido a que me llena de esperanza y fortaleza para seguir por el camino correcto.

A mis padres Pedro Sánchez y Rosa Plaza por ser las personas más importantes de mi vida, les doy gracias por haberme guiado y hacerme ver que debía ser alguien importante en la vida, apoyarme y animarme en todo momento, a pesar de todos los obstáculos que se presentaron para lograr mi meta. A Daisy Plaza y Carlos Palacios por estar conmigo en todo momento, apoyarme y ser mis paños de lágrimas, cuando me dieron mis altibajos que no iba a poder lograr mi objetivo.

Y a mi Tutora Yusneyi Carballo Barrera por aceptarme como tesista y guiarme para lograr la culminación de la carrera de Computación.

Les doy las Gracias por el apoyo.

Milady Rosmar Sánchez Plaza.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Con los continuos avances tecnológicos y las demandas de las empresas por estar al día con dichos avances, cada vez se hace más frecuente que las empresas se vean en la necesidad de automatizar en la medida de lo posible gran parte de sus procesos operativos, con el fin de minimizar costos, tiempo de respuestas e incrementar la calidad de servicios y satisfacer las demandas. Actualmente con la gran oferta y demanda de vehículos existentes en nuestro país, las empresas de seguros ofrecen a sus clientes pólizas de vehículos con el fin de satisfacer las necesidades demandadas por éstos, teniendo que contar para ellos con el recurso humano que dé respuestas rápidas y eficientes al momento en que los clientes requieran hacer uso de las coberturas ofrecidas en dichas pólizas.

Actualmente los clientes buscan obtener respuestas a sus requerimientos de la manera más cómoda y sencilla para ellos, sin importar los efectos positivos o negativos que estos causen en las empresas. Para el caso de estudio en particular, la empresa de seguros “Seguros Carabobo” en adelante denominada Empresa de Seguros, cuenta con un Departamento de Siniestros donde son reportados los daños en vehículos por parte de los clientes. En éste departamento se realizó un levantamiento de información y se pudo constatar que estos reportes de clientes son llevados de forma manual, por lo que la empresa requiere de una automatización para el departamento.

Con la automatización se le permitirá al cliente reportar los incidentes del vehículo a través de una aplicación Web, evitando de esta manera que éste se dirija a las oficinas del seguro para realizar dichos reportes. Con ello la Empresa de Seguros logrará ofrecer al cliente un servicio de autogestión. A través de la aplicación Web el cliente podrá declarar los daños del vehículo mediante una de las funcionalidades (Declaración de Siniestros) a desarrollar en la aplicación, una vez realizada la declaración éste podrá consultar el estatus (pago negado, pago realizado, pendiente por pago, pendiente por recaudos) de la misma. Luego que el cliente realicé la notificación de los daños se incorporará el módulo de Gestión de Documentos, en el que se tendrán las copias de los documentos (cédula de identidad, certificado médico, licencia de conducir y carnet de circulación del vehículo) exigidos por la Ley de Seguros establecida por la Superintendencia de Seguros y que cada cliente deberá tener en su expediente. Por otra parte éste módulo le permitirá a los empleados (Analista y Perito, para el caso en estudio) del seguro consultar; qué clientes han reportado más siniestros por vehículo, cuáles son las partes más frecuentemente afectadas en los

vehículos y obtener la cantidad de siniestros reportados por los clientes en distintos períodos de tiempo (diario, semanal y mensual).

A continuación se detallará la estructura de los capítulos que conforman el Trabajo Especial de Grado:

Capítulo I: Problema de Investigación: en éste capítulo se expone la forma en la que se llevan a cabo actualmente el proceso de declaración de siniestros dentro de un ente de Seguros de auto, se especifica el problema a resolver y la solución más apropiada para el mismo, así como los objetivos del trabajo, su alcance, importancia y justificación del desarrollo, evaluando los beneficios que generará el mismo.

Capítulo II: Marco Conceptual: se presentan los fundamentos conceptuales que sustentan el presente trabajo de investigación dando a conocer lo relacionado con aplicaciones Web. Describiendo su arquitectura y el funcionamiento de las mismas. Adicionalmente, se darán a conocer las principales tecnologías y herramientas utilizadas para el desarrollo de aplicaciones Web.

Capítulo III: Marco Aplicativo: se presenta el desarrollo del prototipo bajo la adaptación de la metodología XP, detallando las fases de Planificación, Diseño, Codificación y Pruebas dentro de cada una de las iteraciones definidas para dicho proceso. También se describe el uso del modelo MVC y las principales tecnologías utilizadas.

Finalmente se proveerán Conclusiones con respecto al trabajo realizado, específicamente orientadas hacia el análisis del alcance de los objetivos planteados, se suministrarán Recomendaciones para la evolución del funcionamiento de la aplicación, y se listarán las Referencias Bibliográficas y Referencias Web consultadas.

Capítulo I: Problema de Investigación

En éste capítulo se describe la problemática que enfrenta el departamento de siniestro de automóvil en la Declaración de Siniestros y la Gestión de Documentos. Actualmente la Empresa de Seguros se encuentra en la automatización de los procesos de dicho departamento para implementar nuevos lineamentos que beneficien tanto a los clientes como al personal que labora en dicha institución.

Luego se destaca el objetivo general, los objetivos específicos y el alcance de este Trabajo Especial de Grado, enfocando las mejoras que se quieren aportar a los procesos actuales de la empresa y los beneficios que van a traer a los grupos involucrados.

1.1 Situación Actual

Hoy en día son muchas las actividades que se desarrollan en el ámbito laboral, institucional, personal, etc., además que cada día se hace necesario tener procesos que permitan administrar de mejor manera el tiempo, las actividades y los recursos con los que se cuenta.

En la mayoría de las instituciones existe la necesidad de asignar y coordinar los recursos económicos, materiales y humanos de forma óptima. Además en algunos casos se necesita automatizar algunos de sus procesos para que se puedan realizar de una manera eficaz y rápida, así como ahorrar tiempo y dinero. El manejo de la información asociada al desarrollo de estos procesos es un aspecto de sumo interés, por lo que el estudio e implementación de estrategias y herramientas para una gestión eficiente de la información es de vital importancia. Esto se quiere llevar a cabo dentro de la Empresa de Seguros debido a que en este momento la empresa lleva los casos de siniestros en forma manual.

Por esta problemática que enfrenta no podrá obtener el alto nivel de gestión y capacidad que presentan los departamentos que la conforman, más aún si hablamos de una institución aseguradora en la que el cliente deposita su confianza para asegurar su vehículo, cuya misión está orientada a la satisfacción rápida y oportuna de los problemas de sus clientes. Esto nos lleva a determinar que el centro del problema a plantear está en cómo se establece la comunicación entre los diferentes departamentos involucrados. El análisis debe entonces centrarse en determinar cuáles son las razones que llevan a retrasos y equivocaciones, a la hora de procesar la información en los expedientes de los clientes,

originando que no se le proporcione al cliente una respuesta a sus solicitudes en un lapso de tiempo estipulado.

1.2 Planteamiento del Problema

Actualmente los asegurados que tienen pólizas de vehículos con Empresa de Seguros, cuentan con dos maneras de declarar el siniestro, una de ellas es presencial, en la que el asegurado debe asistir a las oficinas de atención al cliente y la otra es por vía telefónica mediante el Call Center, donde un analista carga la información suministrada por el asegurado.

A continuación se mostrará el Workflow que se lleva a cabo dentro del departamento de siniestros.

En la actualidad la Empresa de Seguros, está dedicada al mercado asegurador venezolano, realizando su proceso de declaración de siniestros de vehículo a través de la oficina de caja de la Región Capital de forma manual de la siguiente manera: el **cliente asegurado** debe llenar una planilla de declaración de siniestro, encontrándose como primera debilidad el error humano en la transcripción de los datos, adicionalmente muchas veces la información solicitada al asegurado no es suficientemente clara para el **analista**. Posteriormente el asegurado reúne los recaudos que le son solicitados para procesar su solicitud de cobertura y los entrega al analista de seguros, quien podría indicarle que debe llenar alguna información nuevamente debido a enmendaduras y/o tachaduras que el asegurado haya realizado. La declaración de siniestros también se puede realizar vía telefónica, trayendo inconvenientes al cliente tales como: fallas de audición en la comunicación, poco entendimiento de la información que se le suministra y demoras en esperas continúas al momento en que la persona que atienda su llamada requiera verificar alguna información. Luego del llenado de la planilla de declaración de siniestro por parte del cliente asegurado, el **analista** envía una orden de inspección al perito para que éste realice la evaluación del vehículo y proporcione con detalles el resultado de la evaluación. El análisis debe entonces centrarse en determinar cuáles son las razones que causan retrasos y errores a la hora de procesar la información plasmada en las planillas de declaración de siniestro.

Mediante la investigación se ha podido constatar que el proceso manual que implementa la empresa se ha mostrado ineficiente, debido a que la búsqueda y archivo de

los expedientes de los clientes asegurados es bastante engorrosa y en muchos casos los documentos se extravían, se puede decir que de un 100% el 30% se pueden extraviar, generando retrasos y trayendo como consecuencia disgustos por parte de los clientes, quienes en repetidas oportunidades se apersonan en las oficinas de atención al cliente o llaman a la central telefónica para exigir respuestas de sus requerimientos, de un 50% de 100% se puede decir de las personas que se molestan por no tener la respuesta en sus manos. Al mismo tiempo la institución se muestra en desventaja ante otras que utilizan sistemas informáticos con tecnologías modernas, permitiéndose estas últimas ofrecer servicios más óptimos a sus clientes y a todos los grupos involucrados.

En la Figura Nº 1 se puede observar cada uno de los actores que conforman la Declaración de Siniestro en un alto nivel. Analista de Siniestros, Perito, Gerente y el Cliente, cada uno de estos actores interactúan con el sistema, pero en la actualidad el cliente no tiene interacción con el departamento de siniestros, por lo que esta eventualidad será el caso de estudio a desarrollar en el presente Trabajo Especial de Grado.

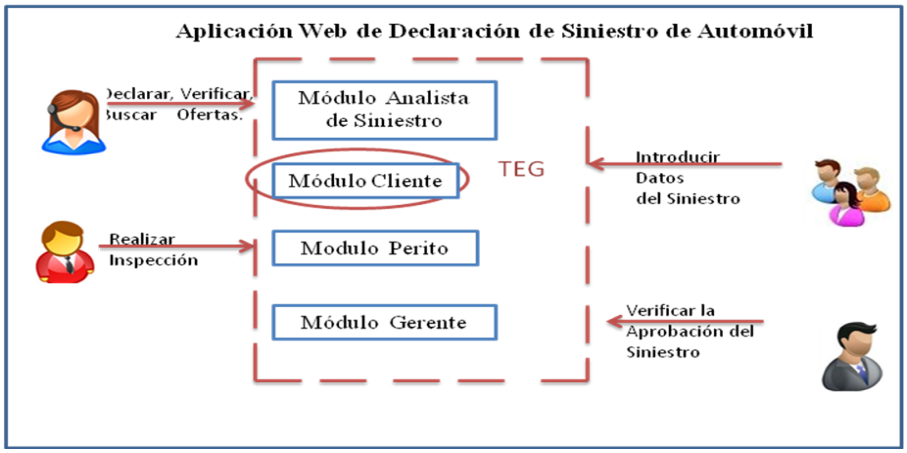


Figura 1. Modelo General de Sistema de Declaración de Siniestro de Vehículo. Fuente: Elaboración Propia (2010).

A continuación se describen las funcionalidades comprendidas por cada actor que interactúa con el sistema:

Analista de Siniestro: permite que el cliente realice la declaración de siniestro, notificando los hechos ocurridos. La responsabilidad de este departamento es formalizar dicha declaración (se verifica que el siniestro se esté reportando antes de los 5 días después de haber sucedido el daño ya que si pasan más días el mismo es rechazado, pero de igual

forma se toma el siniestro), verificar la póliza si se encuentra activa, solicitar los recaudos legales, actualizar los datos si el cliente no lo quiere realizar vía Web, cotizar (repuestos, talleres, auto, entre otras), generar la orden de reparación.

Cliente: debe dirigirse personalmente a las oficinas de la Empresa de Seguros para realizar la declaración del siniestro, ese mismo día puede realizar la inspección del vehículo o tiene un plazo no mayor a 15 días hábiles para efectuarla. Luego que dicha información es tratada por el analista y por el perito, quienes aprobaran o no la orden de reparación del vehículo, el cliente deberá regresar a las oficinas de la Empresa de Seguros para buscar la orden de reparación en caso de que esta haya sido aprobada.

Perito: luego que la declaración fue reportada ante el analista de siniestros, el cliente debe dirigirse personalmente con su vehículo a la inspección. El perito compara la información indicada en el reporte de la declaración realizada por el cliente con la inspección del vehículo, en dicho reporte son anexadas las fotografías que son tomadas por el perito para tener pruebas de los daños, se realiza el levantamiento de información del caso. Si la información detallada por parte del perito coincide con la información que el cliente le indico al analista, se procede a la elaboración de la orden de reparación del vehículo.

Gerente: permite la supervisión de los casos para que sean atendidos en el tiempo correcto,

1.3 Solución Propuesta

Tomando en cuenta la problemática descrita en el punto anterior, se propone desarrollar una aplicación en la que los clientes asegurados que tengan pólizas de vehículos vigentes, puedan realizar la declaración de siniestros de vehículos a traves de la Web. Esto producirá un impacto significativo debido a que dicha aplicación facilitará la gestión de la declaración de siniestro por el mismo asegurado, disminuirá las llamadas realizadas al Call Center y se evitarán dirigirse a las oficinas del seguro. A su vez la aplicación le permitirá al cliente estar informado sobre el estatus del siniestro, con la luego de que el cliente sepa el estatus podrá retirar la orden de reparación del vehículo en las oficinas del seguro.

Por otra parte, la aplicación contará con un módulo Gestión de Documentos que permitirá a la Empresa de Seguros automatizar el proceso de los documentos que se manejan por cada cliente asegurado. Además, los analistas y peritos del departamento de siniestro podrán ver los reportes de los siniestros que hayan ocurrido en detrminados

espacios de tiempo (diario, semanal y mensual), realizar distintas consultas como por ejemplo: cuáles son las partes del vehiculo que son afectadas frecuentemente, los clientes que reportan mayor cantidad de sisniestros, entre otras.

A continuación se detallarán las tecnologías a implementar en el desarrollo de la aplicación Web.

En la Figura N° 4 se muestran las tecnologías del lado del cliente y del servidor que se utilizarán para desarrollar los módulos de Gestión de Documentos y Declaración de Siniestros. En el Capítulo II se explicarán detalladamente, cuáles son sus características sus ventajas y desventajas al momento de utilizarlas.

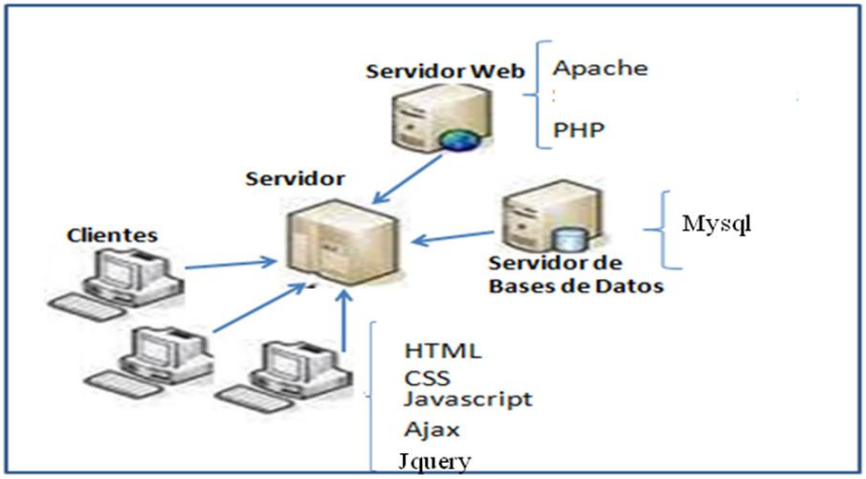


Figura 3. Arquitectura Cliente-Servidor. Fuente: Elaboración Propia (2010).

En específico se plantea incluir en la solución propuesta el uso de las siguientes tecnologías del lado del cliente.

- HTML 4.01 se usará por ser una de las versiones más recientes, además de ser uno de los formatos de documentos más utilizados en el área Web para el despliegue de contenido.
- CSS 2.0 para generar los estilos lógicos y físicos de los documentos HTML.
- JavaScript 1.3 para poder realizar filtrado y validaciones de formularios.
- Ajax para poder realizar consultas de manera asíncrona con el servidor.
- Jquery 1.4.3 es una biblioteca o framework de JavaScript, que permite simplificar la manera de interactuar con los documentos de HTML, permite manejar eventos,

desarrollar animaciones y agregar interacción con la tecnología AJAX, también es software libre de código abierto.

Del lado del servidor de Base de Datos se utilizará:

- Base de datos MySql 5.5.8 es un sistema manejador de bases de datos relacional, multihilo y multiusuario, es un software libre en un esquema de licenciamiento dual, existen varias APIs que permiten a las aplicaciones escritas en diversos lenguajes de programación, acceder a las bases de datos MySql incluyendo PHP.

Del lado del servidor Web:

- Apache 2.2.17 la aplicación se ejecutara bajo este servidor Web HTTP de código abierto por ser uno de los más estable en aplicaciones Web bajo PHP y garantiza la alta disponibilidad y el buen desempeño de la misma atendiendo las numerosas peticiones hechas por sus diversos usuarios.
- PHP 5.3.5 es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas Web dinámicas. Es usado principalmente para la interpretación del lado del servidor (*server-side-scripting*).

1.4 Objetivos

A continuación se presenta el objetivo general de esta investigación, que se deriva de un conjunto de objetivos específicos que permitirán cumplir con el objetivo general.

1.4.1 Objetivo General

Desarrollar una aplicación Web para la Gestión de Declaración de Siniestros y la Administración efectiva de documentos asociados, tomando en cuenta parámetros y requerimientos inherentes al sector de Seguros.

1.4.2 Objetivos Específicos

A fin de lograr el cumplimiento del objetivo general se plantearon los siguientes objetivos específicos:

1. Realizar el levantamiento de información correspondiente al proceso de Declaración de Siniestros y Gestión de Documentos.
2. Diseñar la interfaz gráfica del módulo de Declaración de Siniestro y de la Gestión de Documentos.
3. Diseñar la base de datos donde se guardará la información de la declaración de siniestros de automóvil de Seguros.
4. Aplicar la metodología de desarrollo XP.
5. Construir un conjunto de pruebas que permitan verificar el correcto funcionamiento de la aplicación, adaptándose a los requerimientos previamente identificados.

1.5 Meta y Alcances

Con el desarrollo de la aplicación se quiere lograr satisfacer la demanda de un cliente al momento en que a éste se le presente una eventualidad con su vehículo, tener a su disposición todos los servicios que ofrece Empresa de Seguros de la forma más eficiente posible, logrando siempre la satisfacción del cliente. El alcance que tiene este Trabajo Especial de Grado es la optimización del proceso de declaración de siniestros vía Web, para lo que se desarrollará el módulo Declaración de Siniestros. En este módulo el cliente asegurado podrá realizar por sus propios medios y sin intermediarios el levantamiento de información del siniestro. Igualmente se creará una opción en la que el cliente podrá consultar el estatus en que se encuentra el siniestro.

El módulo antes mencionado se encuentra ligado al módulo de Gestión de Documentos, el cuál le permitirá al personal (Analista y Perito) que labora en Empresa de Seguros gestionar el proceso de documentación, consultar las estadísticas de los siniestros registrados por los clientes asegurados, visualizar las partes del vehículo más afectadas en los siniestros y los clientes que presentan mayor número de siniestros. Estas serán unas de las funcionalidades a desarrollar en la aplicación Web.

1.6 Importancia y Justificación

Uno de los principales beneficios para la empresa es la automatización y el control de la Gestión de Declaración de Siniestros y la Administración de Documentos, esto conlleva a la eficiencia y eficacia en los procesos de negocio interno y externo, además juega un papel muy importante ya que determina el éxito económico de la misma.

La idea de desarrollar un sistema Web que permita controlar la declaración de siniestros en vehículos y el manejo de la documentación asociada a dichos siniestros, surge debido a que se realizaron consultas en clientes asegurados al momento de éstos reportar un incidente o siniestro con su vehículo en cualquiera de las dos modalidades (vía telefónica y presencial) ofrecidas por Empresa de Seguros y los resultados obtenidos fueron en su gran mayoría quejas y reclamos por parte de éstos, debido al tiempo que deben esperar para recibir de parte de Empresa de Seguros una respuesta que satisfaga sus demandas. Es por ello que para subsanar esta problemática y mejorar el proceso llevado a cabo por la empresa, se planteo realizar un sistema Web que permita al cliente asegurado realizar la declaración de un siniestro en línea sin tener que establecer contacto directo vía telefónica o presencial con personal de Empresa de Seguros.

Capítulo II: Marco Conceptual

Este capítulo presenta los fundamentos conceptuales que permitirá el desarrollo de la aplicación del Trabajo Especial de Grado (TEG), el mismo se encuentra dividido en cuatros secciones fundamentales. La primera parte corresponde a la definición de aplicaciones Web, características, ventajas y desventajas, entre otros. Luego se explica la estructura de la arquitectura en tres capas MVC basadas en el modelo Cliente – Servidor, haciendo énfasis de lo más importante de la misma; finalmente se especificará la arquitectura Web de tres capas, donde se expone el patrón de arquitectura MVC.

Una vez dado a conocer los aspectos de las aplicaciones Web y su arquitectura. La segunda parte de este capítulo describe las tecnologías del lado del cliente como *HTML* (*HypertText Markup Language*), *JavaScript, CSS* (*Cascading Style Sheets*), *AJAX* (*Asynchronous JavaScript And XML*) y jQuery; así como también conoceremos las características y ventajas de las tecnologías del lado del servidor, enfocándonos específicamente en PHP (*Hypertext Pre-Processor*), en el servidor de aplicaciones Apache, tomando en cuenta su arquitectura y funcionamiento, para luego estudiar las características, ventajas y desventajas, entre otros aspectos, del sistema manejador de base de datos relacional MySql y estudiaremos el framework CakePHP que permite programar y diseñar páginas Web. También se contemplará las Bases Metodológicas para el desarrollo Web.

2.1 Análisis y Comparación de Tecnologías

Con la llegada de Internet y de la Web en concreto, se han abierto muchas posibilidades en cuanto al acceso a la información desde cualquier sitio. Se emplean tecnologías como Java, PHP, JavaScript, Ajax, entre otras que dan potencia a la interfaz de usuario.

Las aplicaciones Web permiten la generación automática de contenido, la creación de páginas personalizadas según el perfil del usuario o el desarrollo del comercio electrónico. Además, una aplicación Web permite interactuar con los sistemas informáticos de gestión de una empresa. [1] [2]

Afortunadamente, tenemos herramientas potentes para realizar esto, ya que han surgido nuevas tecnologías que permiten el acceso a la base de datos desde la Web. El problema es decidir entre el conjunto de posibilidades, cuál tecnologías escoger en este

caso el desarrollo de la aplicación se enfocara en la tecnología que es utilizada dentro de la Empresa de Seguros.

El viejo *CGI* ha cumplido con el propósito de añadir operabilidad a las páginas Web pero sus deficiencias en el desarrollo de aplicaciones y en la escalabilidad de las mismas ha conducido al desarrollo de *API (Application Programming Interface)*, específicos de servidor como *ASP (Active Server Pages) y PHP (Hypertext Pre-Processor),* que son más eficientes que su predecesor *CGI*.

Una aplicación Web [3] [4] es un sistema informático que los usuarios utilizan accediendo a un servidor Web a través de Internet. Las aplicaciones Web son populares, debido a la practicidad del navegador Web como cliente ligero. La habilidad para actualizar y mantener aplicaciones Web sin distribuir e instalar software en miles de potenciales clientes es otra razón de su popularidad.

En contraste, las aplicaciones Web generan dinámicamente una serie de páginas en un formato estándar, soportado por navegadores Web comunes como HTML o XHTML. Se utilizan lenguajes interpretados del lado del cliente, para añadir elementos dinámicos a la interfaz de usuario. Generalmente cada página Web individual es enviada al cliente como un documento estático, pero la secuencia de páginas provee de una experiencia interactiva.

2.1.1 Arquitectura Cliente – Servidor

Sabemos que la arquitectura es la capacidad que tenemos de diseñar o construir, pero en el entorno Web como se puede ver esto; sin tener claro que es un cliente y un servidor en el entorno Web. Es por ello que se define y describe estos dos términos en el ambiente Web. Desde el punto funcional, se puede definir como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma.

Cliente es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajos a través de redes LAN (*Local Area Network*) o WAN (*Wide Area Network*). La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente. El cliente es el proceso que permite al usuario formular los requerimientos y pasarlos al servidor, se le conoce con el término *Front - End*. [5] [6]

Las funciones que lleva a cabo el proceso cliente se resumen en los siguientes puntos:

1. Administrar la interfaz de usuario.
2. Interactuar con el usuario.
3. Procesar la lógica de la aplicación y hacer validaciones locales.
4. Generar requerimientos de bases de datos.
5. Recibir resultados del servidor.
6. Formatear resultados.

Servidor es el encargado de atender a múltiples clientes que hacen peticiones de algún recurso administrado por él. Al proceso servidor se le conoce con el término *Back – End.* El servidor normalmente maneja todas las funciones relacionadas con la mayoría de las reglas del negocio y los recursos de datos. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs, para proveer de múltiples servicios a los clientes y ciudadanos tales como impresión, acceso a bases de datos, fax, procesamiento de imágines, entre otros. [5] [6]

Las funciones que lleva a cabo el proceso servidor se resumen en los siguientes puntos:

1. Aceptar los requerimientos de bases de datos que hacen los clientes.
2. Procesar requerimientos de bases de datos.
3. Formatear datos para transmitirlos a los clientes.
4. Procesar la lógica de la aplicación y realizar validaciones a nivel de bases de datos.
5. Gestión de periféricos compartidos.
6. Control de accesos recurrentes a bases de datos compartidas.
7. Enlaces de comunicaciones con otras redes de área local o extensa.

Una vez entendido estos dos conceptos podemos decir que la arquitectura "*Cliente - Servidor*" consiste en un servidor que proporciona uno o más servicios a uno o más clientes que desea acceder a dicho servicio a través de una petición, generada por el cliente.

Características de la arquitectura Cliente – Servidor

Las características básicas de una arquitectura Cliente – Servidor son: [6]

- Combinación de un cliente que interactúan con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputos como velocidad del procesador, memoria, velocidad y capacidades del disco *Input – Output Devices*.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basada en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que éstos los que hacen peticiones se servicios a los servidores. Estos últimos tienen un carácter pasivo ya que se esperan las peticiones de los clientes.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente/Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

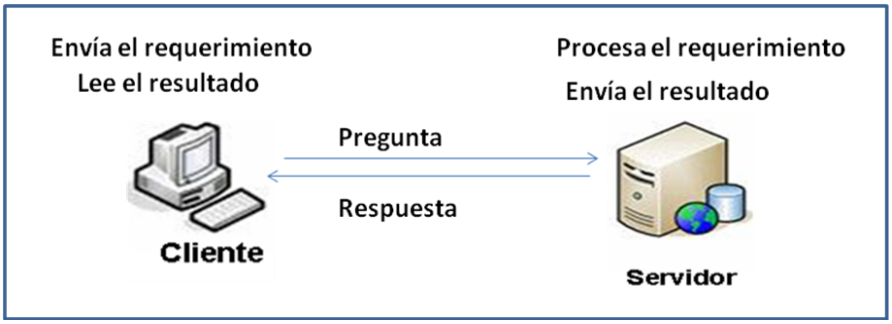


Figura 2. Arquitectura Cliente – Servidor. Fuente: Catarina Márquez (2008).

Como muestra la Figura 6, en esta arquitectura las computadoras de cada uno de los usuarios, llamada **Cliente**, son los encargados de activar la comunicación con el servidor, donde inician un proceso de diálogo; cada uno de estos realiza una acción que produce una petición de alguna información (correo, archivo, imagen, etc.) o solicita recursos (impresora, unidad de disco, etc.). La computadora que recibe la solicitud o petición ejecutada por el

cliente, se conoce como **Servidor**. Este tiene la potestad de atender dicha petición para dar como resultado algún tipo de respuesta al cliente. [6]

Se puede decir que la arquitectura Cliente – Servidor es la integración distribuida de un sistema de red, con los recursos, medios y aplicaciones que definidos modularmente en los servidores; estos administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información; estableciendo así un enlace de comunicación transparente entre los elementos que conforman la estructura.

Existen diversos tipos de servidores, que se clasifican basándose en su funcionalidad; estos son denominados servidores dedicados ya que administran el uso de algún recurso en particular, por ejemplo:

- **Servidores de archivo:** servidor donde se almacena archivos y aplicaciones de productividad como por ejemplo procesadores de texto, hoja de cálculo, etc.
- **Servidores de Bases de Datos:** servidor donde se almacenan las bases de datos, tablas, índices. Es uno de los servidores que más carga tienen.
- **Servidores de Transacciones:** servidor que cumple o procesa todas las transacciones. Valida primero y recién genera un pedido al servidor de bases de datos.
- **Servidores Web:** sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red.
- **Servidores de Correo:** mueven y almacenan el correo electrónico a través de las redes corporativas.
- **Servidores de Aplicaciones:** son aquellos que comunican distintas aplicaciones.
- **Servidores de Impresoras:** son aquellos que llevan la administración de un conjunto de impresoras.

2.1.2 Arquitectura de Tres Capas

La estrategia tradicional de utilizar aplicaciones compactas causa gran cantidad de problemas de integración en sistemas de software complejos como pueden ser los sistemas de gestión de una empresa o los sistemas de información integrados consistentes en más de una aplicación. Estas aplicaciones suelen encontrarse con importantes problemas de escalabilidad, disponibilidad, seguridad e integración. [7] [8]

Para solventar estos problemas se ha generado la división de las aplicaciones en capas que normalmente serán tres: una capa que servirán para guardar los datos (bases de datos), una capa para centralizar la lógica de negocio (modelo) y por último una interfaz gráfica que facilite al usuario el uso del sistema. Esto permite distribuir el trabajo de creación de una aplicación por niveles; de este modo cada grupo de trabajo está totalmente abstraído del resto de niveles, de forma que basta con conocer la API que existe entre niveles.

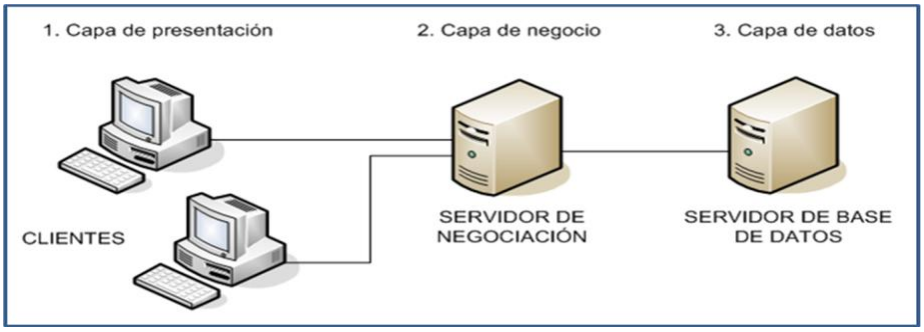


Figura 3. Arquitectura de Tres Capas. Fuente: Master Will (2003)

A continuación se describirá cada una de las capas de la arquitectura como se puede ver detalladamente en la Figura 6, donde se muestra los componentes de la arquitectura de tres (3) capas, donde la capa de presentación son las estaciones de trabajo (Clientes) las cuales son los que inician las peticiones, la capa de negocio (Servidor de Negociación) es la encargada de procesar la solicitud y la capa de datos (Servidor de Bases de Datos) es la que respalda toda la información o dato de un sistema.

Capa de Presentación: es la que ve el usuario, presenta el sistema al usuario, le comunica la información y captura la información del usuario dando un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio.

Capa de Negocio: es donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) pues 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 solicitar al gestor de bases de datos almacenar o recuperar datos.

Capa de Datos: es donde residen los datos. Está formada por uno o más gestor de bases de datos que realiza todo el almacenamiento de datos, reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

La ventaja principal de este estilo, es que el desarrollo se puede llevar a cabo en varios niveles y en caso de algún cambio sólo se ataca al nivel requerido sin tener que revisar entre código mezclado. Además permite distribuir el trabajo de creación de una aplicación por niveles, de este modo, cada grupo de trabajo está totalmente abstraído del resto de niveles, simplemente es necesario conocer la API que existe entre niveles.

En la actualidad el diseño de sistemas informáticos se suele usar la programación por capas. En dichas arquitecturas a cada nivel se le confía una misión simple, lo que permite el diseño de arquitecturas escalables (que pueden ampliarse con facilidad en caso de que las necesidades aumenten).

2.1.3 Patrón Arquitectónico: Modelo – Vista – Controlador (MVC)

Modelo Vista Controlador (MVC) [8] es un patrón de arquitectura aportado originalmente por el lenguaje *SmallTalk* a la ingeniería del software. El paradigma MVC consiste en dividir las aplicaciones en tres partes:

1. Modelo
2. Vista
3. Controlador

El **modelo** es el elemento de la aplicación que responde a una petición. Es la representación específica del dominio de la información sobre la que funciona la aplicación. La lógica de dominio añade significado a los datos. Representa las estructuras de datos, típicamente el modelo de clases contendrá funciones para consultar, insertar y actualizar información de la base de datos. [8][9]

La **vista** presenta el modelo en un formato adecuado para interactuar, usualmente un elemento de interfaz de usuario. Una vez realizadas las operaciones necesarias el flujo vuelve al controlador y este devuelve los resultados a una vista asignada. En pocas palabras es la información presentada al usuario. Una vista puede ser una página Web o una parte de una página. [8][9]

El **controlador** responde a eventos, usualmente acciones del usuario, e invoca cambios en el modelo y probablemente en la vista. Es el encargado de redirigir o asignar una aplicación (un modelo) a cada petición; el controlador debe poseer de algún modo, un "mapa" de correspondencias entre peticiones y respuestas (aplicaciones o modelo) que se les asignan. El controlador actúa como intermediario entre el Modelo, la Vista y cualquier otro recurso necesario para generar una página. [8][9]

Muchos de los sistemas informáticos utilizan un Sistema de Gestión de Bases de Datos para gestionar los datos: en líneas generales del MVC corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre Vista y su correspondencia Controlador de eventos y accesos a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí. [8]

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

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.).
- El controlador recibe (por parte de los objetos de la interfaz- vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (*handler)* o *callback*. El controlador accede al modelo, actualizando, posiblemente modificando la forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- El controlador delega a los objetos de las vistas la tarea de desplegar la interfaz de usuario. La vista obtiene los datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre las vistas. Sin embargo, se podría utilizar el patrón Observador (*Observer*, del patrón de diseño) para proveer cierta indirección entre

el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos al dominio (el modelo) a las vistas aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones las vistas no tiene acceso directo al modelo, dejando que el controlador envía los datos del modelo a la vista.

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

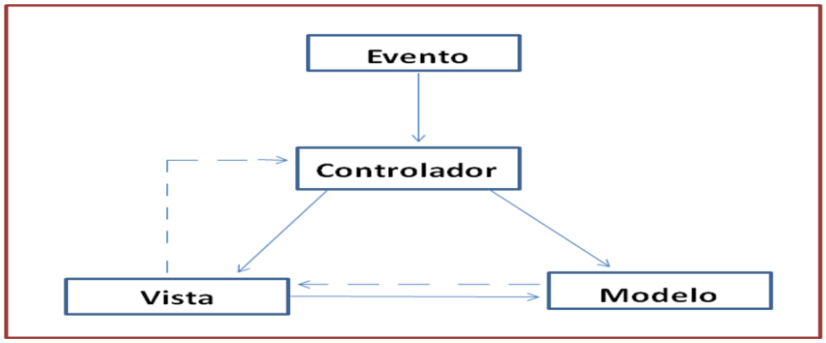


Figura 4. Patrón MVC. Fuente: JM Pérez (2007).

La Figura 7 modela un diagrama sencillo que muestra la relación entre el **modelo, vista y el controlador**. Donde se debe disparar un evento del *controlador* que recibe la petición y decide qué hacer, este a su vez invoca al *modelo* para que procese la solicitud y busque o actualice la información data; una vez hecho esto el controlador decide a que vista va a direccionar y el modelo envía la data solicitada a la vista para su representación. Nota: las líneas sólidas indican una asociación directa, y las punteadas una indirecta. La Figura 8, representa el diagrama de secuencia de la arquitectura MVC, aunque existen muchas implementaciones. [10]

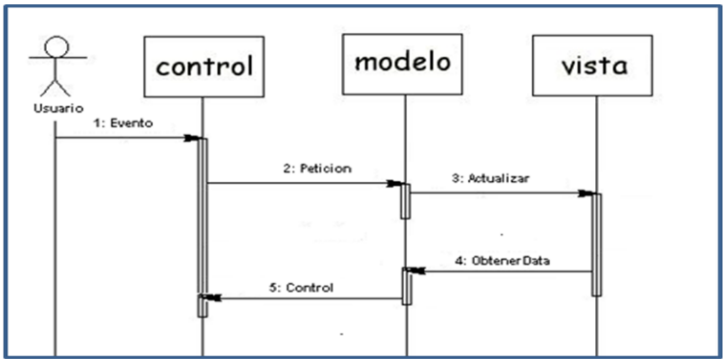


Figura 5. Diagrama de Secuencia de MVC. Fuente: JM Pérez (2007).

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control que sigue generalmente es el siguiente: [10]

- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.).
- El controlador recibe (por parte de los objetos de la interfaz vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario. Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
- El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo. El modelo no debe tener conocimiento directo sobre la vista. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. El controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. 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 de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Desarrollar una aplicación siguiendo este patrón de diseño tiene muchas ventajas: [11]

- La aplicación esta implementada modularmente.
- Sus vistas muestran información actualizada siempre.
- El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el modelo de la aplicación.
- Si se desea hacer una modificación al modelo del dominio, como aumentar métodos o datos contenidos, sólo debe modificarse el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.
- Las modificaciones a las vistas no afectan en absoluto a los módulos de la aplicación.
- MVC está demostrando ser un patrón de diseño bien elaborado pues las aplicaciones que lo implementan presentan una extensibilidad y una mantenibilidad únicas comparadas con otras aplicaciones basadas en otros patrones.
- Hay un API muy bien definido; cualquiera que use el API, podrá reemplazar el Modelo, Vista o el Controlador, sin aparente dificultad.
- La conexión entre el Modelo y sus Vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

Y también encontramos desventajas:

- El tiempo de desarrollo de una aplicación que implementa el patrón de diseño MVC es mayor, al menos en la primera etapa, que el tiempo de desarrollo de una aplicación que no lo implementa, ya que MVC requiere que el programador implemente una mayor cantidad de clases que en un entorno de desarrollo común no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación MVC es muchísimo más mantenible, extensible y modificable que una aplicación que no lo implementa.
- MVC requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos: un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.

- MVC es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.

2.2 Tecnologías en el Desarrollo de Aplicaciones Web

El desarrollo de una aplicación Web requiere el uso de un conjunto de tecnologías que permiten diseñar y estructurar el contenido de las páginas, implementar las funcionalidades y el dinamismo de las mismas, alojarlas y ponerlas en funcionamiento para su utilización por parte de los usuarios. El diseño y desarrollo de aplicaciones Web consiste en implementar sus necesidades, objetivos o ideas en Internet utilizando las tecnologías más idóneas según su proyecto. [12]

El desarrollo de aplicación Web, que no son sino aplicaciones basadas en el modelo Cliente/Servidor donde la comunicación con el usuario se hace utilizando páginas Web. El código de la aplicación se puede ejecutar en el cliente, en el servidor, o distribuirse entre ambos. Suelen utilizar una Base de Datos para organizar y facilitar el acceso a la información. Las ventajas que presentan son: su facilidad de manejo y de desarrollo, accesibilidad y portabilidad. [13]

Entre estas tecnologías tenemos:

- **Tecnologías del Lado del Cliente:** están insertadas en la página de HTML del cliente y son interpretadas y ejecutadas por el navegador. Estas tecnologías son utilizadas fundamentalmente para mostrar la información y dar estética al sitio Web. En particular estudiaremos **HTML, CSS, JavaScript, Ajax y Jquery**.
- **Tecnologías del Lado del Servidor:** permiten construir código que se ejecute en el servidor Web, justo antes de que se envíe la página a través de internet al cliente. Existen diversas tecnologías del lado del servidor, ampliamente utilizadas en el desarrollo de aplicaciones Web, de ésta diversidad se hace énfasis en PHP, por ser una tecnología poderosa, facilita la implementación de aplicaciones Web minimizar el tiempo de desarrollo. En particular estudiaremos **Apache, PHP, Mysql.**
- **Servidores de Aplicaciones:** proporcionan servicios que soportan la ejecución y disponibilidad de las aplicaciones desplegadas. Será descrito el Servidor ***Apache*** por ser uno de los servidores más utilizados para el despliegue de aplicaciones Web desarrolladas en PHP.

- **Sistemas Manejadores de Bases de Datos:** consisten en un conjunto de datos relacionados entre sí y un conjunto de herramientas de software (y/o hardware) para tener acceso a dichos datos, y a su vez procesarlos y administrarlos. Se estudiará MySql como sistema manejador de base de datos de software libre altamente integrable con aplicaciones en PHP.

2.2.1 Tecnologías del Lado del Cliente

Las tecnologías del lado del cliente son todas aquellas que son ejecutadas e interpretadas del lado del cliente en una aplicación Web (*browser* o navegador Web). En muchos casos el buen funcionamiento de estas tecnologías va a depender del tipo de browser y de la versión de cada uno de ellos.

Por lo general estas tecnologías son utilizadas fundamentalmente para mostrar información, para darle formato a esa información, para solicitar datos, etc. Además se pueden generar todo tipo de documento de manera estática como dinámica en un browser. Las tecnologías que se van a explicar a continuación son las siguientes: JavaScript, jQuery y Ajax.

2.2.1.1 JavaScript

JavaScript [16] [17] es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas Web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas Web dinámicas.

Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de Herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores interpretan el código JavaScript integrado dentro de las páginas Web. Para interactuar con una página Web se provee al lenguaje JavaScript de una implementación del DOM.

Es necesario resaltar que hay dos tipos de JavaScript por un lado está el que se ejecuta en el cliente, este es el JavaScript propiamente dicho, aunque técnicamente se

denomina Navigator JavaScript. Pero también existe un JavaScript que se ejecuta en el servidor, es más reciente y se denomina LiveWire JavaScript.

Características de JavaScript

- Es un lenguaje basado en objetos y orientado a eventos, diseñado específicamente para el desarrollo de aplicaciones dentro del ámbito de Internet.
- Es sencillo y pensado para proveer rapidez y ligereza en las actividades realizadas sobre plataforma Web.
- Los programas en JavaScript pueden estar incluidos en los documentos HMTL y/o tener la referencia a un archivo externo de extensión 'js'; y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos y manejar los elementos de la página.
- Es un lenguaje que permite tanto la programación de pequeños scripts, como de programas más grandes, orientados a objetos, con funciones y estructuras de datos complejas.
- Es soportado por la mayoría de los navegadores como Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros.

Ventajas de JavaScript

- Se ejecuta del lado del cliente, por lo tanto no sobrecarga el servidor si existen muchas peticiones.
- Es multiplataforma.
- Es un lenguaje orientado a eventos manteniendo características de programación orientado a objetos.
- Los programas JavaSript tienden a ser pequeños y compactos; no requieren mucha memoria ni tiempo adicional de transmisión. Además, al incluirse dentro de las mismas páginas HTML se reduce el número de accesos independientes a la red.

Desventajas de JavaScript

- El código debe descargarse completamente antes de ser ejecutado, esto puede en algunos casos aumentar el tiempo de respuesta de la página al momento de la carga inicial.

- Por razones de seguridad las opciones de ejecución y uso de recursos del lado del cliente JavaScript están muy limitadas, sobre todo el acceso al hardware y archivos.
- No es posible ocultar el código fuente y evitar la copia y reutilización de éste.
- Los navegadores tienen la opción de evitar la ejecución de códigos Javascript de las páginas Web, por lo tanto será decisión del cliente si lo permitirá o no.

Tabla 1. Diferencias entre Java y JavaScript. Fuente: Elaboración Propia (2010).

JAVA	JAVASCRIPT
Compilado (*bytecodes*). Se descarga del servidor y se ejecuta en el cliente	Interpretado por el cliente
Es necesario definir los tipos de datos de las variables	Los tipos de datos de las variables no se declaran
Se utilizan APPLETS. Se accede a ellos desde documentos.	El código se integra e incrusta en documentos HTML
Orientado a Objeto (Basado en clases)	Basado en objetos
No se puede escribir automáticamente en el disco duro.	No se puede escribir automáticamente en el disco duro.

2.2.1.2 AJAX

AJAX [18] [19] son las siglas del acrónimo de ***A****synchronous* **J**avaScript **A**nd **X**ML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que permite hacer las páginas de internet más iterativas. Es una técnica de desarrollo Web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de algunas tecnologías ya existentes:

- **XHTML** o **HTML** y **CSS** para el diseño que acompaña a la información.

- **DOM** accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto **XMLHttpRequest** para intercambiar datos asincrónicamente con el servidor Web.
- **XML** es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML pre formateado, texto plano, entre otros.

El modelo clásico de aplicaciones Web funciona de la forma en que 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, habla con varios sistemas propietarios), y le devuelve una página HTLM al cliente. Este acercamiento tiene mucho sentido a nivel técnico, pero no lo tiene para una gran experiencia de usuario. Mientras el servidor está haciendo lo suyo, ¿qué está haciendo el usuario? Esperando. Y, en cada paso de la tarea, el usuario espera por más. Obviamente, si estuviéramos diseñando la Web desde cero para aplicaciones, no querríamos hacer esperar a los usuarios. Una vez que la interfaz está cargada, ¿por qué la interacción del usuario debería detenerse cada vez que la aplicación necesita algo del servidor? De hecho, ¿por qué debería el usuario ver la aplicación yendo al servidor?

Una aplicación AJAX, elimina la naturaleza “arrancar-frenar-arrancar-frenar” de la interacción en la Web, esto lo hace introduciendo un intermediario “*motor AJAX*” entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario. Cada vez que el usuario ejecuta una acción (un click, la presión de una tecla, el arrastre de un objeto del navegador, etc.) debe solicitar datos al servidor a través de Internet, para luego regenerar la página que el usuario está viendo. Estamos acostumbrados a un modelo de interacción sincrónica basada en click-petición-presentación. Con AJAX la interacción pasa a ser asíncrona. Cada vez que se hace click no necesariamente se establece una conexión con el servidor. Básicamente, la principal virtud de AJAX está en la potencia que se le puede extraer al trabajo asíncrono de peticiones al servidor.

El modelo de AJAX propone descargar un motor especial para cada aplicación, sirviéndose de las tecnologías antes mencionadas y presentes en la gran mayoría de los navegadores. De esta forma, los usuarios pueden acceder de inmediato al contenido sin

interrupciones gracias a que el motor de AJAX intercambia datos silenciosamente con el servidor para que estén listos cuando el usuario los requiera y sin recargar la página. Ahí está la magia precisamente, en que ciertos procesos se muestran en la página sin retardo alguno, ya que mientras el usuario miraba unos datos en la pantalla, AJAX le ha estado preparando los siguientes que iba a necesitar.

En el modelo de aplicación Web de AJAX en vez de cargar una página Web, al inicio de la sesión, el navegador carga el motor AJAX. Este motor es el responsable de renderizar la interfaz que el usuario ve y de comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo. La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor.

Ventajas de AJAX:

- Recuperación asíncrona de datos, el usuario no tienen que esperar después de una petición.
- La página se asemeja a una aplicación de escritorio.
- Es portable (no requiere plug – in como flash y Apple como Java) .
- Reduce tiempos de escritura y tiempos de espera.
- Reduce el ancho de banda.
- Reduce la carga al servidor (validación de formularios).
- Utiliza tecnologías ya existentes.
- Soportada por la mayoría de los navegadores modernos.

Desventajas de AJAX:

- Pueden aumentar las llamadas al servidor.
- Peligro de incompatibilidades en navegadores.
- Deja de existir el botón atrás (los usuarios deben cambiar su manera de entender los sitios Web, debido a que se pierde el concepto de volver a la página anterior).
- Uso excesivo de Javascript: seguridad, compatibilidad, accesibilidad y complejidad.
- La percepción de cambio es menor.

- Dependiendo de la carga del servidor podemos experimentar tiempos tardíos de repuesta que desconciertan al visitante.
- Requiere programadores que conozcan todas las tecnologías que intervienen en AJAX.
- No funciona si el usuario tiene desactivado el JavaScript del navegador.

2.2.1.3 jQuery

El jQuery [20] [21] es una librería liviana de JavaScript para interactuar con los elementos de una Web por medio del DOM de un modo simplificado. Lo que lo hace tan especial es su sencillez y su reducido tamaño.

Las aplicaciones en internet son más complejas, ya que incorporan efectos visuales, drag and drop, auto-completar, animaciones, etc. El desarrollar todos estos conceptos desde cero puede resultar complicado sobre todo si tenemos que presentar la solución con muy poco tiempo, en este tipo de situaciones el empleo de librerías como el jQuery nos facilita el desarrollo de la aplicación. Otra ventaja paralela es despreocuparnos cuando codificamos en la compatibilidad de navegadores, ya que la librería resolverá esto.

Esta librería de JavaScript permite interactuar con los documentos HTML permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX. Es de código abierto bajo la licencia de MIT y GPL v2) e increíblemente ligera. Entre sus usuarios podemos encontrar a Google, Microsoft, IBM, Amazon, Twitter, entre otros…., este framework JavaScript, nos ofrece una infraestructura con la que tendremos mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery obtendremos ayuda en la creación de interfaces de usuario, efectos dinámicos, aplicaciones que hacen uso de Ajax, etc. Cuando programemos JavaScript con jQuery tendremos a nuestra disposición una interfaz para programación que permitirá hacer cosas con el navegador que estemos seguros que funcionarán para todos los visitantes.

Para poder utilizar esta librería lo primero que se tiene que hacer es descargar el script desde su página Web, subirlo a nuestro servidor, y ejecutarlo con la etiqueta script. jQuery tiene una sintaxis muy sencilla. Los comandos se reconocen por comenzar con el símbolo “$”. Por ejemplo en la siguiente línea se presenta la llamada de la librería js.

```
<Script type=”text/javascript” src=”jquery.js”></Script>
```

Ventajas de jQuery

- Ahorra muchas líneas de código.
- Hace transparente el soporte de nuestra aplicación para los navegadores principales.
- Provee de un mecanismo para la captura de eventos.
- Proporciona un conjunto de funciones para animar el contenido de la página en forma muy sencilla.
- Integra funcionalidades para trabajar con Ajax.

Desventajas de jQuery

- Pueden ser lentos (para el motor js del cliente).
- Algunos novatos solo aprenden desde el framework y dependen de él, y no llegan a dominar js puro.

2.2.2 Tecnologías del Lado del Servidor

Las tecnologías del lado del servidor consisten en el procesamiento de una petición de un usuario mediante la interpretación de un script en el servidor Web para generar páginas HTML dinámicamente como respuesta. [22]

Los primeros servidores Web permitían visualizar exclusivamente información estática. Esto representó bien pronto una limitación; sobre todo desde el momento en el que la actividad publicitaria y comercial comenzó a concentrarse también en la red Internet. La primera solución técnica realizada fue la posibilidad que el servidor Web lanzase programas residentes en la máquina de servicio. Esta tecnología, conocida como CGI permite al servidor Web lanzar programas escritos principalmente en C. Si bien la tecnología CGI resolvía el problema de la presentación exclusivamente estática de la información, al mismo tiempo presentaba dos limitaciones importantes: una era el problema de seguridad que podía representar el hecho que mediante la llamada a una página se pueda ejecutar programas indeseados en el servidor, la segunda era de carga del servidor (si una misma página que lanzaba un programa era llamada desde 100 clientes concurrentemente, en el servidor se ejecutaban 100 procesos, uno por cada cliente que solicitaba esa página).

Para resolver estos problemas, se buscó desarrollar una tecnología que permitiera ejecutar, en un único proceso del servidor, todos los pedidos de ejecución de código del

servidor Web sin importar la cantidad de clientes que se conectaban concurrentemente. Esta solución, denominada servlet en tecnología Java o filtro en tecnología Microsoft, permite el poder ejecutar código en un único proceso externo que gestiona todas las llamadas realizadas por el servidor Web, impidiendo al mismo tiempo que el servidor Web pueda llamar a ejecutar programas del sistema operativo.

Desde ese momento, se comienzan a desarrollar tecnologías del lado del servidor que controlan los problemas y posibles limitaciones que presentan las aplicaciones Web y que hacen más eficientes el procesamiento, la comunicación con el cliente y la presentación de las páginas finales.

Fueron entonces desarrollados lenguajes que pueden ser incluidos al interno de archivos HTML. Estos comandos pueden ser interpretados (como por ejemplo las páginas ASP o PHP) o precompilados (como en las páginas JSP o ASP.NET).

2.2.2.1 PHP

PHP acrónimo de ***H****ypertext* ***P****re-Processor*. [23][24] Es un lenguaje libre (Open Source) interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. Usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios Web. Últimamente también para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfico. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características especificas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas. No es un lenguaje de marcas como podría ser HTML o XML. Está más cercano a JavaScript o a C.

A diferencia de JavaScript que se ejecutan en el navegador, PHP, se ejecuta en el servidor por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado es enviado al navegador. El resultado es normalmente una página HTML pero también podría ser una página *WML* (Wap).

Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, sin embargo, para que sus páginas PHP funcionen el servidor donde están alojadas debe soportar PHP.

Características de PHP

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas:

- Soporte para una gran cantidad de bases de datos: MySql, PostgreSQL, Oracle, MS SQL Server, Sybase, mSQL, Informix, ODBC, entre otras.
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Soportado por una gran comunidad de desarrolladores, como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y reparen rápidamente.
- El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar las capacidades de PHP.
- Con PHP se puede hacer cualquier cosa que podemos realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas.
- PHP tiene soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, POP3, HTTP, COM (en Windows) y muchos otros.

Ventajas de PHP

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad.
- Leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos.
- Posee una amplia documentación en su página oficial, entre las que se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.

- Permite las técnicas de Programación Orientada a Objetos.
- Permite crear los formularios para la Web.
- Biblioteca nativa de funciones sumamente amplia e incluída
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.

Desventajas de PHP

- Todo el trabajo lo realiza el servidor y no delega al cliente. Por tanto puede ser más ineficiente a medida que las solicitudes aumenten de número.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La orientación a objetos es aún muy deficiente para aplicaciones grandes.

2.2.2.2 Apache

Apache [25] [26] es un servidor de páginas Web de tecnología *Open Source* sólido y para uso comercial desarrollado por la *Apache Software Foundation*. Actualmente en su versión 2.2.17. Escrito en C, sistema operativo multiplataforma.

Apache está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y los diferentes entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades, o que una misma característica o funcionalidad sea implementada de diferente manera para obtener una mayor eficiencia. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios Web elegir que funcionalidades van a ser incluidas en el servidor seleccionando que módulos se van a usar, ya sea al compilar o al ejecutar el servidor. La configuración de cada módulo se hace mediante la configuración de las directivas que están contenidas dentro del módulo. Los módulos de Apache se pueden clasificar en tres categorías:

- **Módulos Base:** módulo con las funciones básicas de Apache.
- **Módulos Multiproceso:** son los responsables de la unión con los puertos del computador, acepando las peticiones y enviando a los hijos a atender a las peticiones.
- **Módulos Adicionales:** cualquier otro módulo que le añada una funcionalidad al servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta Apache, optimizando el rendimiento y rapidez del código.

Características de Apache

- **Sencillo, con la configuración basada en un poderoso archivo:** el servidor Apache no posee una interfaz de usuario grafica para su administración. Se trata de un sencillo archivo de configuración llamado httpd.conf que se puede utilizar para configurar Apache.
- **Independencia de plataforma:** Apache funciona en casi todas las plataformas actuales. Debido a esto es posible escoger la plataforma que se desee, y también cambiar la plataforma si en un momento determinado una plataforma ofrece más ventajas que la que se esté utilizando.
- **Soporte para CGI (Common Gateway Interface):** Apache soporta CGI utilizando los módulos mod cgi y mod cgid. Es compatible con CGI y aporta características extendidas como personalización de las variables de entorno y soporte de reparación de errores o debugging, que son difíciles de encontrar en otros servidores Web.
- **Soporte de scripts PHP:** este lenguaje de script ha comenzado a ser muy utilizado y Apache tiene un amplio soporte de PHP utilizando el módulo mod php.
- **Soporte de Secured Socket Layer (SSL):** puede crear fácilmente un sitio Web SSL utilizando OpenSSL y el módulo mod ssl de Apache.
- **Autenticación de diferentes tipos:** Apache permite la autenticación de usuarios en varias formas. Apache permite el uso de bases de datos DBM para la autenticación de usuarios. De esta forma se puede restringir el acceso a determinadas páginas de un sitio Web de una forma sencilla y de fácil mantenimiento.
- **Respuestas personalizadas ante errores del servidor:** Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

Ventajas de Apache

- Amplias librerías disponibles, especialmente en Perl y PHP.
- Disponible para numeroso Sistemas Operativos.
- Servidor Potente y Estable.
- Código Abierto, por lo que se pueden programas Módulos personalizados e integrarlos al servidor.
- Altamente configurable.
- Estabilidad.
- Independencia de la plataforma.
- Complejidad.
- Falta de integridad.

2.2.2.3 Sistemas Manejadores de Bases de Datos

El sistema manejador de bases de datos (SMBD) [27] [28] es la porción más importante del software de un sistema de base de datos. Un SMBD es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

El SMBD es esencial para el adecuado funcionamiento y manipulación de los datos contenidos en la base de datos. Se puede definir como: *"El Conjunto de programas, procedimientos, lenguajes, etc. que suministra, tanto a los usuarios no informáticos como a los analistas, programadores o al administrador, los medios necesarios para describir, recuperar y manipular los datos almacenados en la base, manteniendo su integridad, confidencialidad y seguridad"*. [27] [28]

2.2.2.3.1 MySql

Es un sistema manejador de base de datos relacional, multihilo y multiusuario, cuya característica principal es la de ser *Open Source* o de código abierto, permitiendo su uso y adaptación a las necesidades de los usuarios.

Características de MySql

Entre las principales características del manejador se pueden mencionar las siguientes:

- **Rendimiento:** MySql permite obtener un buen rendimiento del hardware, ya que aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo, lo que optimiza los tiempos de respuesta.
- **ACID:** son las propiedades que una base de datos debe cumplir para que el sistema manejador de base de datos (SMBD) maneje correctamente la transacción, el término ACID viene de atomicidad, consistencia, aislamiento, durabilidad. MySql cumple con este principio.
- **Multi-Usuario:** MySql es capaz de soportar que varios usuarios se conecten al mismo tiempo y que puedan manipular y administrar las distintas bases de datos existentes y administrar al SMBDR como tal, en el mismo instante de tiempo.
- **Multiplataforma**.
- **Almacenamiento:** proporciona sistemas de almacenamiento transaccionales y no transaccionales (no permiten hacer *rollback*).
- **Seguridad:** MySql implementa un protocolo que se encarga de emplear diferentes algoritmos para cifrar los datos que viajan a través de una red pública (ejemplo: Internet), tiene por nombre capa de seguridad de sockets (*Secure Sockets Layer – SSL*). Este protocolo trabajaría en el cliente y el servidor MySql.
- **Velocidad:** cuando se manipulan datos con el tipo de tabla MyISAM. en la utilización de *joins* y procesos de optimización para los *left join y right join*.

Las siguientes características son implementadas únicamente por MySql:

- Múltiples motores de almacenamiento (MyISAM, Merge, InnoDB, BDB, Memory/heap, MySql Cluster, Federated, Archive, CSV, Blackhole y Example en 5.x), permitiendo al usuario escoger la que sea más adecuada para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

Ventajas de MySql

- MySql es un software Open Source.
- Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mayor rendimiento.

- Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- Facilidad de configuración e instalación.
- Soporta gran variedad de sistemas operativos.
- Baja probabilidad de corromper los datos.
- Su conectividad, velocidad y seguridad hacen de MySql altamente apropiado para acceder bases de datos en internet.
- El software MySql usa la licencia GPL.

Desventajas de MySql

- Un gran porcentaje de las utilidades de MySql no están documentadas.
- No es intuitivo, como otros programas por ejemplo access.

2.3 Framework

Los frameworks se utilizan en el ámbito de la programación de aplicaciones desde hace décadas. Recientemente han comenzado a utilizarse para programar y diseñar aplicaciones Web, por lo que ya existen decenas de frameworks para CSS. Actualmente existen decenas de frameworks.

Genéricamente, un framework es un conjunto de herramientas, librerías, convenciones y buenas prácticas que pretenden encapsular las tareas repetitivas en módulos genéricos fácilmente reutilizables. De la misma forma, un framework es un conjunto de herramientas, hojas de estilos y buenas prácticas que permiten al diseñador Web olvidarse de las tareas repetitivas para centrarse n los elementos únicos de cada diseño en los que pueda aportar valor.

Los framework más completos incluyen utilidades para que el diseñador no tenga que trabajar en ningún aspecto genérico del diseño Web. Por este motivo, es habitual que los mejores frameworks incluyan herramientas para:

- Neutralizar los estilos por defecto que aplican los navegadores.
- Manejar correctamente el texto, de forma que todos los contenidos se vean exactamente igual en todos los navegadores y que sean adaptables para mejorar su accesibilidad y permitir su acceso en cualquier medio y/o dispositivo.

- Crear cualquier estructura compleja o layout de forma sencilla, con la seguridad de que funciona correctamente en cualquier versión de cualquier navegador. [31]

2.3.1 Framework CakePHP

Es un framework *open-source* orientado al desarrollo rápido y mantenimiento de aplicaciones Web sobre el lenguaje PHP. Este framework utiliza patrones de diseños habituales como MVC (*Mode-View-Controller*) y ORM (*Object-relational mapping*), reduce los costos de desarrollo y ayuda a los programas a escribir menos códigos. [32]

Características del Framework

- Comunidad activa y amistosa.
- Licencia Flexible.
- Compatible con PHP4 y PHP5.
- CRUD integrado para la interacción con la Base de Datos.
- Soporte de Aplicaciones [*Scaffolding*]
- Generación de Códigos.
- Arquitectura Modelo Vista Controlador (MVC).
- Despachador de peticiones [Dispatcher], con URLs y rutas personalizadas y limpias.
- Validación integrada.
- Plantillas rápidas y flexibles (Sintaxis de PHP, con ayudantes [*helpers*]).
- Ayudantes para AJAX, JavaScript, formularios HTML y más.
- Componentes de Email, Cookies, Seguridad, Sesión, y Manejo de Solicitudes.
- Listas de control de acceso flexibles.
- Limpieza de datos.
- Caché flexible.
- Localización.
- Funciona en cualquier subdirectorio del sitio Web, con poca o ninguna configuración de Apache.

Los archivos y carpetas que se generan a la hora de la instalación:

- App.
- Cake.

- Vendors
- .htaccess.
- Index.php.
- Readme

La carpeta App es donde se hará la mayor parte del desarrollo de su aplicación. Veamos un poco más de cerca las carpetas dentro de App.

- Config: mantiene los (pocos) archivos de configuración que CakePHP usa. Detalles de conexión a bases de datos, arranque [bootstraping], archivos de configuración del núcleo y más deberían ser almacenados aquí.
- Controllers: contiene los controladores de tu aplicación y sus componentes.
- Locale: almacena archivos de cadenas para la internacionalización.
- Models: contiene los modelos de su aplicación, comportamientos [*behaviors*], y orígenes de datos [*datasources*].
- Plugins: contiene los paquetes de plugins.
- Tmp: aquí es donde CakePHP almacena información que actualmente se almacena depende en cómo haya configurado a CakePHP, pero esta carpeta es usualmente usada para almacenar descripciones de modelos, registros (logs), y algunas veces información de sesiones.
- Vendors: cualquier grupo de clases o librerías de terceros debería ser ubicado aquí. Hacerlo así hace que sean más fáciles de acceder a ellas usando la función App::Import[‘Vendor’]. Los observadores meticulosos notarán que esto parece redundante, ya que también existe una carpeta vendors en el nivel superior de nuestra estructura de directorios. Entraremos en las diferencias entre las dos cuando discutamos acerca de la administración de múltiples aplicaciones e instalaciones más complejas.
- Views: los archivos de presentación son ubicados aquí: elementos [*elements*], páginas de error, ayudantes [*helpers*], *layouts* y archivos de vistas.
- Webroots: en una configuración de producción, esta carpeta debería servir como la raíz del sitio [*document root*] para su aplicación. Las carpetas aquí también sirven como lugares de almacenamiento para hojas de estilos en cascada [*CSS stylesheets*], imágenes y archivos JavaScript.

2.4 Bases Metodológicas del Desarrollo de la Aplicación

En este punto se describe el método de desarrollo que será utilizado para realizar el trabajo especial de grado. El procedimiento metodológico a seguir para lograr los objetivos específicos, se constituirá en la aplicación de una instanciación de la metodología XP. Se tomo esta metodología ágil XP, porque es muy adaptable a las necesidades que se presenta dentro de la empresa ya que no sigue un régimen muy estricto para poderla seguir. Además al tener un enfoque en el trabajo en grupo no es más fácil el dividir las contribuciones en el proyecto. [33] [34] [35].

Además a comparación a otras metodologías como la RUP es mucho más rápido, ya que la metodología XP conlleva menos protocolos. Lo que evita que existan jerarquías dentro del grupo.

“Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.” “Kent Beck”.

2.4.1 Programación Extrema (XP)

Nace como una disciplina de desarrollo de software hace aproximadamente unos seis años. La Programación Extrema es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado. [33]

Las cuatro (4) variables:

- Costo: máquinas, especialistas y oficinas.
- Tiempo: total y de Entregas.
- Calidad: externa e Interna.
- Alcance: Intervención del cliente.

Objetivos de XP

Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita. Por tanto, debemos

responder muy rápido a las necesidades del cliente, incluso cuando los cambios sean al final de ciclo de la programación.

El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

Para comenzar a desarrollar un buen software debemos seguir los cuatros (4) valores, debido a que el ciclo de vida del desarrollo de un proyecto software los cambios van a aparecer, cambiarán los requisitos, las reglas de negocio, el personal, la tecnología, todo va a cambiar. Por tanto el problema no es el cambio en sí, ya que este va a suceder sino la incapacidad de enfrentarnos a estos cambios.

Como en otra cualquier actividad humana necesitamos valores para desarrollar nuestro trabajo y conseguir los planteamientos iniciales, la Programación Extrema se basa en los cuatros (4) valores que debe conformar un desarrollo del sistema como los son:

- **Comunicación:** XP recomienda que el código debe ser autodocumentado debido a que es más fiable, y mientras más sencillo sea el código es mejor para el entendimiento de cualquier programador.
- **Sencillez:** para XP la base de la programación es la simplicidad del diseño para agilizar el desarrollo con esto se garantiza que otro programador pueda realizar modificaciones en el código, siempre y cuando manteniendo la refactorización del código para así asegurar que el código se mantenga simple.
- **Retroalimentación (feedback):** con la Programación Extrema se garantiza que los requerimientos del proyecto se cumplan debido a que uno de los integrantes de desarrollo es el mismo cliente, que permite conocer el estado del mismo en tiempo real.
- **Valentía:** la Programación Extrema consiste en que el desarrollo es en pareja por lo tanto se debe de confiar en el equipo de trabajo ya que esto garantiza que la programación en pareja benéfica la calidad del código sin repercutir negativamente en la productividad.

Los programadores de XP se comunican con sus clientes y compañeros programadores. Mantienen su diseño simple y limpio. XP es una alternativa para cambiar los métodos tradicionales de desarrollo de software.

La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de XP consideran que los cambios de requisitos sobre la marcha son aspectos naturales, inevitables e incluso deseables del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

El proceso de desarrollo de XP se divide en cuatro (4) actividades (Planificación, Diseño, Codificación y Pruebas).

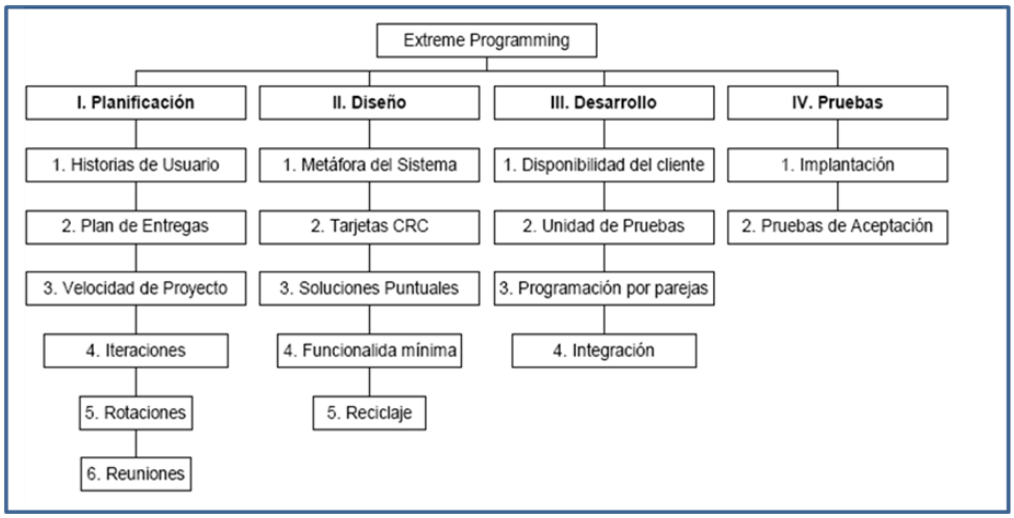


Figura 6. Fases de la Metodología XP. Fuente: Fernández Gerardo (2002).

1ª Fase: Planificación del Proyecto:

Historias de usuario: el primer paso de cualquier proyecto que siga la metodología XP es definir las historias de usuario con el cliente. Las historias de usuario tienen la misma finalidad que los casos de uso pero con algunas diferencias: Constan de 3 ó 4 líneas escritas por el cliente en un lenguaje no técnico sin hacer mucho hincapié en los detalles; no se debe hablar ni de posibles algoritmos para su implementación ni de diseños de base de datos adecuados, etc. Son usadas para estimar tiempos de desarrollo de la parte de la aplicación que describen. También se utilizan en la fase de pruebas, para verificar si el programa cumple con lo que especifica la historia de usuario. Cuando llega la hora de implementar una historia de usuario, el cliente y los desarrolladores se reúnen para concretar y detallar lo que

tiene que hacer dicha historia. El tiempo de desarrollo ideal para una historia de usuario es entre 1 y 3 semanas.

Release planning: después de tener ya definidas las historias de usuario es necesario crear un plan de publicaciones, en inglés "*Release Plan*", donde se indiquen las historias de usuario que se crearán para cada versión del programa y las fechas en las que se publicarán estas versiones. Un "*Release Plan*" es una planificación donde los desarrolladores y clientes establecen los tiempos de implementación ideales de las historias de usuario, la prioridad con la que serán implementadas y las historias que serán implementadas en cada versión del programa. Después de un "*Release Plan*" tienen que estar claros estos cuatro factores: los objetivos que se deben cumplir (que son principalmente las historias que se deben desarrollar en cada versión), el tiempo que tardarán en desarrollarse y publicarse las versiones del programa, el número de personas que trabajarán en el desarrollo y cómo se evaluará la calidad del trabajo realizado. (*Release Plan* Planificación de publicaciones).

Iteraciones: todo proyecto que siga la metodología XP se ha de dividir en iteraciones de aproximadamente 3 semanas de duración. Al comienzo de cada iteración los clientes deben seleccionar las historias de usuario definidas en el "*Release planning*" que serán implementadas. También se seleccionan las historias de usuario que no pasaron el test de aceptación que se realizó al terminar la iteración anterior. Estas historias de usuario son divididas en tareas de entre 1 y 3 días de duración que se asignarán a los programadores.

Velocidad del proyecto: la velocidad del proyecto es una medida que representa la rapidez con la que se desarrolla el proyecto; estimarla es muy sencillo, basta con contar el número de historias de usuario que se pueden implementar en una iteración; de esta forma, se sabrá el cupo de historias que se pueden desarrollar en las distintas iteraciones. Usando la velocidad del proyecto controlaremos que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración. Es conveniente reevaluar esta medida cada 3 ó 4 iteraciones y si se aprecia que no es adecuada hay que negociar con el cliente un nuevo "*Release Plan*".

Programación en pareja: la metodología XP aconseja la programación en parejas pues incrementa la productividad y la calidad del software desarrollado. El trabajo en pareja involucra a dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método o función es adecuado y está bien diseñado. De esta forma se consigue un código y diseño con gran calidad.

Reuniones diarias: es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta. Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

2ª Fase: Diseño

Diseños simples: la metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e implementar que a la larga costará menos tiempo y esfuerzo desarrollar.

Glosarios de términos: usar glosarios de términos y un correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.

Riesgos: si surgen problemas potenciales durante el diseño, XP sugiere utilizar una pareja de desarrolladores para que investiguen y reduzcan al máximo el riesgo que supone ese problema.

Funcionalidad extra: nunca se debe añadir funcionalidad extra al programa aunque se piense que en un futuro será utilizada. Sólo el 10% de la misma es utilizada, lo que implica que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.

Refactorizar: refactorizar es mejorar y modificar la estructura y codificación de códigos ya creados sin alterar su funcionalidad. Refactorizar supone revisar de nuevo estos códigos para procurar optimizar su funcionamiento. Es muy común rehusar códigos ya creados que contienen funcionalidades que no serán usadas y diseños obsoletos. Esto es un error porque puede generar código completamente inestable y muy mal diseñado; por este motivo, es necesario refactorizar cuando se va a utilizar código ya creado.

Tarjetas C.R.C.: el uso de las tarjetas C.R.C (*Class, Responsabilities and Collaboration*) permiten al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedimental clásica. Las tarjetas C.R.C representan objetos; la clase a la que pertenece el objeto se puede escribir en la parte de arriba de la tarjeta, en una columna a la izquierda se pueden escribir las responsabilidades u objetivos que debe cumplir el objeto y a la derecha, las clases que colaboran con cada responsabilidad.

3ª Fase: Codificación

Como ya se dijo en la introducción, el cliente es una parte más del equipo de desarrollo; su presencia es indispensable en las distintas fases de XP A la hora de codificar una historia de usuario su presencia es aún más necesaria. No olvidemos que los clientes son los que crean las historias de usuario y negocian los tiempos en los que serán implementadas. Antes del desarrollo de cada historia de usuario el cliente debe especificar detalladamente lo que ésta hará y también tendrá que estar presente cuando se realicen los test que verifiquen que la historia implementada cumple la funcionalidad especificada.

La codificación debe hacerse ateniendo a estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente y facilita su comprensión y escalabilidad. Crear test que prueben el funcionamiento de los distintos códigos implementados nos ayudará a desarrollar dicho código. Crear estos test antes nos ayuda a saber qué es exactamente lo que tiene que hacer el código a implementar y sabremos que una vez implementado pasará dichos test sin problemas ya que dicho código ha sido diseñado para ese fin. Se puede dividir la funcionalidad que debe cumplir una tarea a programar en pequeñas unidades, de esta forma se crearán primero los test para cada unidad y a continuación se desarrollará dicha unidad, así poco a poco conseguiremos un desarrollo que cumpla todos los requisitos especificados.

Como ya se comentó anteriormente, XP opta por la programación en pareja ya que permite un código más eficiente y con una gran calidad. XP sugiere un modelo de trabajo usando repositorios de código dónde las parejas de programadores publican cada pocas horas sus códigos implementados y corregidos junto a los test que deben pasar. De esta forma el resto de programadores que necesiten códigos ajenos trabajarán siempre con las últimas versiones. Para mantener un código consistente, publicar un código en un repositorio es una acción exclusiva para cada pareja de programadores.

XP también propone un modelo de desarrollo colectivo en el que todos los programadores están implicados en todas las tareas; cualquiera puede modificar o ampliar una clase o método de otro programador si es necesario y subirla al repositorio de código. El permitir al resto de los programadores modificar códigos que no son suyos no supone ningún riesgo ya que para que un código pueda ser publicado en el repositorio tiene que pasar los test de funcionamiento definidos para el mismo.

La optimización del código siempre se debe dejar para el final. Hay que hacer que funcione y que sea correcto, más tarde se puede optimizar. XP afirma que la mayoría de los proyectos que necesiten más tiempo extra que el planificado para ser finalizados no podrán ser terminados a tiempo se haga lo que se haga, aunque se añadan más desarrolladores y se incrementen los recursos. La solución que plantea XP es realizar un nuevo "*Release plan*" para concretar los nuevos tiempos de publicación y de velocidad del proyecto.

A la hora de codificar no seguimos la regla de XP que aconseja crear test de funcionamiento con entornos de desarrollo antes de programar. Nuestros test los obtendremos de la especificación de requisitos ya que en ella se especifican las pruebas que deben pasar las distintas funcionalidades del programa, procurando codificar pensando en las pruebas que debe pasar cada funcionalidad.

4ª Fase: Pruebas

Uno de los pilares de la metodología XP es el uso de test para comprobar el funcionamiento de los códigos que vayamos implementando. El uso de los test en XP es el siguiente:

- Se deben crear las aplicaciones que realizarán las pruebas con un entorno de desarrollo específico para todas las pruebas.
- Hay que someter a pruebas las distintas clases del sistema omitiendo los métodos más triviales.
- Se deben crear los test que pasarán los códigos antes de implementarlos; en el apartado anterior se explicó la importancia de crear antes los test que el código.
- Un punto importante es crear test que no tengan ninguna dependencia del código que en un futuro evaluará. Hay que crear los test abstrayéndose del futuro código, de esta forma aseguraremos la independencia del test respecto al código que evalúa.
- Como se comentó anteriormente los distintos test se deben subir al repositorio de código acompañados del código que verifican. Ningún código puede ser publicado en el repositorio sin que haya pasado su test de funcionamiento, de esta forma, aseguramos el uso colectivo del código (explicado en el apartado anterior).
- El uso de los test es adecuado para observar la refactorización. Los test permiten verificar que un cambio en la estructura de un código no tiene porqué cambiar su funcionamiento.

- Test de aceptación. Los test mencionados anteriormente sirven para evaluar las distintas tareas en las que ha sido dividida una historia de usuario. Para asegurar el funcionamiento final de una determinada historia de usuario se deben crear "Test de aceptación"; estos test son creados y usados por los clientes para comprobar que las distintas historias de usuario cumplen su cometido.
- Al ser las distintas funcionalidades de nuestra aplicación no demasiado extensas, no se harán test que analicen partes de las mismas, sino que las pruebas se realizarán para las funcionalidades generales que debe cumplir el programa especificado en la descripción de requisitos.

2.5 Terminologías asociadas al área de estudio del problema: Seguros y Declaración de Siniestros

A continuación se presentará un breve resumen de las principales palabras claves que se manejan en la Declaración de Siniestros, citados textualmente del glosario de términos de "Seguros Carabobo". [36]

Seguro

"Es la institución económica que elimina o reduce los perjuicios que en el patrimonio de una persona producen determinados acontecimientos fortuitos, distribuyendo aquellos perjuicios sobre una serie de personas en las cuales gravita el mismo riesgo, aunque no se haya cumplido". [36]

Accidente

"Es el acontecimiento inesperado, repentino e involuntario que puede ocasionar daños a personas (reportando consecuencias económicas o personales) o cosas independientemente de su voluntad". [36]

Asegurado

"Es el titular del interés asegurado, o la persona que está expuesta al riesgo de pérdida sobre ella misma, sus bienes o intereses económicos. Persona que mediante el pago de una prima traspasa el riesgo que recae sobre ella a una compañía aseguradora. Cuando contrata el seguro pasa a tener la doble calidad de Asegurado o Tomador". [36]

Asegurador

"*Es la empresa que asume la cobertura del riesgo, previamente autorizada a operar como tal por la Superintendencia de Seguros de la Nación, la empresa (sociedad anónima especial) constituida en conformidad a la ley, que asume el riesgo que terceras personas le traspasan, a cambio de una prima".* [36]

Aviso de Siniestro

"Comunicación efectuada por el Asegurado, donde le informa al Asegurador, la ocurrencia de determinado accidente, cuyas características guardan relación, en principio, con las circunstancias previstas en la póliza para que se efectúe la indemnización". [36]

Denuncia del Siniestro

"El tomador, o derecho habiente, en su caso, debe comunicar al asegurador el acaecimiento del siniestro dentro de los cinco días hábiles de conocerlo". [36]

Siniestro

"Considerado desde el punto de vista del Seguro, la ocurrencia del riesgo para la realización del evento previsto y garantizado por la póliza. En tal sentido, constituye siniestro la muerte de una persona sobre cuya cabeza se había estipulado un seguro de vida o naufragio de un buque, etc. Según la Guía Práctica de Seguros Lec (2001), indica que el siniestro es un acontecimiento futuro, incierto y que no depende de la voluntad del beneficiario, el hecho de la ocurrencia del mismo conlleva a que el asegurado de la empresa proceda a declararlo en las oficinas de la misma para posteriormente proceder a la reclamación del pago por daño ocurrido". [36]

Siniestro de Vehículo

"*La presente investigación se enfoca hacia los siniestros específicos en el caso de automóvil, los cuales tienen por objetivo amparar las pérdidas que pudieron ocasionarse parcial o totalmente en el vehículo".*

Las compañías aseguradoras otorgan las siguientes coberturas:

Pérdida Parcial *"conjuntamente con Pérdida Total (Cobertura Amplia)".*

***Pérdida Total**: "esta comprende las pérdidas parciales y la pérdida total del vehículo, considerándose total el robo o hurto del mismo, o cuando el importe de la reparación sea igual o mayor que setenta y cinco por ciento (75 %) del valor asegurado del vehículo, incluyendo sus accesorios.*

"La pérdida total a consecuencia de motín o disturbios callejeros podrá ser cubierta mediante anexo, pero no se indemnizarán las pérdidas parciales ocasionadas por los riesgos antes expuestos". [36]

Liquidación de Siniestro

"Es importante destacar que la cancelación por el daño ocurrido, o también llamada liquidación del siniestro, según la Guía Práctica de Seguros de Lec (2001). "Es el resarcimiento económico del daño o perjuicio causado. Por indemnización o liquidación de siniestros se entiende la contraprestación que debe pagar el asegurador como consecuencia del daño sufrido por el asegurado" (p. 6), es decir, el pago que debe efectuar una compañía aseguradora al presentarse un siniestro y el cual vendrá establecido por una serie de condiciones establecidas previamente en la suscripción de la póliza y que determinará el monto a cancelar, reserva de siniestro, y según la Guía Práctica de Seguros de Lec (2001)":

"Es el monto de una determinada cantidad de dinero que el asegurador estima debe constituir con parte de las primas satisfechas por los asegurados, para hacer frente a sus futuros compromisos frente a estos. Son las estimaciones que el asegurador hace respecto de sus futuras responsabilidades". [36]

"La finalidad de la reservas es dar la máxima garantía a los asegurados de que sus reclamos les serán indemnizados, o que a sus beneficiarios adquiridos les serán pagados o la parte de la prima no consumida le será devuelta, aún en el caso de que el asegurador suspenda pagos, quiebre o no disponga de fondos suficientes. El asegurado es acreedor privilegiado con respecto a las reservas técnicas que mantiene la compañía aseguradora". [36]

"Esta situación de saneamiento económico, a través de la vigilancia estatal, consolida la Institución del Seguro Privado (p.20)". [36]

"La liquidación del siniestro comienza con la comunicación que el asegurado o el beneficiario del seguro debe formular a la entidad aseguradora, para que este abone capital asegurado. Para la liquidación de los siniestros resultan necesarias tres etapas":

- *La comprobación del siniestro*
- *Su valoración*
- *La liquidación, para proceder a su pago.*

"A fin de comprobar debidamente el siniestro, el asegurador exige una serie de pruebas tendientes a ese objeto y realiza todas las gestiones que considera convenientes para cerciorarse del suceso que se haya cubierto por el seguro". [36]

Inspección

"Acto de realizar visita o revisión del bien por asegurar o asegurado para verificar las condiciones de asegurabilidad y emitir recomendaciones si se considera conveniente".

Capítulo III: Marco Aplicativo

El marco aplicativo abarca la adaptación del proceso de desarrollo XP al caso particular de la aplicación Web a desarrollar, especificando las tareas a realizar dentro de cada una de las fases de dicho proceso.

3.1 Definición de Módulos de la Aplicación Web

El sistema se subdivide en dos módulos, Declaración de Siniestro y Gestión de Documentos cada uno tiene sus funcionalidades, que van hacer detalladas a continuación:

Módulo de Declaración de Siniestro: permite una nueva alternativa a los asegurados a la hora de declarar los siniestros de auto desde la comodidad de su hogar u oficina, ofreciendo el servicio de una forma rápida y fácil. Otra de las funcionalidades de este módulo es que el cliente asegurador podrá ingresar para consultar el status de sus siniestros declarados y además tiene la opción de descargar en archivo pdf la declaración de siniestro realizada por el mismo. Los actores que interactúan con este módulo de la aplicación son dos tipos de clientes: los clientes externos a la empresa y clientes internos de la empresa ambos pueden jugar el papel de asegurados e internamente van hacer tratados de la misma forma.

Módulo de Gestión de Documentos: permite la funcionalidad de anexar a la declaración de siniestro reportada por el cliente, los documentos que deben ser entregados por el cliente para que dicha declaración tome validez, como lo indica la Ley de Seguros establecida por la superintendencia de bancos y también se anexan las fotos de los daños del vehículo con su respectiva información. Los actores que intervienen en este módulo son los siguientes: Analista, Perito y Gerente.

Analista: se encarga de verificar los documentos legales que les exige la superintendencia de bancos, que debe presentar cada asegurador de seguros a la hora de ocurrir algún siniestro. Luego que son verificados se cargan al expediente del cliente a través de la aplicación Web.

Perito: tiene la responsabilidad de realizar la evaluación del vehículo, comparando el reporte declarado por el cliente con el levantamiento de información que se realizó en ese momento del peritaje. Dicha declaración se encuentra en la aplicación al momento de realizar la búsqueda del siniestro. El Perito debe seleccionar en la aplicación las partes afectadas y

anexar las fotografías de los daños, para que luego pueda ser vista por el analista y obtenga los resultados apropiados del siniestro.

Cliente Asegurador: es el ente que cuenta con una póliza de automóvil dentro de Empresa de Seguros, el mismo va a interactuar con el módulo de Declaración de Siniestro, al momento en que su vehículo tenga algún siniestro.

3.2 Diseño de la Aplicación Web

A continuación se detalla el diseño de la aplicación a través de las fases de la metodología XP (Programación Extrema), partiendo de la especificación de la estructura de la base de datos como se puede modelar en el Diagrama Entidad – Relación (E/R).

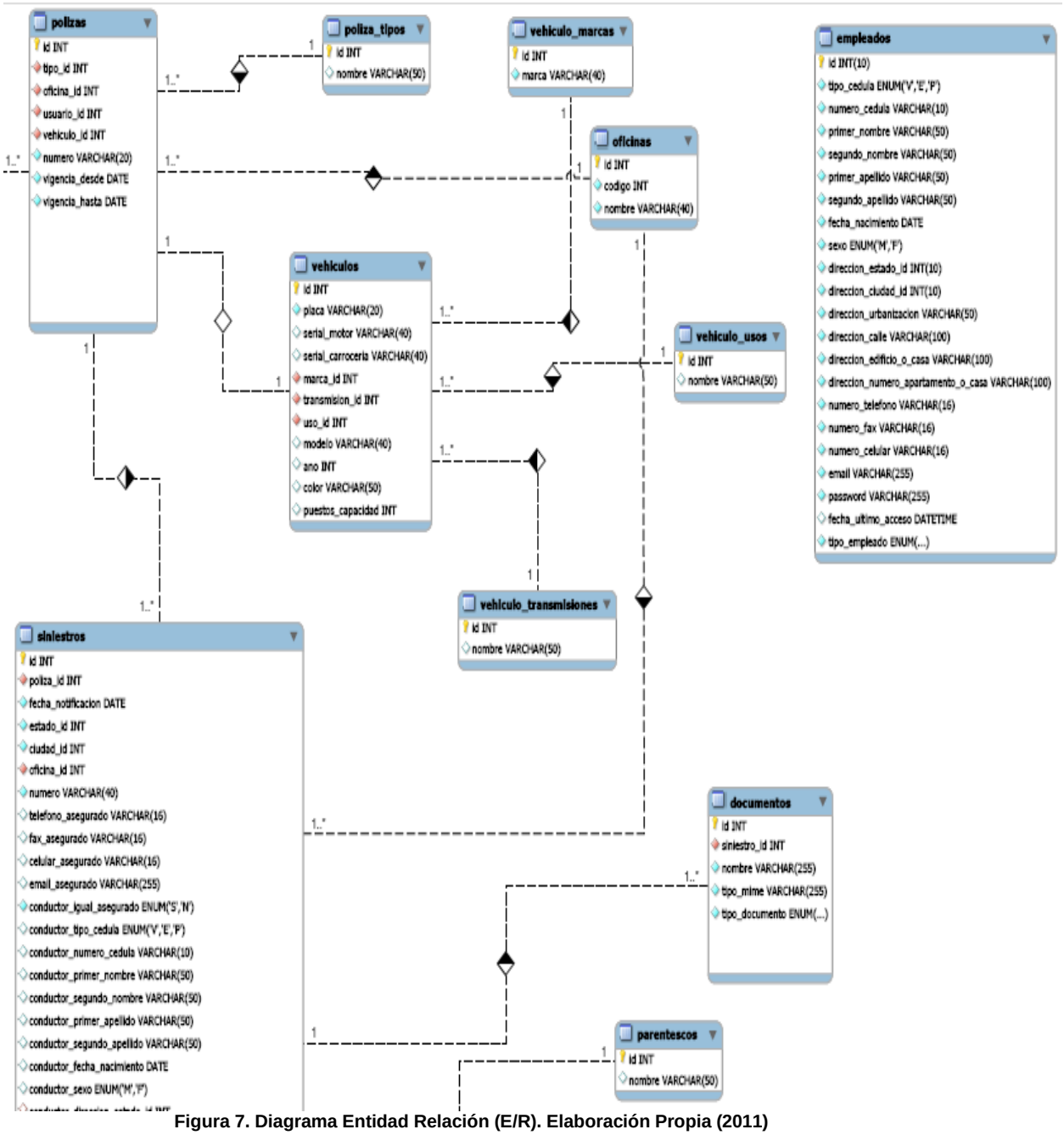
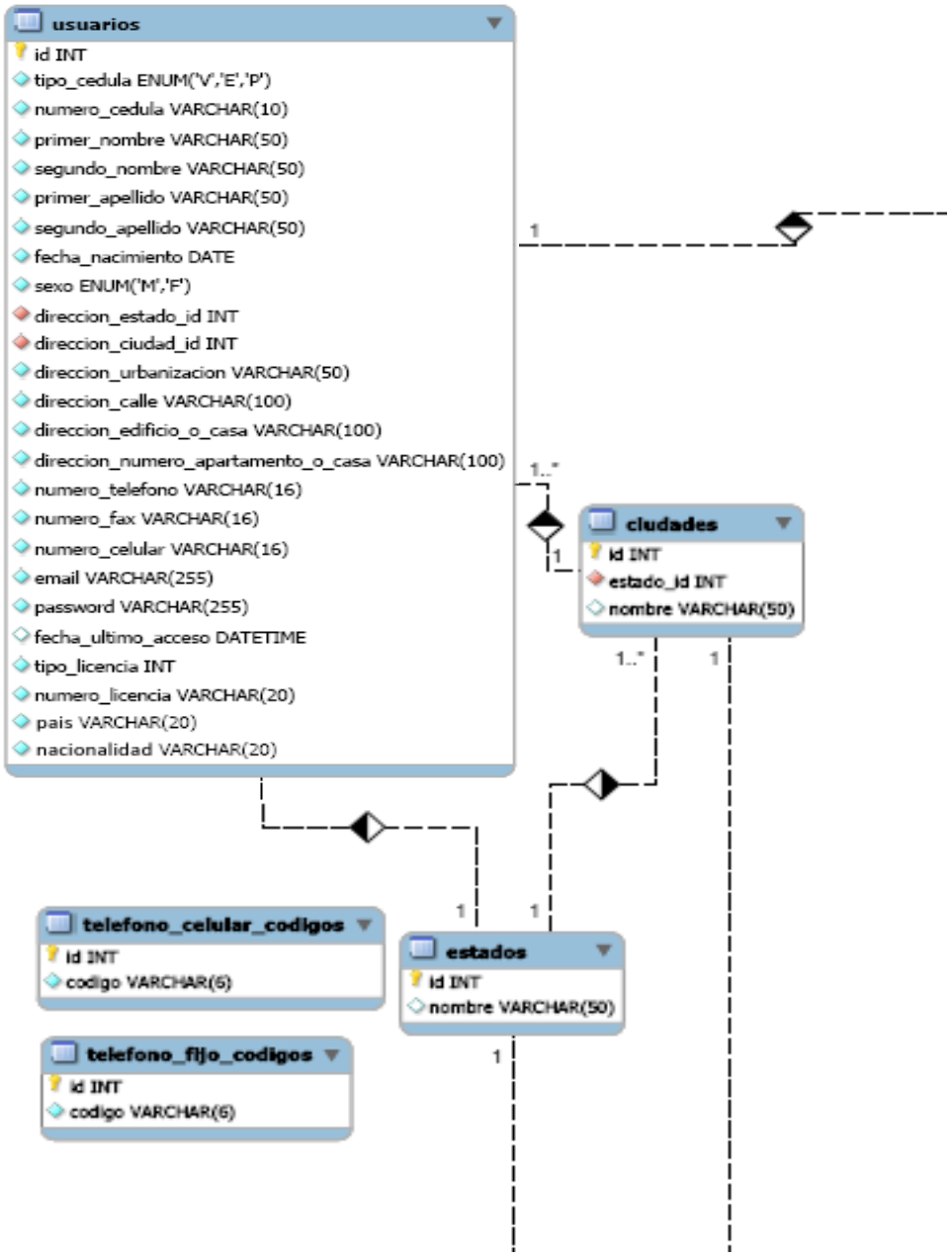
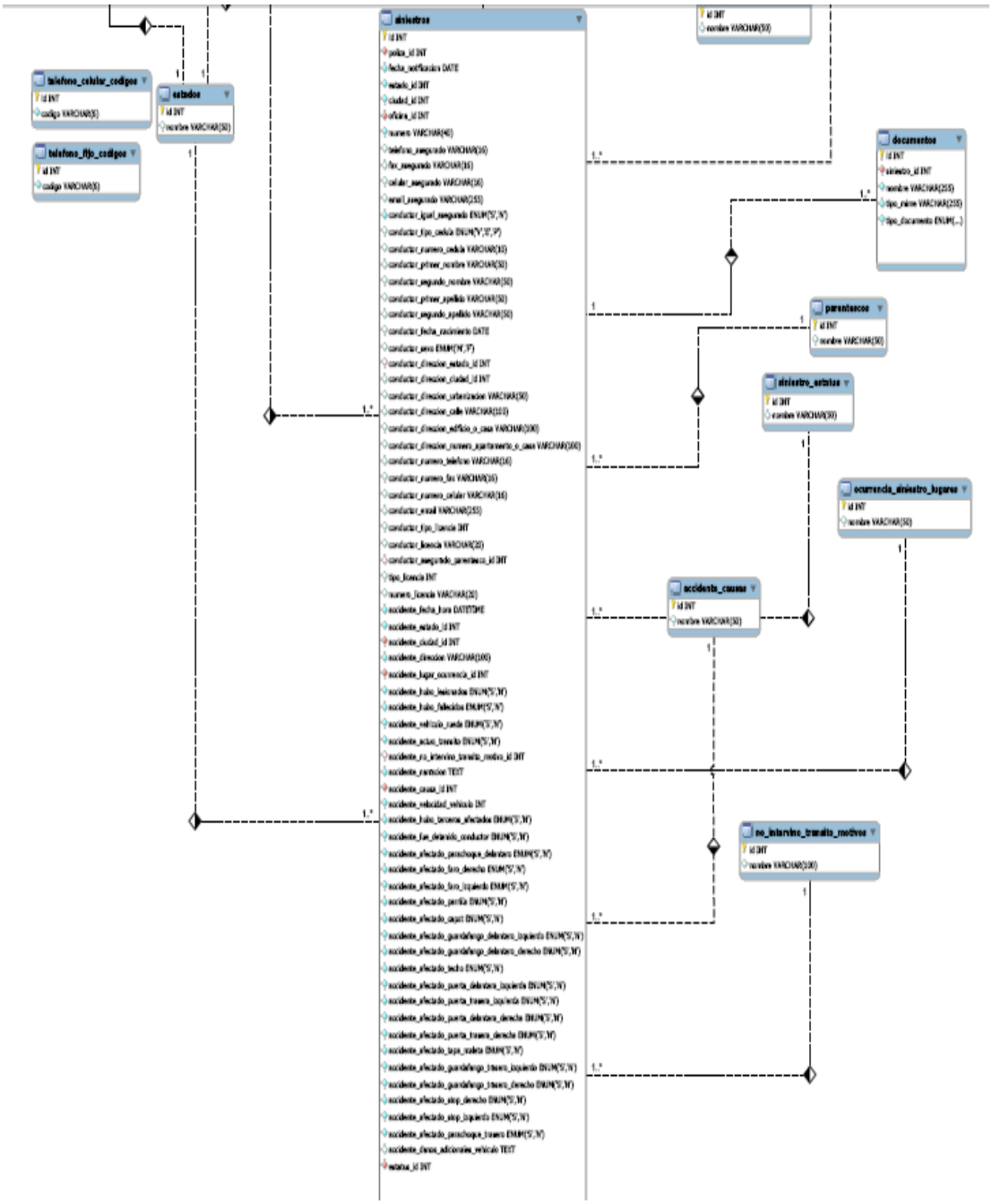


Figura 7. Diagrama Entidad Relación (E/R). Elaboración Propia (2011)


usuarios
id INT
tipo_cedula ENUM('V','E','P')
numero_cedula VARCHAR(10)
primer_nombre VARCHAR(50)
segundo_nombre VARCHAR(50)
primer_apellido VARCHAR(50)
segundo_apellido VARCHAR(50)
fecha_nacimiento DATE
sexo ENUM('M','F')
direccion_estado_id INT
direccion_ciudad_id INT
direccion_urbanizacion VARCHAR(50)
direccion_calle VARCHAR(100)
direccion_edificio_o_casa VARCHAR(100)
direccion_numero_apartamento_o_casa VARCHAR(100)
numero_telefono VARCHAR(16)
numero_fax VARCHAR(16)
numero_celular VARCHAR(16)
email VARCHAR(255)
password VARCHAR(255)
fecha_ultimo_acceso DATETIME
tipo_licencia INT
numero_licencia VARCHAR(20)
pais VARCHAR(20)
nacionalidad VARCHAR(20)
ciudades
id INT
estado_id INT
nombre VARCHAR(50)
telefono_celular_codigos
id INT
codigo VARCHAR(6)
estados
id INT
nombre VARCHAR(50)
telefono_fijo_codigos
id INT
codigo VARCHAR(6)
1
1..*
1
1..*
1
1
1
1


siniestros
telefono_celular_codigos
estados
telefono_ftjo_codigos
documentos
parentescos
siniestro_actores
ocurrencia_siniestro_lugares
accidente_causas
no_interviene_transito_motivos

3.3 Desarrollo de la aplicación Web siguiendo la metodología XP

Después de un análisis de las funcionalidades que tiene el Módulo de Declaración de Siniestro y la Gestión de Documentos (Ver pág.16), se determinó que se podía realizar la elaboración del sistema por cada módulo dos iteraciones que se subdividen en una iteración por cada funcionalidad dentro del módulo, pero hay una primera iteración que se basa en la configuración básica del sistema. El objetivo de cada iteración es obtener una versión del sistema que incluya la implementación de las historias de usuario planteadas en la planificación del proceso de desarrollo XP. Los resultados de las iteraciones son evaluados por el cliente hasta que sean aprobadas por el mismo.

Por cada iteración se aplicarán las cuatro actividades definidas en la adaptación de XP, mostrando la planificación de las historias de usuario que se implementarán, el diseño con diagramas de clases, la codificación y finalmente las pruebas realizadas.

A partir de las necesidades indicadas por la Empresa de Seguros se hace uso de las plantillas especificadas a continuación, para definir las historias de usuario, siguiendo la metodología XP, la misma contiene los siguientes elementos:

- **Número de Historia:** número que identifica la historia.
- **Nombre de la Historia:** texto que identifica la historia.
- **Tipo de Historia:** define el tipo de historia el cual puede catalogarse como técnica o funcional.
- **Descripción:** texto descriptivo de la actividad con el siguiente enunciado: “Como un <Usuario de Sistema> quiero poder <> para justificación”.
- **Importancia:** número que le otorga el nivel de importancia a esta historia utilizando una escala del 1 al 10 siendo el 10 el más importante.
- **Cómo probarlo:** serie de pasos a seguir para verificar que la tarea planteada se ha desarrollado y cumple con el requerimiento o funcionalidad.
- **Horas de desarrollo:** número de horas en la que se estima se complete la actividad.

Se utiliza la plantilla para proceder a definir cada una de las historias de usuarios identificadas por Empresa de Seguros resultando en la lista total y final de funcionalidades que presentemos a continuación en la siguiente tabla:

Iteración I. Declaración de Siniestro

La siguiente iteración se subdividió en varias historias de usuario, que se detallaran a continuación:

Número de Historia: 1

Nombre de la Historia: Configuración Básica del Sistema

Tipo de Historia: Requerimiento

Descripción: como usuario del sistema queremos poder contar con la mejor tecnología del mercado, que nos garantice la satisfacción tanto del cliente asegurado como la de nuestros analistas y peritos que son los que van hacer uso de la aplicación a desarrollarse.

Importancia: 10

Cómo Probarlo:

- Se les organizó una reunión para la explicación de las tecnologías implementadas en el desarrollo.
- Se les mostró la estructura de directorios, conexión con la bases de datos y servidores.
- Analizar el diseño y estructura de la Base de Datos de la aplicación Web.
- Crear los diagramas Entidad / Relación (E/R) y Diagrama de Clases.

Horas de Desarrollo: 20 horas

Número de Historia: 2

Nombre de la Historia: Autenticación de usuario

Tipo de Historia: Funcional

Descripción: como cliente asegurador y empleado de la Empresa de Seguros queremos poder ingresar al sistema para poder realizar nuestras actividades y mantener un registro personalizado de la misma, a su vez si se le olvidó la contraseña al usuario (cliente

asegurador, analista y perito) se le dé la opción de recibir la contraseña por medio del correo electrónico, para mantener la seguridad de la aplicación.

Importancia: 10

Cómo Probarlo:

- Se le solicita al cliente asegurador, analista o perito el nombre de usuario, contraseña, verificación de letras (captcha), luego de completar cada uno de estas funcionalidades se puede ingresar al sistema.
- Se verifica que los datos ingresados sean correctos, de lo contrario se indica un mensaje de error.
- Se verifica que la pantalla de inicio, luego de ingresar en el sistema, sea la correspondiente al tipo de usuario (cliente asegurador, analista y perito).
- Verificar que la contraseña del usuario (cliente asegurador, analista y perito) llegue al correo electrónico.

Horas de Desarrollo: 32 horas

Diseño y Codificación

Para el diseño Web se estableció el ambiente de trabajo conexión con la base de datos para la verificación de los campos cédula y contraseña, el diseño de la interfaz y la capa de negocio. Cómo se basa en la metodología XP el diseño es simple para ahorrar horas de desarrollo. A continuación se muestra el diseño Web con las funcionalidades detalladas anteriormente y parte de su codificación:

Figura 8. Interfaz de Autenticación para Declaración de Siniestro / Gestión de Documentos.

Fuente: Elaboración Propia (2011).

La **codificación** de la siguiente pantalla sería de la siguiente manera:

```
funcion login(){

      $this->layout= 'login';  //usuar la plantilla de login
//si vienen datos en el formulario se procesa el login
   if (!empty($this->data)) {
         //validar formulario
$this->data['Usuario']['cedula'] = '';
$this->Usuario->set($this->data);
/*
* validar el captcha* */

$valido = $this->Usuario->validates();
         if ($valido) {
//verificar que la captcha es correcta
$valor_captcha = $this->Session->read('securimage_code_value');
$this->Session->delete('securimage_code_value');
//si la captcha es incorrecta
//se validan los otros campos del formulario
//y se retorna  la pantalla de login
if (strtolower($this->data['Usuario']['captcha']) != strtolower($valor_captcha)) {
$this->data['Usuario']['captcha'] = '';
$this->data['Usuario']['password'] = '';
$this->set('error_login','<div class="msgerror">El texto ingresado no coincidió con el mostrado en la imagen. Intente nuevamente.</div>');
return;            } else {
//captcha correcta
//se procesa el login
//verificar que es un login válido
$tipo = $this->data['Usuario']['tipo_cedula'];
$numero = $this->data['Usuario']['numero_cedula'];
$password =$this->data['Usuario']['password'];
$login = $this->Usuario->cargarUsuarioLogin($tipo,$numero,$password);
                  if (!empty($login)) {
//guardar la info en sesión
$this->Session->write('Usuario',$login['Usuario']);
$this->Session->write('Usuario.estado',$login['DireccionEstado']['nombre']);
$this->Session->write('Usuario.ciudad',$login['DireccionCiudad']['nombre']);
```

Número de Historia: 3

Nombre la Historia: Declaración de Daños de Vehículo

Tipo de Historia: Funcional

Descripción: como cliente de Empresas de seguros quiero poder ingresar a la página Web y poder realizar la declaración de siniestro de forma rápida y sencilla y que se me muestre la opción de ver planilla en la computadora.

Importancia: 10

Cómo probarlo:

- Se verifica que la información suministrada sea la correcta.
- Se verifica que se le muestre al cliente todos los vehículos asegurados.
- Se verifica que el usuario pueda ingresar a todas las opciones del formulario para realizar la declaración.
- Verificar que se muestren los vehículos asociados al asegurado, que se encuentren afiliados al seguro.
- Se verifica que se genere la planilla con los datos personales, del vehículo del asegurado y la información reportada del siniestro.

Horas de Desarrollo: 66 horas.

Diseño y Codificación

Se puede observa a través del formulario las tablas donde aparecen los distintos vehículos del asegurado que se encuentra en la cobertura de Empresa de Seguros y al seleccionar el vehículo que tuvo el siniestro se presiona el botón de continuar, se activan las pantallas que deben ser completada por el asegurado con la información del siniestro. Las interfaces poseen la siguiente apariencia.

Figura 9. Interfaz de selección del vehículo. Fuente: Elaboración Propia (2011)

Figura 10. Interfaz de Notificación de Datos del Siniestro. Fuente: Elaboración Propia (2011)

A continuación se presentará parte de la **codificación** de las pantallas:

```
<?php
/*
 * Clase controladora que tiene todas las acciones relativas
 * al siniestro
 * */
class SiniestrosController extends AppController {
        var $name = 'Siniestros';
   /*
    * Carga el valor de sesión
    *  que contiene los formularios de declaración
    *  de cada uno de los pasos recorridos
    *   */
   function _getValoresSesionDanos($codigo_sesion)
   {
      /*
       *verificar que viene el código de sesión y es un código válido
       * */
      $valor_sesion = $this->Session->read('DeclaracionDano' . $codigo_sesion);
          if (empty($valor_sesion)) {
         return  null;
      }
      else {
         return $valor_sesion;
      }
   }
        /*
    * Inicia los valores de sesión
```

```
 * luego de seleccionar la póliza
 * para guardar todos los pasos de llenado
 * del formulario de declaración de daños
 *
 * la función retorna el código aleatorio de sesión
 * que se usará para guardar los formularios de cada paso
 * */
function _iniciarSesionDeclaracionDanos($poliza)
{
    //crear un valor aleatorio para
    //guardar las cosas en sesión
    //el valor aleatorio se usa como indice en sesión

    $rand = rand();
    $this->Session->write('DeclaracionDano'. $rand. '.poliza' ,$poliza);
```

Número de Historia: 4

Nombre la Historia: Consulta de Siniestros

Tipo de Historia: Funcional

Descripción: como cliente de Empresas de seguros, quiero poder realizar las consultas de la declaración de siniestros de forma rápida y sencilla y que se me muestre el estatus del mismo, para luego dirigirme a la oficina del seguro y realizar el retiro de la orden de reparación.

Importancia: 10

Cómo probarlo:

- Se verifica que la información suministrada sea la correcta.
- Se verifica que se le muestre al cliente el estatus del requerimiento.
- Se verifica que puedan visualizar y descargar el documento en pdf donde se encuentra el detalle del siniestro declarado.

Horas de Desarrollo: 32 horas.

Diseño y Codificación

Se diseñó la interfaz de Consulta de Siniestros, el cual permite la conexión con la base de datos y mostrar los siniestros asociados al cliente asegurado y anexar el documento en pdf. A continuación se muestra parte de la lógica de negocio.

Figura 11. Consulta de Siniestro. Fuente Elaboración Propia (2011)

```
function consulta_siniestros()
   {       //cargar lista de siniestros
      $siniestros = $this->Siniestro->cargarListaResumenSiniestro($this->Session->read('Usuario.id'));
           //setear valores para la vista
      $this->set('siniestros' , $siniestros);
      $this->set('n_siniestros' , count($siniestros));
          $this->set('usuario',$this->Session->read('Usuario'));
      $this->set('fecha_ultimo_acceso',$this->Session->read('fecha_ultimo_acceso'));
      //migajas de pan
      $crumbs = Array(
          'Inicio' => Array('link' => '/usuarios/'),
          'Consulta de Siniestros' => Array('link' => '/siniestros/consulta_siniestros'),
      );
      $this->set('breadcrumbs',$crumbs);
   }
   /*
    * Pantalla con el detalle del siniestro.
    * Ese el mismo contenido del pdf pero en pantalla
    * */
   function detalle($id = 0)
   {
  if (empty($id)) {
          $this->redirect('/usuarios/index/') }
```

Culminación de la Iteración I

Esta primera iteración no fue completada, debido a que cuando se realizaron las pruebas con el cliente, el mismo indicó faltan realizar algunas validaciones y quiere otra funcionalidad dentro de la declaración de los daños por lo tanto se debe realizar otra iteración para poder culminar.

A continuación la figura 12 se presenta la evaluación realizada para las historias de usuarios correspondiente a este módulo. La escala de evaluación es de la siguiente forma: 1-Excelente, 2- Muy Bien, 3- Bien, 4- Suficiente, 5- Malo

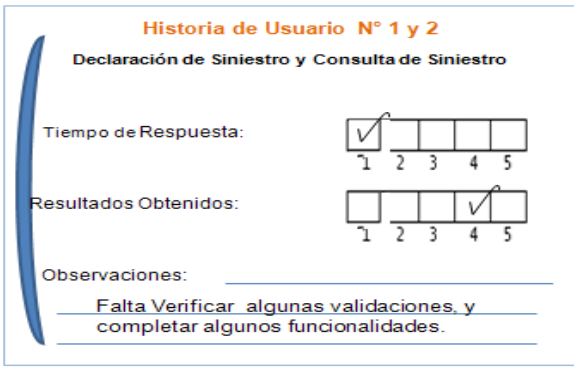


Figura 12. Evaluación de la Historia de Usuario. Fuente: Elaboración Propia (2011)

En las siguientes iteraciones, se podrán observar las historias de usuarios correspondientes a la funcionalidad de Gestión de Documentos.

Iteración II. Gestión de Documentos

El objetivo de la iteración es obtener una versión que incluya la implementación de las historias de usuarios, planteadas en la planificación del proceso de desarrollo XP. El resultado de la iteración fue evaluada por el cliente para su aprobación.

Número de Historia: 5

Nombre la Historia: Gestión de Documentos (Analista)

Tipo de Historia: Funcional

Descripción: como analista quiero poder cargar los documentos asociados al siniestro reportado por el cliente asegurador.

Importancia: 10

Cómo probarlo:

- Se verifica que los documentos que se van a cargar en la aplicación Web sean los correctos.
- Se verifica que se haya anexado el siniestro declarado por el cliente asegurado en la sesión del analista.
- Se verifica que el sistema haya registrado los cambios.

Horas de Desarrollo: 40 horas.

Diseño y Codificación

El Analista para poder adjuntar los documentos asociados al siniestro reportado por el cliente asegurado, debe realizar primero la búsqueda por el número de siniestro o por el número de cédula del cliente, posteriormente se habilita la pantalla que permite la opción de subir los documentos del cliente asegurado, que van a permitir legalizar el proceso de declaración del siniestro. A continuación se presenta las pantallas de la funcionalidad de Adjuntar documento.

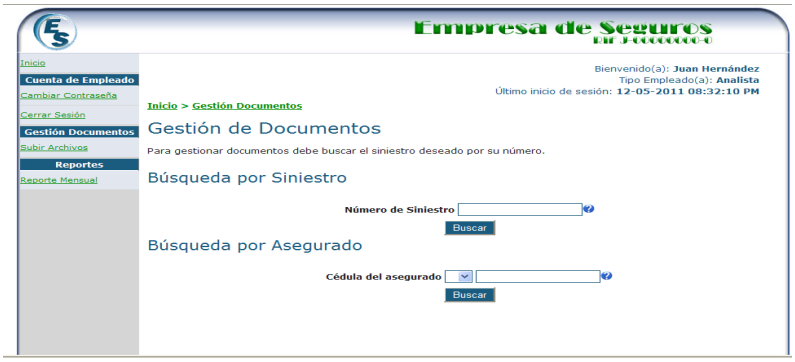


Figura 13. Interfaz Ingresar al Módulo de Gestión de Documentos. Fuente: Elaboración Propia (2011)

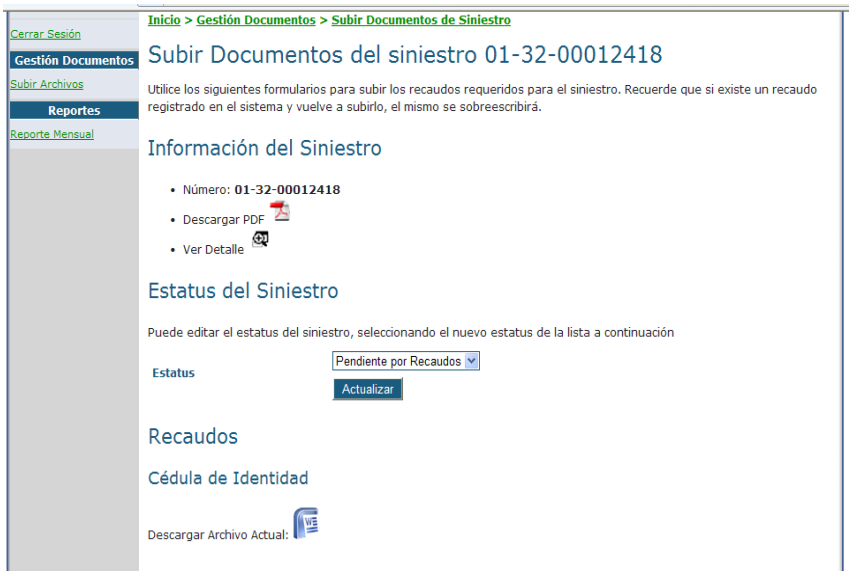


Figura 14. Interfaz de los Documentos Cargados por el Analista. Fuente: Elaboración Propia (2011)

La **codificación** es la siguiente:

```
function peritaje_siniestro()
  {
    //si el empleado no es perito no se le permite el acceso
    if ($this->Session->read('Empleado.tipo_empleado') != 'perito') {
       $this->redirect('/empleados/index');
       return;
    }
    //mensajes de error del buscador
    $error_buscar_siniestro = '';
    $error_buscar_cedula = '';

    //ver si viene data del formulario
    if (!empty($this->data)) {
       //verificar si viene data de cedula o de
       //siniestro
       if ($this->data['Siniestro']['busqueda'] == 'siniestro') {
          /*
           * Búsqueda por Siniestro
           * */
          //normalizar el número de siniestro, en caso de que el usuario
          //lo haya ingresado sin guiones
          $num_siniestro_sin_guiones = str_replace('-','',$this->data['Siniestro']['numero']);

          $s1 = substr($num_siniestro_sin_guiones,0,2);
          //rellenar con cero el numero de oficina
          if (strlen($s1) == 1) {
             $s1 = '0' . $s1;
          }
```

```
            $s2 = substr($num_siniestro_sin_guiones,2,2);
            //rellenar con cero el tipo de declaración
            if (strlen($s2) == 1) {
               $s2 = '0' . $s2;
            }
            $s3 = substr($num_siniestro_sin_guiones,4);

            $num_siniestro = "$s1-$s2-$s3";

            //verificar si el siniestro está en BD
            $id_siniestro = $this->Siniestro->cargarIdSiniestro($num_siniestro);

            if (!empty($id_siniestro)) {
               //redirigir a resultado
               $this->redirect('/siniestros/form_peritaje/' . $id_siniestro);
               return;
            } else {
               $error_buscar_siniestro = "El siniestro número <strong>$num_siniestro</strong> no está registrado en el sistema.";
            }
```

Número de Historia: 6

Nombre la Historia: Gestión de Documentos (Perito)

Tipo de Historia: Funcional

Descripción:

- Se verifica que los documentos que se van a cargar en la aplicación Web sean los correctos.
- Se verifica que se haya anexado el siniestro declarado por el cliente asegurado en la sesión del analista.
- Se verifica que el sistema haya registrado los cambios.

Horas de Desarrollo: 40 horas.

Diseño y Codificación

El Perito para poder adjuntar los documentos asociados al siniestro reportado por el cliente asegurado, debe realizar primero la búsqueda por el número de siniestro o por el número de cédula del cliente, posteriormente se habilita la pantalla que permite la opción de subir las fotos de los daños del vehículo del cliente asegurado a su vez el Perito debe verificar los campos que se encuentran seleccionados con las partes afectadas del vehículo,

estas casillas de encuentran seleccionadas porque se captura dicha información de la declaración de siniestro realizada por el cliente asegurado.

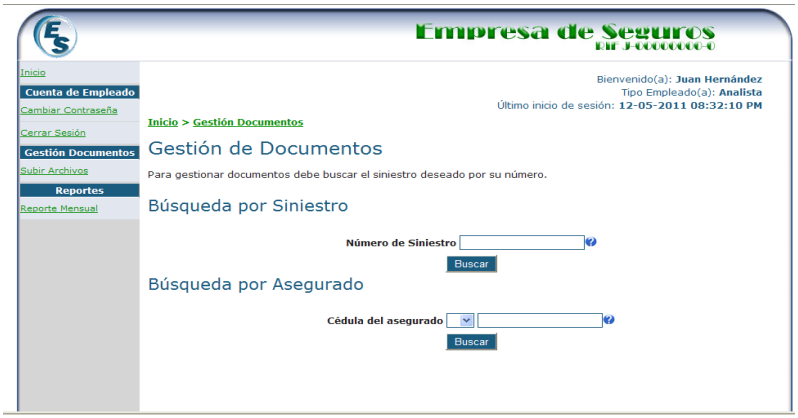


Figura 15. Búsqueda del Cliente Gestión de Documentos. Fuente: Elaboración Propia (2011)

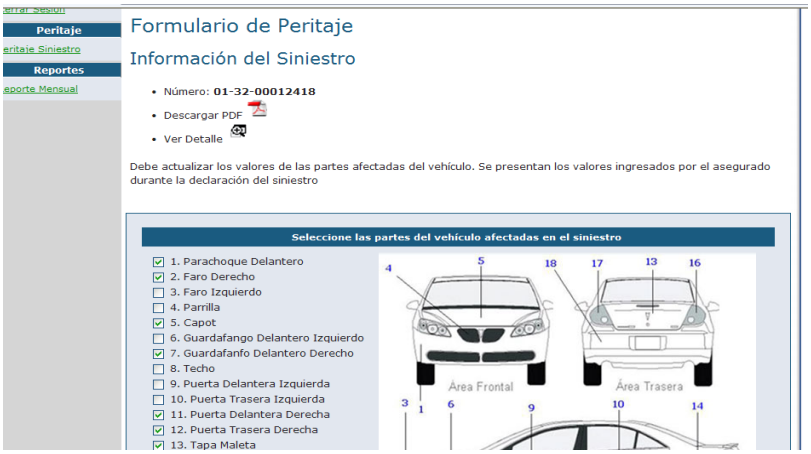


Figura 16. Interfaz de Peritaje. Fuente: Elaboración Propia (2011)

Codificación

```
function gestion_documentos()
  {
    //si el empleado no es analista no se le permite el acceso
    if ($this->Session->read('Empleado.tipo_empleado') != 'analista') {
       $this->redirect('/empleados/index');
       return;
    }
    //mensajes de error del buscador
    $error_buscar_siniestro = '';
    $error_buscar_cedula = '';
```

```
  //ver si viene data del formulario
 if (!empty($this->data)) {
     //verificar si viene data de cedula o de
     //siniestro
     if ($this->data['Siniestro']['busqueda'] == 'siniestro') {
        /*
         * Búsqueda por Siniestro
         * */
        //normalizar el número de siniestro, en caso de que el usuario
        //lo haya ingresado sin guiones
        $num_siniestro_sin_guiones = str_replace('-','',$this->data['Siniestro']['numero']);
        $s1 = substr($num_siniestro_sin_guiones,0,2);
        //rellenar con cero el numero de oficina
        if (strlen($s1) == 1) {
           $s1 = '0' . $s1;
        }
        $s2 = substr($num_siniestro_sin_guiones,2,2);
        //rellenar con cero el tipo de declaración
        if (strlen($s2) == 1) {
           $s2 = '0' . $s2;
        }
        $s3 = substr($num_siniestro_sin_guiones,4);
        $num_siniestro = "$s1-$s2-$s3";
        //verificar si el siniestro está en BD
        $id_siniestro = $this->Siniestro->cargarIdSiniestro($num_siniestro);
        if (!empty($id_siniestro)) {
           //redirigir a resultado
           $this->redirect('/siniestros/subir_documentos1/' . $id_siniestro);
           return;
        } else {
           $error_buscar_siniestro = "El siniestro número <strong>$num_siniestro</strong> no está registrado en
el sistema.";
        }
```

Número de Historia: 7

Nombre la Historia: Reportes

Tipo de Historia: Funcional

Descripción: como un usuario de Empresa de Seguros (Analista y Perito) quiero poder realizar las consultas estadísticas de todos los siniestros reportados.

Importancia: 10

Cómo probarlo:

- Verificar que la información suministrada sea la correcta.
- Verificar que el reporte quede registrado en el sistema.

Horas de Desarrollo: 60 horas

Diseño y Codificación

En la siguiente interfaz se pueden observar los siniestros reportados. A continuación se muestra parte de la lógica de negocio.

Figura 17. Interfaz de Reportes de Siniestro. Fuente: Elaboración Propia (2011)

Codificación:

```
 /*
   * Acción para el reporte mensual de siniestralidad
   * */
  function reportes()
  {

     $mensaje = '';
     //si viene data del formulario se procesa el reporte
     if (!empty($this->data)) {
        $reporte = null;
          //validar el formulario
        $mes_desde = trim($this->data['Siniestro']['mes_desde']);
        $mes_hasta = trim($this->data['Siniestro']['mes_hasta']);
        $ano_desde = trim($this->data['Siniestro']['ano_desde']);
        $ano_hasta = trim($this->data['Siniestro']['ano_hasta']);
         $valido = ($mes_desde != '') & ($mes_hasta != '') && ($ano_desde != '') && ($ano_hasta != '') &&
         !(($mes_desde == $mes_hasta) && ($ano_desde == $ano_hasta));
 if (!$valido) {
           $mensaje = 'Debe seleccionar el mes inicial y final del reporte, y los mismos deben ser distintos';
         }
         $reporte = $this->Siniestro->reporteMensual($mes_desde,$ano_desde,$mes_hasta,$ano_hasta)
}else {
         $reporte = null;
```

}

Culminación de la Iteración II

Esta segunda iteración no fue completada, debido a que cuando se realizaron las pruebas con el cliente, el mismo indicó que faltan realizar algunas validaciones y quiere agregarle otras funcionalidades como que el analista pueda observar las fotos anexadas por el perito y agregar otros reportes dentro del módulo de Gestión de Documentos, por lo tanto se debe realizar otra iteración para poder culminar.

A continuación, la figura 18 presenta la evaluación realizada para las historias de usuarios correspondiente a este módulo. La escala de evaluación es de la siguiente forma: 1-Excelente, 2- Muy Bien, 3- Bien, 4- Suficiente, 5- Malo

A continuación se presenta la evaluación realizada para la Historia de Usuario correspondiente al módulo de Gestión de Documentos.

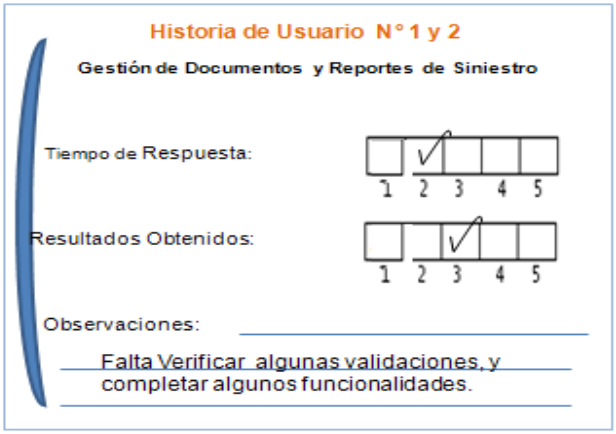


Figura 18. Evaluación de las Historias de Usuario. Fuente: Elaboración Propia (2011)

Iteración III. Declaración de Siniestro y Gestión de Documentos.

Número de Historia: 8

Nombre la Historia: Generar documento pdf.

Tipo de Historia: Funcional

Descripción: como cliente de Empresas de seguros, quiero poder realizar la consulta del siniestro declarado por el cliente en formato pdf y que se pueda imprimir y guardar, con sus respectivas validaciones.

Importancia: 8

Cómo probarlo:

- Se verifica que la información suministrada sea la correcta.
- Se verifica que se le muestren las validaciones correspondientes.
- Se verifica que puedan visualizar y descargar el documento en pdf donde se encuentra el detalle del siniestro declarado.

Horas de Desarrollo: 23 horas.

Diseño y Codificación

En la siguiente interfaz se puede observar el documento en pdf.

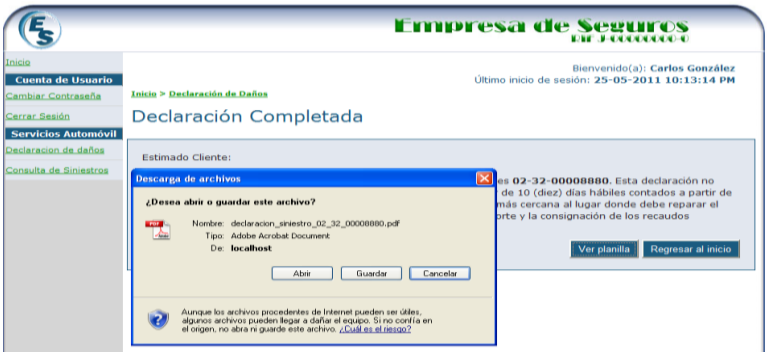


Figura 19. Generación de la Declaración de Siniestro en pdf. Fuente: Elaboración Propia (2011)

Codificación

```
<h1>Información del siniestro <?php echo $num_siniestro ?></h1>

<table class="resumen" cellspacing="0" style="font-size: 8pt; width: 97%; border: 2px solid #000000" >
   <tr>
      <th colspan="4"> DATOS DE LA DECLARACIÓN DEL SINIESTRO </th>

   </tr>
</table>
<table class="resumen" cellspacing="0" style="font-size: 8pt; width: 97%; border: 2px solid #000000" >

   <tr>
      <th colspan="4"> DATOS DE LA PÓLIZA </th>
   </tr>
   <tr>
      <td>Sucursal: <strong><?php echo $sucursal ?></strong></td>
      <td>Ramo: <strong><?php echo $ramo ?></strong></td>
      <td>Poliza: <strong><?php echo $numero ?></strong></td>
      <td>Fecha Notificación: <strong><?php echo $fecha_notificacion . ' ' . $hora_notificacion ?></strong></td>
   </tr>
</table>
<table class="resumen" cellspacing="0" style="font-size: 8pt; width: 97%; border: 2px solid #000000" >
   <tr>
      <th colspan="2"> DATOS DEL ASEGURADO </th>
   </tr>
   <tr>
      <td>Apellidos y Nombres:</td>
      <td><strong><?php echo $apellidos_asegurado  ?> <?php echo $nombres_asegurado ?></strong></td>
   </tr>
   <tr>
      <td>Cédula de Identidad o Pasaporte: </td>
      <td><strong><?php echo $cedula_asegurado ?></strong></td>
   </tr>
</table>
<table class="resumen" cellspacing="0" style="font-size: 8pt; width: 97%; border: 2px solid #000000" >
   <tr>
      <th colspan="6"> DATOS DEL VEHÍCULO ASEGURADO </td>
   </tr>
   <tr>
      <td>Marca:</td>
      <td><strong><?php echo $vehiculo_marca ?></strong></td>
      <td>Modelo</td>
      <td><strong><?php echo $vehiculo_modelo ?></strong></td>
      <td>VERSIÓN:  </td>
      <td><strong><?php echo $vehiculo_version ?></strong></td>
   </tr>
   <tr>
      <td>Año:</td>
      <td><strong><?php echo $vehiculo_ano ?></strong></td>
```

```
        <td>Transmisión</td>
        <td><strong><?php echo $vehiculo_transmision ?></strong></td>
        <td>Color</td>
        <td><strong><?php echo $vehiculo_color ?></strong></td>
    </tr>
    <tr>
        <td>Serial Motor:</td>
        <td><strong><?php echo $vehiculo_serial_motor ?></strong></td>
        <td>Serial Carrocería: </td>
        <td><strong><?php echo $vehiculo_serial_carroceria ?></strong></td>
        <td> </td>
        <td> </td>
    </tr>
    <tr>
        <td>Placa:</td>
        <td><strong><?php echo $vehiculo_placa ?></strong></td>
        <td> </td>
        <td> </td>
        <td>Uso:</td>
        <td><strong><?php echo $vehiculo_uso ?></strong></td>
    </td>
</tr>
</table>
```

Número de Historia: 9

Nombre la Historia: Generar Reportes

Tipo de Historia: Funcional

Descripción: como cliente de Empresas de seguros quiero poder hacer reportes sobre los clientes que declaran más siniestros y que el analista pueda ver las fotos que carga el perito.

Importancia: 9

Cómo probarlo:

- Se verifica que la información suministrada sea la correcta.
- Se verifica que se muestren los reportes.
- Se verifica que el usuario pueda ingresar a todas las opciones del formulario para realizar la declaración.
- Verificar que el analista pueda visualizar las fotos cargadas por el perito.

Horas de Desarrollo: 66 horas.

Diseño y Codificación

Figura 20. Interfaz de Reportes. Fuente: Elaboración Propia (2011)

Codificación

```
<div class="msginfo">
        Reporte de frecuencia de partes afectadas sobre todo el
        histórico de siniestros
      </div>
<?php
}
?>
<?php
if ($maximo != $minimo) {
?>
<h2>Resumen de Estadística</h2>
<ul>
   <li>Frecuencia Máxima: <strong><?php echo $maximo ?></strong></li>
   <li>Partes afectadas con frecuencia máxima: <strong><?php echo implode(', ', $partes_maximo) ?></strong></li>
   <li>Frecuencia Mínima: <strong><?php echo $minimo ?></strong></li>
   <li>Partes afectadas con frecuencia mínima: <strong><?php echo implode(', ', $partes_minimo) ?></strong></li>
</ul>

<br />
<?php
}
?>

<h2>Tabla de frecuencias de partes afectadas</h2>
<?php
if (!is_null($reporte) && !empty($reporte)) {
```

```
?>
   <?php
   if (!$excel) {
      //reporte excel
   ?>

      <div style="border: none; text-align: center; width: 550px; margin: auto;">
      <table class="resumen" style="width: 500px;" id="tablareporte">
         <tr>
            <th>Parte</th>
            <th>Frecuencia</th>
         </tr>
         <?php
         foreach ($reporte as $p => $n) {
         ?>
         <tr>
            <td stlye="text-align: center;"><?php  echo $nombre_partes[$p]?></td>
            <td stlye="text-align: center;"><?php echo $n ?></td>
         </tr>
         <?php
         }
         ?>
```

Figura 21. Interfaz de carga de la foto del siniestro. Fuente: Elaboración Propia (2011)

Codificación

```
<?php
   if (!empty($documentos['cedula'])) {
   ?>
   <div>
      Descargar Archivo Actual:

      <?php
      /*sacar el ícono para el link de descarga*/
      $tipo = $documentos['cedula']['Documento']['tipo_mime'];
```

```
$id_documento = $documentos['cedula']['Documento']['id'];

if (empty($iconos_extensiones[$tipo])) {
    $icon = 'icon_archivo.gif';
} else {
    $icon = $iconos_extensiones[$tipo];
}
echo " ";
echo $this->Html->link($this->Html->image($icon, Array('alt' => 'Descargar Documento', 'style' => 'border: none;')), '/documentos/descargar/' . $id_documento , Array('escape' => false, 'class' => 'linkdescarga'));
 ?>
```

Número de Historia: 10

Nombre la Historia: Buscar por fecha de notificación

Tipo de Historia: Funcional

Descripción: como cliente de Empresas de seguros quiero poder hacer la búsqueda de los siniestros por fecha.

Importancia: 9

Cómo probarlo:

- Se verifica que la información suministrada sea la correcta.
- Se verifica que se muestren los reportes.
- Se verifica que el usuario pueda ingresar a todas las opciones del formulario para que el analista y perito puedan realizar su levantamiento del siniestro.

Horas de Desarrollo: 15 horas.

Figura 22. Búsqueda de los siniestros. Fuente: Elaboración Propia (2011)

Codificación

```
<h1>Gestión de Documentos</h1>

   <p>
   Para gestionar documentos debe buscar el siniestro deseado por su número.
   </p>


<div>
   <h2>Búsqueda por Número de Siniestro</h2>

   <?php
   if (!empty($error_buscar_siniestro)) {
   ?>
      <div class="msgwarning">
         <?php echo $error_buscar_siniestro  ?>
         </div>
   <?php
   }
   ?>
   <div style="text-align: center;">
      <?php
      echo $this->Form->create('Siniestro',Array('id' => 'formsiniestro'));
      echo $this->Form->label('Siniestro.numero' , 'Número de Siniestro');
      echo "<br />";
       echo $this->Html->link($this->Html->image('icon_ayuda.gif', Array('alt' => 'Ayuda')),'#',Array('id' => 'ayudasiniestro', 'escape' => false, 'style' => 'border: none;')) ;
      echo $this->Form->text('Siniestro.numero');
      echo $this->Form->hidden('Siniestro.busqueda', Array('value' => 'siniestro'));
      echo $this->Form->submit('Buscar');
```

```
        echo $this->Form->end();
        ?>
    </div>

</div>

<div>
    <h2>Búsqueda por Fecha de Notificación de Siniestro</h2>

<?php
        if (!empty($error_buscar_fecha)) {
?>
<div class="msgwarning">
    <?php echo $error_buscar_fecha  ?>
</div>
<?php
        }
?>
```

Culminación de la Iteración III (Muerte)

En esta tercera iteración se puede culminar satisfactoriamente con la aprobación del cliente, cumple con los requerimientos, satisfacción, rendimiento y confiabilidad exigidos por el cliente Empresa de Seguros, como no se tienen más historias de usuarios para ser incluidas en el sistema, se procede a realizar la documentación y se da por culminado dicho sistema.

A continuación, la figura 23 se presenta la evaluación realizada para las historias de usuarios correspondiente a este módulo. La escala de evaluación es de la siguiente forma: 1-Excelente, 2- Muy Bien, 3- Bien, 4- Suficiente, 5- Malo

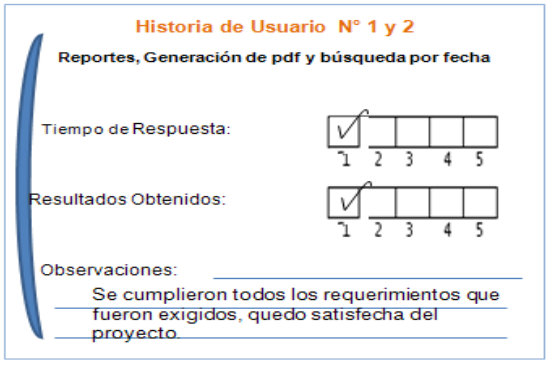


Figura 23. Prueba de Aceptación. Fuente: Elaboración Propia (2011)

.3.4 Tecnologías y librerías involucradas en el desarrollo de la Aplicación Web

Las tecnologías que permitieron el desarrollo de esta aplicación Web fueron las siguientes:

Por ser un desarrollo basado en Web, es necesario contar con un servidor HTTP, un servidor de aplicaciones y un servidor de Base de Datos. Se decidió hacer uso de las herramientas Apache Web Server Versión 2.2.17 como servidor HTTP, PHP *hypertext Processor* Versión 5.3.5 como servidor de aplicaciones en la capa de negocio y MySql *Database Server* Versión 5.5.8 como servidor de Base de Datos en la capa de persistencia. Los tres servicios han sido instalados en el servidor de desarrollo utilizando un paquete que se conoce como WampServer Versión 2.1. Para agilizar el proceso de codificación se seleccionó el framework CakePHP en su versión 1.3 este ambiente de desarrollo de aplicaciones Web en PHP bajo el patrón de diseño de MVC (Modelo - Vista - Controlador).

Para la programación de las interfaces de usuario, se usaron las siguientes librerías:

Librerías de PHP

TCPDF: es una librería usada para generar archivos PDF en php. La librería se encuentra ubicada en el directorio “Vendors” dentro de cada uno de los sitios (panel_usuarios y panel_empleados).

- **Securimage:** es una librería usada para crear imágenes captcha. Esta librería genera una imagen aleatoria con un texto distorsionado. El color, la distorsión, el fondo, etc. Son configurables. La librería se encuentra ubicada en el directorio “Vendors” dentro de cada uno de los sitios (panel_usuarios y panel_empleados).
- **CakePHP: versión 1.3:** es un framework para desarrollar aplicaciones web en PHP, la misma se basa en el patrón de diseño MVC. Este framework ayuda a desarrollar de manera más rápida y confiable, dando asistencia para el manejo de BD, plantillas, etc.
- Librerías JavaSript
- **jQuery:** se usó la biblioteca jquery como base para facilitar la manipulación del DOM a nivel del cliente en javascript y diversas librerías o plugins basados en jquery para varias funcionalidades, como interactuar con los documentos HTML,

manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas Web.

- Las diversas funcionalidades dinámicas de la aplicación, se implementaron utilizando plugins de jquery, las mismas son pequeñas extensiones a la librería para ejecutar una tarea en específico. La lista de plugins usados se muestra a continuación:
- **jQuery Datepick:** es un componente seleccionador de fechas tipo almanaque o "Date Picker", el mismo se puede utilizar en los formularios para asistir al usuario a ingresar una fecha. El objetivo de este plugin es que el usuario seleccione la fecha de manera sencilla sin necesidad de teclearla manualmente, la misma reduce la probabilidad de errores de validación.
- **jQuery Colorize:** es un componente que permite fácilmente crear un esquema de colores alternados para las filas de una tabla creada con el elemento.
- **jQuery MaxLength:** es un plugin que permite restringir la longitud máxima de entrada en los campos de texto de los formularios.
- **jQuery JStepper:** es un plugin que ayuda a restringir entradas sólo numéricas en campos de texto de los formularios.
- **jQuery Alphanumeric:** es un plugin que ayuda a restringir entradas alfabéticas en los campos de texto de los formularios.
- **jQuery Simple Modal:** un plugin para mostrar ventanas modales hechas a base de un contenedor div de html.
- **jQuery SimpleTip:** es un plugin para mostrar tooltips sobre cualquier elemento los cuales se activan al posicionar el puntero del mouse sobre el elemento.
- **jQuery Popup Windows:** es un plugin para lanzar ventanas emergentes (popup) a partir de un link.

Librerías de CSS

- **Blueprint CSS:** es un framework CSS diseñado para reducir los tiempos de desarrollo y mejorar la compatibilidad entre los distintos navegadores web cuando se trabaja con los estilos en cascada (CSS). También sirve como base para muchas herramientas que permiten el desarrollar CSS fácilmente y de una manera más accesible para principiantes.

A continuación se presenta el funcionamiento de CakePHP en la aplicación cuya ejecución será llevada a cabo siguiendo el patrón de diseño MVC (Modelo – Vista - Controlador).

Modelo – Vista - Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

El funcionamiento MVC en CakePHP es el siguiente:

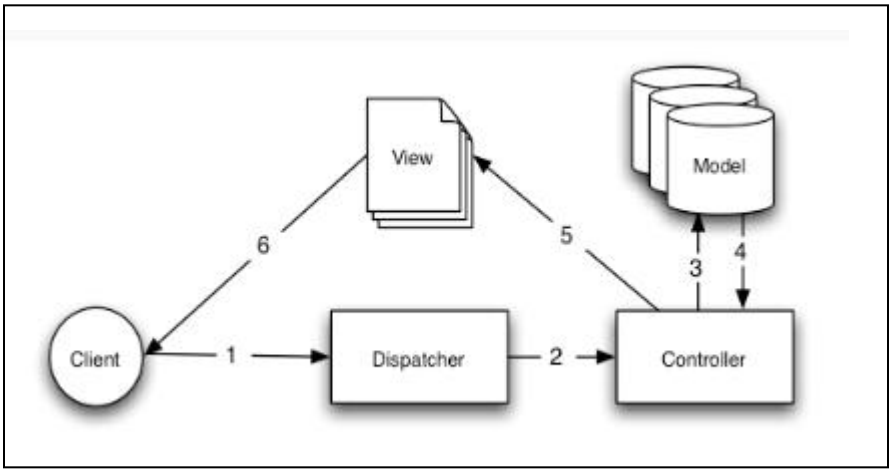


Figura 24. MVC en el Framework CakePHP Fuente: book.CakePHP

De esta forma se puede visualizar la interacción del modelo MVC dentro la aplicación:

1. El usuario ingresa la dirección en el navegador (por ejemplo http://localhost/sitio/panel_usuarios/usuarios/login El framework CakePHP actúa como un despachador haciendo lo siguiente:
2. El framework actúa como un despachador y saca de la URL la base: **http://localhost/sitio/panel_usuarios** y la acción usuarios/login
3. A partir de la acción se determina el Controlador: **Usuarios** por lo que se busca en el directorio **controllers** la clase **usuarios_controller.php** y la página buscada es **login**, por lo que se invoca la función **login()** dentro del controlador.

4. La función **login()** ejecuta la lógica que se desee. En el caso del login puede ser validar la contraseña, por lo que carga el login y contraseña del formulario, y consulta con el modelo **Usuario**, e mismo está definido en el directorio **models** archivo **usuario.php**. En el modelo se escriben las funciones de lógica de acceso a Base de Datos, la misma es usada por el controlador para tomar las decisiones del contenido a mostrar.
5. En la carpeta **Models** se encuentran todas las consultas a la Base de Datos y acciones como validar formularios o cualquier otra que sea necesaria para la aplicación.
6. Luego de usar el modelo para hacer la lógica, el controlador carga una vista, que no es más que una plantilla de HTTP con PHP, en donde está la página web que se desea mostrar. En nuestro ejemplo la vista se cargaría del directorio **views\usuarios** en particular el archivo **login.ctp** donde corresponde a la acción **login()** del controlador. Un archivo **ctp** en realidad es un archivo con PHP y HTML por lo que se puede abrir con cualquier editor de texto igual que se trabajan los PHP. El controlador carga esta vista y sustituye los valores generados luego de hacer la lógica con el modelo.
7. Finalmente luego de que se ejecuta la acción del controlador, el framework toma la vista (**Views**) con todos los valores y la emite a la salida del navegador Web como una página.

Conclusiones

Finalmente con este Trabajo Especial de Grado, se puede concluir que se lograron los objetivos principales de la investigación. En el levantamiento de información que se le realizó previamente al departamento de siniestros se pudo constatar la deficiencia en la que llevaban el proceso de la declaración de siniestros y a su vez como eran tratados los documentos asociados al siniestro reportado por el cliente asegurado; con esta problemática que se encontró en dicho departamento se desarrolló el sistema de Declaración de Siniestro, con estos módulos se llegó a la automatización de los procesos en que los clientes asegurados podrán realizar su declaración de los daños a través de la aplicación Web, disminuyendo así las llamadas del Call Center y se evitarán dirigirse a las oficinas del seguro. Por otra parte contará con el módulo de Gestión de Documentos que permitirá que la Empresa de Seguros automaticé el proceso de los documentos que se manejan por cada cliente asegurado.

Las funcionalidades que poseen cada uno de estos módulos son los siguientes: la Declaración de Siniestro permite la creación de daños del vehículo, creación del reporte de siniestro en pdf, generar consultas de estatus del siniestro vía Web y Gestión de Documentos se crearon las opciones de anexar los documentos asociados al siniestro, ver los reportes estadísticos de los siniestros, consultar las partes de los vehículos que han sido más afectadas en los siniestros y ver los clientes que han tenido más siniestros dentro de Empresa de Seguros, con la automatización de estos procesos se quiere lograr la satisfacción del cliente.

Otro punto importante, es el proceso de desarrollo de la aplicación Web, la misma fue desarrollada mediante las tecnologías: JavaScript, Ajax, jQuery, PHP bajo el servidor de Apache y manteniendo la persistencia con la Base de Datos MySql, también se seleccionó un framework de desarrollo CakePHP bajo el patrón de diseño de MVC, el mismo permite separar la problemática y atacar los problemas de cada una en forma más independiente del resto de las capas (presentación, negocio y persistencia), permitiendo así trabajar y avanzar con el desarrollo en forma paralela una vez que se especificaron las interfaces, con el uso de patrones y framework se facilita el desarrollo de la aplicación, ya que imponen un orden y una estructura. Se utilizó la metodología XP, esta metodología centra las prioridades en las personas, y no en los procesos esta metodología se implementó en el desarrollo debido a que es una metodología de desarrollo común, sencillo y adaptable a las características cambiantes existentes en las empresas.

En cuanto a las recomendaciones, se sugiere a futuro dentro del módulo de Declaración de Siniestros se agregue la funcionalidad donde el cliente asegurado pueda descargar la orden de reparación, crear la opción de Notificación de Pérdida Total pero esto si se tomaría más tiempo de aprobación debido que es más riesgoso para la Empresa de Seguros por la leyes de seguros. También se sugiere que se cree la funcionalidad de que las pólizas colectivas puedan ser declaradas por la aplicación Web, debido a que no está en el alcance de este Trabajo Especial de Grado.

Referencias Bibliográficas

[1] Luján Mora, Sergio. (2009). Programación de Aplicaciones Web: Historia, Principios Básicos y Clientes Web. Consultado el día 13 de Noviembre de 2010 de la World Wide Web: http://gplsi.dlsi.ua.es/~slujan/materiales/pi-cliente2-muestra.pdf

[2] Moreno, Manuel. (2010). Aplicación Web. Consultado el día 13 de Noviembre de 2010 de la World Wide Web: http://www.alegsa.com.ar/Dic/aplicacion%20web.php

[3] Vegas, Jesús. (2002). Desarrollo de aplicaciones Web. Consultado el día 13 de Noviembre de 2010 de la World Wide Web:

http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node17.html

[4] Mateu, Carles. (2004). Desarrollo de aplicaciones Web. Software Libre. Pág. 378.

[5] Instituto Tecnológico de Colima. (2006). Arquitectura Cliente – Servidor. Consultado el día 14 de Noviembre de 2010 de la World Wide Web:

http://www.itcolima.edu.mx/profesores/tutoriales/fundamentosbd/sd_u1_6.htm

[6] Márquez, Catarina. (2008). Trabajo de Investigación: Cliente – Servidor. Consultado el día 14 de Noviembre de 2010 de la World Wide Web:

http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/capitulo5.pdf

[7] Sánchez González, Carlos. (2004). Trabajo de Investigación: Aplicaciones en Capas. Consultado el día 22 de Noviembre de 2010 de la World Wide Web:

http://oness.sourceforge.net/proyecto/html/ch03s02.html

[8] Santa Cruz, de la Sierra. (2004). Diseño de Aplicaciones en tres Capas. Consultado el día 25 de Noviembre de 2010 de la World Wide Web: http://www.geocities.com/trescapas

[9] Webtutoriales. (2008). Tutorial de Modelo – Vista – Controlador. Consultado el día 25 de Noviembre de 2010 de la World Wide Web:

http://www.webtutoriales.com/tutoriales/programacion/modelo-vista-controlador.54.html

[10] Pavón Maestras Juan, Estructuras de las Aplicaciones Orientadas a Objetos Modelo – Vista- Controlador. (2010). Modelo – Vista – Controlador. Consultado el día 25 de Noviembre de 2010 de la World Wide Web: http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.MVC.pdf

[11] Pantoja, Ernesto (2010). El patrón de diseño Modelo – Vista – Controlador (MVC). Consultado el día 02 de Diciembre de 2010 de la World Wide Web:

http://www.scribd.com/doc/13673155/Modelo-vista-controlador-con-java-swing

[12] Bad Software Applications, S.L. (2009). Diseño y Desarrollo de Aplicación Web. Consultado el día 02 de Diciembre de 2010 de la World Wide Web: http://www.bab-soft.com/es/diseno_desarrollo_aplicaciones_web.php

[13] Cuesta Morales, Pedro. (2004). Desarrollo de Aplicaciones Distribuidas basadas en Tecnología Web. Consultado el día 08 de Diciembre de 2010 de la World Wide Web:

http://trevinca.ei.uvigo.es/~pcuesta/publicaciones/TecWeb.pdf

[14] Guiarte Multimedia S.L. DesarrolloWeb.com. (2008). Manual de HTML. Consultado el 15 de Diciembre de 2010 de la World Wide Web:

http://www.desarrolloweb.com/manuales/21

[15] Pérez, Javier. (2008). Introducción a CSS. Consultado el 20 de Diciembre de 2010 de la World Wide Web:

http://www.librosweb.es/css/pdf/introduccion_css.pdf

[16] Pérez, Javier. (2009). Introducción a JavaScript. Consultado el 20 de Diciembre de 2010 de la World Wide Web:

http://www.librosweb.es/javascript/pdf/introduccion_javascript.pdf

[17] Álvarez, Miguel. (2009). Manual de JavaScript. Consultado el 26 de Diciembre de 2010 de la World Wide Web:

http://www.manualdejavascript.com/manualjavascript/archivos-codigo-javascript.html

[18] Pérez, Javier. (2008). Introducción a Ajax. Consultado el 26 de Diciembre de 2010. Manual de Ajax.

[19] Pérez Javier Eguilez. (2009). Curso de Ajax. Consultado el 26 de Diciembre de 2010 de la World Wide Web: http://www.ajaxya.com.ar/.

[20] Guiarte Multimedia S.L. DesarrolloWeb.com. (2008). Manual de JQuery. Consultado el 03 de Enero de 2011 de la World Wide Web:

http://www.scribd.com/doc/43914919/Manual-de-Jquery-en-PDF-Desarrollowebcom.

[21] Murphey Rebecca, (2009). Fundamentos de jQuery. Consultado el 03 de Enero de 2011 de la World Wide Web:

http://www.javascriptya.com.ar/jquery/index.php?inicio=0

[22] DesarrolloWeb. (2010). Script del lado del Servidor. Consultado el 03 de Enero de 2011 de la World Wide Web:

http://www.desarrolloweb.com/descargas/descargar.php?descarga=7760

[23] Aulbach, Alexander; Egon, Schmid; Jim, Winstead; Rasmun, Lerdorf; Zeev, Suraski; Andrei, Zmievski; Jouni , Ahto y Stig, Bakken. (2001). Manuel de PHP. Pág. 1063.

[24] Martínez, Rafael. (2002). Manual de PHP. Consultado el 03 de Enero de 2011 de la World Wide Web:

http://www.educarm.es/templates/portal/ficheros/websDinamicas/98/php_manual.pdf

[25] Van Der Henst. (2010). Servidor HTTP Apache. Consultado el 03 de Enero de 2011 de la World Wide Web:

http://www.maestrosdelweb.com/editorial/phpmysqlap/

[26] Ciberaula.com. (2010). Una introducción a Apache. Consultado el 06 de Enero de 2011 de la World Wide Web:

http://linux.ciberaula.com/articulo/linux_apache_intro/

[27] Oracle. (2010). Manual de MySQL. . Consultado el 06 de Enero de 2011 de la World Wide Web:

http://dev.mysql.com/doc/refman/5.0/es/introduction.html

[28] Oracle. (2010). Manual de Referencia MySQL. Consultado el 06 de Enero de 2011 de la World Wide Web:

http://www.google.com/url?sa=t&source=web&cd=2&sqi=2&ved=0CCIQFjAB&url=http%3A%2F%2Fdownloads.mysql.com%2Fdocs%2Frefman-5.0-es.a4.pdf&rct=j&q=manuales%20de%20MySQl&ei=jsQoTfCJDYH7lwfs1pzcAQ&usg=AFQjCNEPfMsx7J64BMUfGnQlUq-0OsdYpw&sig2=sLW-qAwrePG5hR7A-hC1qA&cad=rja

[30] Sitio Oficial de SUN. (2002). Consultado el día 06 de Enero de 2010 de la World Wide Web: http://java.sun.com/blueprints/patterns/TransferObject.html.

[31] Librosweb.es. (2009). Frameworks. Consultado el día 01 de Abril de 2011 de la World Wide Web: www.librosweb.es/css_avanzado/capitulo5.html

[32] AE Tecnología y Desarrollo.com. (2009). Introducción al Framework CakePHP. Consultado el día 01 de Abril de 2011 de la World Wide Web:

www.arrayexception.com/desarrollo/how-to/introduccion-al-framework-cakephp/

[33] Fernández Escribano, Gerardo. (2002) Introducción a Extreme Programming. Consultado el día 06 de Enero de 2010 de la World Wide Web:

http://java.sun.com/blueprints/patterns/TransferObject.html.

[34] Calero Solís, Manuel. (2003).Programación Extrema XP. Consultado el día 06 de Enero de 2010 de la World Wide Web:

http://www.willydev.net/descargas/prev/explicaxp.pdf

[35] Beck Kent. (2010).Una Explicación de la Programación Extrema.

[36] Portal de Seguros Carabobo. Glosario (2010). Definiciones de Seguros. Consultado el día 01 de Abril de 2011 de la World Wide Web:

http://www.seguroscarabobo.com/portal/glosario.jsp