Manizales, 27 de Marzo de 2009

Señores
COMITÉ DE TRABAJOS DE INVESTIGACIÓN
Facultad de Ingeniería
Universidad de Manizales
Manizales

Acorde a la resolución 006 de 2006 emanada por el Consejo de la facultad de Ingeniería, presentamos el informe final del trabajo de grado (tres discos) titulado **Sistema de información integrada para corredores y agencias de seguros orientado a web**, realizado como requisito parcial para optar el título de Ingeniero de Sistemas y Telecomunicaciones, con el fin de ser sometido a la evaluación por los jurados que ustedes se sirvan designar.

El proyecto está incluido en el Campo de Conocimiento de **Electrónica y telecomunicaciones**, específicamente en el Área de Conocimiento denominado: **Desarrollo Tecnológico**.

Las personas que somos responsables del proyecto, ya que hemos estado involucradas directamente en el planteamiento y desarrollo del proyecto, aparecemos en seguida:

Rol	Nombre completo	Código	Teléfono	Correo electrónico
Autor(es)	**Gerónimo Barreneche Renaud**	**82200213529**	**2607193**	geronimo.barreneche@cormacarena.com.co
Presidente	**Jaime Ariel Rios**			jaro@umanizales.edu.co

A continuación indicamos las partes que se presentan en el informe final de nuestro proyecto:

X	Portada
X	Contraportada
X	Agradecimientos (opcional)
X	Página de aceptación
X	Contenido
X	Listas especiales (lista de figuras, lista de tablas, lista de anexos)
X	Glosario

X	Resumen / Abstract
X	Resumen analítico
X	Introducción
X	1. Área problemática
X	2. Objetivos (general y específicos)
X	3. Justificación (novedad, interés, utilidad)
X	4. Marco teórico
X	5. Metodología (Tipo de trabajo, procedimiento)
X	6. Resultados (Descripción y discusión)
X	7. Conclusiones
X	8. Recomendaciones
X	Bibliografía
X	Anexos (diagramas de análisis y diseño, manuales, etc.)
	Otro. <definir>

Asimismo se adjuntan:

X	Artículo científico para la Revista Ventana Informática (no es copia del informe final sino un artículo construido a partir de él), titulado <>
X	Propuesta aprobada, - Aplicación completa.
X	Carta de aceptación del informe por parte de la empresa (en caso de haberse realizado en una empresa específica)
	Otro. <definir>

Atentamente,

_____	_____
Jaime Ariel Ríos O.	**Gerónimo Barreneche Renaud**
Presidente	**Estudiante**

SISTEMA DE INFORMACIÓN INTEGRADA PARA CORREDORES Y AGENCIAS DE SEGUROS ORIENTADO A WEB

GERÓNIMO BARRENECHE RENAUD

UNIVERSIDAD DE MANIZALES
FACULTAD DE INGENIERÍA
PROGRAMA SISTEMAS Y TELECOMUNICACIONES
MANIZALES
2009

SISTEMA DE INFORMACIÓN INTEGRADA PARA CORREDORES Y AGENCIAS DE SEGUROS ORIENTADO A WEB

GERÓNIMO BARRENECHE RENAUD

Trabajo de Grado presentado como opción parcial para optar
al título de **Ingeniero de Sistemas y Telecomunicaciones**

Presidente
Jaime Ariel Ríos O.
Ingeniero de Sistemas
Docente Universidad de Manizales
Facultad de Ingeniería

UNIVERSIDAD DE MANIZALES
FACULTAD DE INGENIERÍA
PROGRAMA SISTEMAS Y TELECOMUNICACIONES
MANIZALES
2009

PÁGINA DE ACEPTACIÓN

_____
_____
_____
_____
_____
_____
_____

_____
<NOMBRE COMPLETO>
JURADO

_____
<NOMBRE COMPLETO>
JURADO

_____
<NOMBRE COMPLETO>
JURADO

Manizales, <día> de <mes> de <año>

CONTENIDO

LISTA DE FIGURAS

LISTA DE ANEXOS

GLOSARIO

APACHE: Servidor web, diseñado principalmente bajo el sistema operativo Linux, posee compatibilidad con otros sistemas operativos.

ASP: Active Server Page. Es una tecnología propietaria de Microsoft. Se trata básicamente de un lenguaje de tratamiento de textos (scripts), basado en Basic, y que se denomina VBScript (Visual Basic Script).

CGI: Common Gateway Interface. CGI es una norma para establecer comunicación entre un servidor web y un programa, de tal modo que este último pueda interactuar con Internet. También se usa la palabra CGI para referirse al programa mismo, aunque lo correcto debería ser script.

CHILISOFT: Es una compañía que ofrece la tecnología ASP a través de chilisoft.asp, es el mismo funcionamiento que ASP de Microsoft, pero corre en sistemas operativos como UNIX, LINUX, SOLARIS, entre otros.

HTML: Hypertext Markup Language (Lenguaje de Marcado por Hipertexto). HTML es el lenguaje con el que se definen las páginas Web. Básicamente se trata de un conjunto de etiquetas que sirven para definir la forma en la que presentar el texto y otros elementos de la página.

HTTP: Es la abreviatura de Hypertext Transfer Protocol (Protocolo de Transferencia de Hipertexto). Es un conjunto de reglas, o protocolo, que gobierna la transferencia de hipertexto entre dos o más computadoras. La World Wide Web agrupa el universo de información que está disponible vía HTTP.

IIS: Internet Information Server. Servidor Web Microsoft para los sistemas operativos Windows NT, Windows Workstation, Windows 2000 server, Windows XP.

INSTANT ASP: Es un software que actúa como intérprete para el funcionamiento de ASP en plataforma LINUX/UNIX.

JSP: Java Server Page. Es una extensión de la tecnología Java Servlets. Creado por Sun Microsystems

PHP: Hypertext Processor. Es un lenguaje interpretado de alto nivel embebido en páginas HTML.

PWS: Personal Web Server. Es un producto de Microsoft que viene junto con algunos productos de la casa como Windows 98 o Frontpage 98. Se trata de un servidor de páginas web personal que puede ser utilizado por un máximo de hasta 10 usuarios de forma concurrente. Puede utilizarse para probar el funcionamiento de nuestra Web sin necesidad de publicarla en Internet o para formar una pequeña intranet corporativa.

TOMCAT: Servidor Web de Apache Group, para el manejo de páginas JSP.

SERVIDOR WEB: Sitio en el que se alojan las páginas Web.

SCRIPT: Conjunto de instrucciones que se ejecutan paso a paso, instrucción a instrucción.

JAVA: Es un lenguaje de programación, de alto nivel y orientado a objetos utilizado principalmente para programar en Internet o intranets.

LINUX: Es un sistema operativo de libre distribución basado en UNIX.

VBSCRIPT: Es un lenguaje de scripting utilizado para agregar funcionalidad a las páginas Webs, el cual fue adaptado para poder generar archivos ejecutables que faciliten ciertas funciones del sistema operativo, si el usuario lo desea. El mismo está basado en el lenguaje de programación Visual Basic, y es desarrollado por Microsoft.

CLIENTE/SERVIDOR: Es el procesamiento cooperativo de la información por medio de un conjunto de procesadores, en el cual múltiples clientes, distribuidos geográficamente, solicitan requerimientos a uno o más servidores centrales.

RESUMEN

El documento corresponde a elementos evidénciales de la creación de un software para empresas aseguradoras en seguros. Dicho sistema esta implementado e implantado bajo tecnologías Web, permitiendo una comunicación efectiva con sus clientes y actores administrativos.

En la región del eje cafetero, existe una empresa llamada Gilberto Arenas y CIA LTDA, la cual es una agencia asesora de seguros que tiene como domicilio principal la ciudad de Pereira, teniendo. Entre sus funciones principales se tiene la venta de seguros y el asesoramiento en temas relacionados con su función. Lo que cubre seguros personales como seguros de vida, de salud, vivienda y vehiculares y seguros corporativos los cuales hacen referencia a las pólizas para las instituciones o llamadas comúnmente seguros generales.

La problemática está centrada en los procesos de venta de servicios, los cuales se coordinan vía telefónica con los diferentes agentes ubicados estratégicamente en la región, éstos dependen de las oficinas principales para la emisión de información para el cliente al momento de la negociación, lo que ocaciona inconvenientes de orden procedimental, generando problemas para el flujo ágil de la información.

El proyecto, plantea una solución de software de orden distribuido, es decir, el diseño e implementación de un producto que funciona bajo características Web, permitiendo mejorar la comunicación con los clientes.

PALABRAS CLAVES: Software, Web, Seguros

ABSTRACT

The document is evidentiality elements of the creation of software for insurance companies in insurance. This system is implemented and enforced on Web technologies, allowing effective communication with clients and administrative actors.

In the coffee-growing region, a company called Gilbert Arenas and CIA LTDA, which is an agency that advises insurance is the principal city of Pereira, having. Among its main functions is the sale of insurance and advice on issues related to their function., Which covers personal insurance such as life insurance, health, housing and vehicle insurance business and which relate to policies for the institutions .

The problem centers on the process of selling services, which are coordinated via telephone with the various agents strategically located in the region, they rely on the main offices for issuing information to the customer at the time of negotiation, which occasion of procedural shortcomings, creating problems for the smooth flow of information.

The project raises a software solution of a distributed, the design and implementation of a product that works under Web features, allowing better communication with customers.

KEY WORDS: Software, Web, Insurance

RESUMEN ANALÍTICO

Título del Proyecto	Sistema de información integrada para corredores y agencias de seguros orientado a web
Autor(es)	Gerónimo Barreneche Renaud geronimo.barreneche@cormacarena.com.co
Presidente	Ríos Jaime Ariel jaro@umanizales.edu.co Ingeniero de Sistemas
Tipo de documento	Trabajo de grado
Institución	Ingeniería de Sistemas y Telecomunicaciones Facultad de Ingeniería Universidad de Manizales
Palabras claves	Software, Web, Seguros
Descripción	El documento corresponde a elementos evidénciales de la creación de un software para empresas aseguradoras en seguros. Dicho sistema esta implementado e implantado bajo tecnologías Web, permitiendo una comunicación efectiva con sus clientes y actores administrativos.
Fuentes	Jaime Ariel Ríos Gilberto Arenas y CIA LTDA
Contenido	Área problemática Objetivos Objetivo general Objetivos específicos Justificación Marco teórico Antecedentes Metodología Resultados Conclusiones Recomendaciones Bibliografía Anexos
Metodología	En términos generales, la metodología está fundamentada por fases, las cuales, permiten asegurar

	la calidad, desde la recolección de datos, hasta la creación de un producto de software.
Conclusiones	Haciendo síntesis sobre los resultados más importantes y significativos, se concluye el proyecto, el cual, logró la construcción de un sistema bajo plataformas Web, permitiendo cumplir con los objetivos planteados en la Empresa. A continuación se enmarcan en contexto general, los puntos sobre los cuales se tiene consideración más relevante.
Anexos	Anexo A (Requisitos) Anexo B (Análisis) Anexo C (Diseño) Anexo D (Base de Datos) Anexo F (Definición Técnica)

INTRODUCCIÓN

Gilberto Arenas y CIA LTDA es una agencia asesora de seguros que tiene como domicilio principal la ciudad de Pereira y tiene cobertura completa en el eje cafetero (Manizales – Pereira – Armenia). Su función principales el asesoramiento y venta de todo tipo de seguros, lo que cubre seguros personales como seguros de vida, de salud, vivienda y vehiculares y seguros corporativos los cuales hacen referencia a las pólizas para las instituciones.

En la actualidad los procesos de venta de servicios se coordinan vía telefónica con los diferentes agentes ubicados estratégicamente en la región y los cuales dependen de las oficinas principales para la emisión de información exacta para el cliente al momento de la negociación y además presentando falencias en el control estadísticos de los clientes en el nivel central ya que la información puede ser inexacta.

El proyecto plantea un software orientado a la WEB que permita realizar cotizaciones de los diferentes productos en línea, permitiendo de ésta forma fortalecer las comunicaciones y logrando un mejor control sobre todas las productos cotizados, no facturados y facturados, así como el seguimiento de la información y estado de cada cliente.

Para fortalecer la comunicación entre los diferentes agentes se plantea la construcción de un sistema de comunicación interno, dónde se enviarán y recibirán comunicaciones referentes a los hechos comerciales, permitiendo hacer auditoría sobre los mensajes, tanto los leídos, como los reenviados. Lo anterior permite conocer como avanza la gestión con los clientes por parte de la agencia, y así poder aplicar correctivos cuando sea necesario.

El proyecto plantea un gestor de clientes integral CRM, el cual permite un manejo global de la información del cliente como persona natural o jurídica, dicha información será aprovechada por el departamento de ventas para buscar los diferentes negocios en el mercado.

El manejo integral de la tesorería y la cartera estarán totalmente integrados con todos los módulos del desarrollo propuesto, permitiendo calcular y registrar comisiones en las ventas, cuentas por cobrar, entre otros.

Para el control de procesos y procedimientos, se plantea la construcción de un módulo que permita hacer auditoria sobre la gestión comercial, esto se denomina una hoja de ruta.

1. ÁREA PROBLEMÁTICA

En el momento la empresa Gilberto Arenas y CIA LTDA, cuenta con un sistema de contabilidad, el cual es utilizado únicamente para llevar el control financiero.

Se hace necesario contar con un sistema de información que controle los aspectos, desde los clientes, agentes, compañías, cartera, partes de siniestros, pólizas, seguimiento de correspondencia, pagos electrónicos, recibos, proveedores y procesos de comunicación interna. Por consiguiente se requiere crear una aplicación que tenga interfaces amigables con el usuario y tenga un manejo intuitivo pero que cubra los requerimientos que la empresa necesita para el control de todos sus procesos.

Se hace necesario el seguimiento y control de clientes, dónde brinde la posibilidad de controlar a aquellos clientes con los que hay que contactar un día o mes determinado, por que se haya dejado alguna gestión pendiente, vencimiento de alguna póliza que tiene contratada con otra compañía, el vencimiento de una factura financiada o simplemente con el fin de felicitarle el día de su cumpleaños. En el trabajo de los asesores de seguros, es importante mantener una relación de amistad y confianza con sus asegurados.

Para la aseguradora Gilberto Arenas y CIA LTDA se hace imprescindible un sistema que gestione información general, cotizaciones on-line, gestión de Solicitudes, gestión de comunicaciones con los clientes, gestión de los siniestros, información on-line a los clientes, gestión de seguimientos, gestión de pagos, gestión de cotizaciones, gestión de pagos, gestión de cartera; por otro lado es de vital relevancia que contemple agentes y subagentes para el manejo automático de comisiones.

Otro aspecto fundamental es un sistema de mensajería interno, que permita comunicar a todos los asesores de seguros entre ellos y con los clientes en forma controlada, dicho sistema debe ser contemplado en un sistema WEB, el cual deberá permitir la validación de usuarios al momento de si ingreso y asignándole los roles adecuados a su perfil, permitiendo mejorar la calidad en la comunicación al interior de la empresa.

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Analizar, diseñar y desarrollar un sistema WEB para la gestión de la información en la seguradora Gilberto Arenas y CIA LTDA.

2.2 OBJETIVOS ESPECÍFICOS

- Analizar, diseñar e implementar un módulo para la gestión del conocimiento del cliente (CRM).
- Analizar, diseñar e implementar un módulo para el control de tesorería y cartera.
- Analizar, diseñar e implementar un módulo para la hoja de ruta (control de negocios).
- Analizar, diseñar e implementar un módulo para cotizaciones en línea.
- Analizar, diseñar e implementar un módulo para pagos en línea dependiendo del perfil.
- Analizar, diseñar e implementar un módulo para gestión de la mensajería interna.
- Analizar, diseñar e implementar un módulo para emitir informes requeridos previamente por la empresa.
- Analizar, diseñar e implementar un módulo administrativo para perfiles y roles del sistema.
- Instalar un sitio web para las consultas en línea.
- Realizar interfaz con el software contable.
- Desarrollar toda la documentación de manuales para el manejo de usuarios tanto especializados como finales.

3. JUSTIFICACIÓN

Se hace necesario un sistema de información orientado a trabajar en línea, el cual permita la comunicación entre las agencias distribuidas en el eje cafetero, para disminuir los costos en las comunicaciones y mejorar el control y agilidad de las mismas. Dicho sistema debe ser un gestor de mensajería que permita llevar un control de tiempos a los documentos y la información de clientes y financiera.

La gestión de cotizaciones en línea mediante un software a la medida permite controlar a los agentes, logrando medir su gestión con base a la comparación entre cotizaciones y ventas reales.

Para la aseguradora Gilberto Arenas y CIA LTDA es novedoso gestionar el conocimiento del cliente, factor clave de éxito para los vendedores de seguros, puesto que al tener buena información se logra un mejor servicio.

La definición de rutas en procesos de cotización y negocios en forma sistemática, aumenta la calidad del servicio que se presta actualmente, permitiendo ejercer mayor control.

La construcción de un software integral que permita la gestión de clientes, tesorería, cotizaciones, ventas, control de rutas y gestión de documentos, hace de la empresa una organización más competitiva en el mercado regional.

El proyecto plantea un sistema novedoso de mensajería interna, el cual consiste en la creación de un documento en el sistema (digital), luego éste lo envía a otro usuario interesado y así sucesivamente hasta llegar a la culminación del proceso de comunicación pudiendo cambiar de estados pendiente, exitoso, caducado. Lo anterior se denomina ruta, la cual puede ser consultada por un auditor o un usuario con nivel alto de prioridad (por ejemplo: un derecho de petición en el transcurso de todo su proceso).

La novedad y la solución están enmarcadas para la empresa en un ámbito global, es decir, teniendo en cuenta las necesidades de desarrollo de un software a la medida que presenta y tomando la situación actual de los sistemas de información, es importante anotar que la aplicación de los objetivos son de gran relevancia para la empresa ya que tiene una serie de requerimientos específicos que no han sido encontrados en software ya existentes.

4. MARCO TEÓRICO

4.1Gilberto Arenas y CIA LTDA

Gilerto Arenas y CIA LTDA es una empresa dedicada al sector de los seguros, su funcionamiento básico radica en la operación de las diferentes aseguradoras.

MISIÓN:

Basado en principios éticos y morales de tradición y seriedad, prestará el mejor servicio a cada uno de nuestros asegurados y respectivos beneficiarios. Contando con su excelente talento humano, buena y moderna infraestructura, comprometidos para satisfacer expectativas y dudas en el momento que alguno de ellos lo requieran.

VISIÓN:

Ser líder a mediano plazo en la prestación de servicios, buscando los netos entre la Compañía de Seguros y las Entidades que tienen sus programas contratados con nuestra Agencia de Seguros.

POLÍTICA DE CALIDAD:

Fundamentalmente la satisfacción de todos y cada uno de nuestros clientes a través del mejoramiento continúo de los servicios en forma eficaz y oportuna, contribuyendo a mejorar su nivel de vida.

VALORES:

Respeto, honestidad, compromiso, amor y cooperación.

4.2 UML

Un modelo es una descripción de (parte de) un sistema, descrito en un lenguaje bien definido. Un lenguaje bien definido es un lenguaje con una sintaxis y semántica precisa, y que puede ser interpretado y manipulado por un computador [1].

Dicha herramienta, es el factor clave para el desarrollo de sistemas de información, puesto que permite establecer mediante elementos gráficos, todo el contexto de la aplicación Web a desarrollar.

El lenguaje de modelado UML es el estándar más utilizado para especificar y documentar cualquier sistema de forma precisa. Sin embargo, el hecho de que

dicha herramienta sea una notación de propósito muy general obliga a que muchas veces sea deseable poder contar con algún lenguaje más específico para modelar y representar los conceptos de ciertos dominios particulares. Los Perfiles UML constituyen el mecanismo que proporciona para extender su sintaxis y su semántica para expresar los conceptos específicos de un determinado dominio de aplicación.

Para ello el UML, (Unified Modeling Language), está compuesto por una gama de diagramas o artefactos, que permiten graficar o tomar una radiografía a los procesos para una interpretación de los mismos desde el punto de vista de usuario como de los desarrolladores de Software, el UML es como que si fuese el Castellano que utiliza un abecedario compuesto de letras, las cuales formaran silabas, palabras, oraciones, párrafos y documentos que contendrán un pensamiento, mediante el castellano se escribe una novela, una canción una poesía entres otros, Algo análogo a UML, es el hecho que un lenguaje el cual usa sus diagramas como que si fuera el abecedario del idioma castellano, y estos servirán para plasmar los procesos de un determinado negocio, para de esta manera construir modelos o maquetas de la interpretación de los mismos entre usuarios y desarrolladores de los sistemas de información[2]. Existiendo para ello un lenguaje común de comunicación, Cuando no se usa la notación UML(conjuntos de Diagramas) se tiene la problemática de no encontrar los términos adecuados, ya que los analistas usan términos informáticos de difícil entendimiento por los usuarios, y por otra parte los desarrolladores no entienden el lenguaje que usan los usuarios del mundo real de estudio[3].

Para la contextualización teórica de los temas referentes a UML, es necesario hacer hincapié sobre los siguientes temas [4]:

Clase
Los objetos que tengan los mismos atributos y comportamiento se agrupan en clases. Todos los alumnos tienen una serie de atributos comunes: nombre, apellido Paterno, apellido materno, fecha de nacimiento y un comportamiento común: podemos hacer referencia a un alumno para matricularlo o retirarlo. Los valores de los atributos podrán ser distintos para cada una de ellos, pero todos comparten los mismos atributos y comportamiento (las operaciones que se pueden realizar sobre ellos).

Abstracción
Se Refiere a quitar las propiedades y acciones de un objeto para dejar sólo aquellas que sean necesarias.

Herencia
El concepto de herencia se refiere a la compartición de atributos y operaciones basada en una relación jerárquica entre varias clases. Una clase puede definirse de forma general y luego refinarse en sucesivas subclases. Cada clase hereda

todas las propiedades (atributos y operaciones) de su superclase y añade sus propiedades particulares.

Polimorfismo
El polimorfismo permite que una misma operación pueda llevarse a cabo de forma diferente en clases diferentes. Por ejemplo, la operación borrar, es distinta para una mota, un corrector y un borrador, pero ambos objetos pueden servir para borrar. Una operación es una acción o transformación que realiza o padece un objeto. La implementación específica de una operación determinada en una clase determinada se denomina método.

Encapsulamiento
La esencia del encapsulamiento(o encapsulación), es cuando un objeto trae consigo funcionalidad, esta última se oculta.

Envío de Mensajes
Un sistema de Objetos Trabaja en conjunto. Esto se logra mediante el envío de mensajes entre ellos. Un objeto envía a otro un mensaje para realizar una operación, y el objeto receptor ejecutará la operación.

Asociaciones
Los Objetos se relacionan entre sí, de alguna forma. Una Clase puede Asociarse con mas de una clase distinta. La multiplicidad en un importante aspecto de las asociaciones, Indica la cantidad de objetos de una clase que se relacionan con otro objeto en particular de la clase asociada.

Agregación
Es cuando los objetos se integran pero conservan su independencia. Una Pc es un ejemplo de composición ya que sus objetos como el mouse, los parlantes, el teclado, son objetos que pueden sacarse de una computadora a otra.

Composición
El concepto de composición es similar al de la agregación, pero sus objetos que lo integran no tendrán su independencia, por ejemplo si analizamos una camisa, el objeto compuesto seria un bolsillo de la camisa que no podría integrarse a otra por la diferencia de color, tamaño entre otros.

4.3 Base de Datos

El concepto de base de datos manejado en éste apartado está orientado a su definición universal, se explicarán algunos términos para la aplicación del proyecto en la sección correspondiente a Postgres.

Básicamente es definida como una colección de archivos interrelacionados, son creados con un DBMS*. El contenido de una base de datos engloba a la información concerniente (almacenadas en archivos) de una organización, de tal manera que los datos estén disponibles para los usuarios, una finalidad de la base de datos es eliminar la redundancia o al menos minimizarla. Los tres componentes principales de un sistema de base de datos son el hardware, el *software* DBMS y los datos a manejar, así como el personal encargado del manejo del sistema.

Los sistemas de base de datos se diseñan para manejar grandes cantidades de información, la manipulación de los datos involucra tanto la definición de estructuras para el almacenamiento de la información como la provisión de mecanismos para la manipulación de la información, además un sistema de base de datos debe de tener implementados mecanismos de seguridad que garanticen la integridad de la información, a pesar de caídas del sistema o intentos de accesos no autorizados[1].

En las bases de datos existen conceptos para hacer relación a su estructura interna, por ejemplo entidad, entidades y conjunto de entidades, concepto de relación, conjunto de relaciones, tipos de relaciones, limitantes de mapeo, qué es una llave primaria, diagrama Entidad Relación, reducción de diagramas E-R a tablas. Agregación. Estos términos se pueden ampliar ingresando a la página que se hace referencia al pie de página[2].

También se entiende el modelo relacional como estructura de los datos, integridad de los datos, manipulación de los datos. Tipos de relaciones, las cuales se basan en claves primarias y dependencias funcionales, y para lograr su correcto funcionamiento es necesario plantear una correcta normalización[3].

4.4 JDBC

JDBC*, es conocido como el puente entre Java y las bases de datos, su función primordial es lograr la conexión mediante DRIVERS** y lograr el desarrollo de

---

* Comúnmente conocido como el motor de bases de datos, el cual es el programa o conjunto de programas que se usan para almacenar y acceder a la información en forma rápida y eficaz.

1 BASES DE DATOS. Introducción a los Conceptos de Bases de Datos. [En Línea]. Fecha de Consulta: 10.11.2007. Disponible en : http://academicos.cualtos.udg.mx/Informatica/

2 Ibid.

3 Proceso de formas normales para llegar de un conjunto de datos a la estructura deseada, son cinco formas normales pero en su aplicación se puede llegar a la cuarta sin generar problemas de diseño.

* Java Data Base Conectivy

** Archivo que representa la conectividad con la base de datos

software útil y con almacenes de datos grandes. JDBC es usado en el proyecto para lograr la conexión con POSTGRES[***].

4.5 Postgres

Los sistemas de mantenimiento de Bases de Datos relacionales tradicionales (DBMS,s) soportan un modelo de datos que consisten en una colección de relaciones con nombre, que contienen atributos de un tipo específico[5]. En los sistemas comerciales actuales, los tipos posibles incluyen numéricos de punto flotante, enteros, cadenas de caracteres, cantidades monetarias y fechas. Está generalmente reconocido que este modelo será inadecuado para las aplicaciones futuras de procesado de datos[6]. El modelo relacional sustituyó modelos previos en parte por su "simplicidad espartana". Sin embargo, como se ha mencionado, esta simplicidad también hace muy difícil la implementación de ciertas aplicaciones. Postgres ofrece una potencia adicional sustancial al incorporar los siguientes cuatro conceptos adicionales básicos en una vía en la que los usuarios pueden extender fácilmente el sistema [7]:

- Clases
- Herencia
- Tipos
- Funciones

PostgreSQL es un gestor de bases de datos orientadas a objetos (SGBDOO o ORDBMS en sus siglas en inglés) muy conocido y usado en entornos de software libre porque cumple los estándares SQL92 y SQL99, y también por el conjunto de funcionalidades avanzadas que soporta, lo que lo sitúa al mismo o a un mejor nivel que muchos SGBD comerciales. El origen de PostgreSQL se sitúa en el gestor de bases de datos POSTGRES desarrollado en la Universidad de Berkeley y que se abandonó en favor de PostgreSQL a partir de 1994. Ya entonces, contaba con prestaciones que lo hacían único en el mercado y que otros gestores de bases de datos comerciales han ido añadiendo durante este tiempo[8].

PostgreSQL puede funcionar en múltiples plataformas (en general, en todas las modernas basadas en Unix) y, a partir de la versión 8.0 (actualmente en 8.3), también en Windows de forma nativa. Para las versiones anteriores existen versiones binarias para este sistema operativo, pero no tienen respaldo oficial.

4.6 Java

[***] Motor de bases de datos seleccionado para el desarrollo del proyecto

Java 2 Plataform, Standard Edition es la tecnología básica para muchos estilos diferentes de desarrollo de software, incluidos applets, aplicaciones clientes y aplicaciones servidores individuales. Java es, desde su diseño, un lenguaje capaz de superar las diferencias que hay entre los computadores de una red heterogénea, al ser independiente del sistema operativo[9].

La última versión de J2SE incorpora nuevas funciones, un mayor rendimiento y mejoras de escalabilidad que representan un paso adelante para la tecnología Java. Con la versión 6, se puede utilizar la tecnología Java para desarrollar aplicaciones más complicadas con menos esfuerzo y en menos tiempo[10]. Permite aplicaciones escalables y de gran rendimiento para su implantación en cualquier plataforma.

Java es un potente y versátil lenguaje de programación que puede trabajar en todo tipo de entornos, desde servidores de aplicaciones middletier hasta clientes Web[11]. Independientemente del tipo de aplicación que usted desarrolle y del tipo de máquina en la que se ejecute el código, su aplicación seguramente tendrá que acceder a datos almacenados en algún tipo de base de datos. Las bases de datos relacionales son la elección obvia en la mayor parte de las empresas, y han evolucionado espectacularmente en los últimos años hasta convertirse en potentes sistemas de gestión de bases de datos.

4.7 JSP

El servidor de paginas bajo Java (JSP), representa una tecnología cada vez más popular para construir aplicaciones web dinámicas, las cuales pueden acceder a bases de datos y proporcionar una experiencia interactiva a los usuarios de su sitio web. JSP se construye sobre el lenguaje de programación Java[19].

Actualmente se utilizan de forma dominante las aplicaciones Web en contraposición a los sitios Web estáticos. En términos generales, es necesario la construcción de Web dinámicos, es decir, presentar la información en línea, para permitir que los usuarios entren en el sistema y personalicen la apariencia del sitio, o para ofrecer la posibilidad de adquirir sus servicios. JSP ofrece una estructura que permite obtener información del usuario y crear páginas Web rápidamente, cómo los componentes JavaBeans y las bibliotecas de comandos le permiten hacer su código más legible y fácil de mantener y, por supuesto, cómo funciona el propio lenguaje Java. Este libro también explica cómo tratar los errores en el código, los mejores métodos para el diseño de aplicaciones web y culmina con estudio global de un caso: un sitio web para una organismo de turismo local.

4.8 Análisis comparativo de las herramientas de programación Web: PHP, ASP y JSP, bajo los sistemas operativos Linux y Windows[*]

La programación en la Web ha generado como consecuencia la creación de varias herramientas de desarrollo, por lo que es importante identificar cuáles ofrecen un mejor rendimiento y bajo qué Sistema Operativo. Las herramientas de programación Web analizadas fueron: PHP, ASP y JSP, bajo los sistemas operativos Linux y Windows utilizando criterios comunes. Se llevó a cabo un estudio descriptivo - deductivo; se desarrolló un prototipo en el que se muestra el funcionamiento de las herramientas mencionadas, con bases de datos bajo Windows/Linux. Se diseñó y desarrolló una página Web prototipo, y se la implementó en PHP, ASP y JSP. Para esto se instalaron los servidores Web ISS, PWS, Apache, Tomcat e Instan ASP para realizar las pruebas, teniendo en cuenta las siguientes variables: Tiempo de respuesta, complejidad de la programación, integridad de la base de datos, arquitectura de software y hardware, detección de fallas, confiabilidad y portabilidad[*].

Las tres herramientas son portables de Windows a Linux y viceversa; esto quiere decir que se puede migrar de un sistema operativo a otro sin realizar cambios en el código. Sin embargo, es importante destacar que ASP [22] no fue diseñado para trabajar en ambientes Linux, ya que es un producto exclusivo de Microsoft. Debido a esto, la empresa SUN desarrolló una herramienta llamada One Active Server Page, la cual interpreta el código ASP y permite el funcionamiento de éste bajo Linux[23]. Esto tiene algunas limitantes para los usuarios Linux, ya que la herramienta no es de tipo Freeware, lo cual reduce el uso de ASP sobre Linux y en caso de que el programador la utilice, si desea migrar de un sistema operativo hacia otro, debe asegurarse de utilizar componentes ADO para la conexión con las bases de datos.

4.9 Tomcat

Tomcat funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Dado que fue escrito en Java, funciona en cualquier sistema operativo que disponga de una máquina virtual Java. Es cada vez más utilizado por las empresas en los entornos de producción debido a su contrastada estabilidad [20].

---

[*] Basado en el documento: Análisis comparativo de las herramientas de programación Web: PHP, ASP y JSP, bajo los sistemas operativos Linux y Windows por: Daladier Jabba Molinares, Adalgisa Alcocer Olaciregui, Carmenza Rojas Morales.

[*] Estudio llevado a cabo por los autores referenciados

El servidor de aplicaciones Tomcat de Apache y las tecnologías afines proporcionan a los programadores de Java un completo conjunto de herramientas para crear de forma rápida sofisticadas aplicaciones web.

TOMCAT es mantenido y desarrollado por miembros de la *Apache Software Foundation* y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la *Apache Software Licence*[21]. Las primeras distribuciones de TOMCAT fueron las versiones 3.0.x. Las versiones más recientes son las 6.x, que implementan las especificaciones de SERVLET 2.4 y de JSP 2.0. A partir de la versión 4.0, JAKARTA TOMCAT utiliza el contenedor de SERVLETS Catalina[4].

4.10 Netbeans

Los Entornos de Desarrollo Integrado, también conocidos como IDEs (de sus siglas en inglés Integrated Development Environment), son programas que disponen de un conjunto de herramientas: un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

NetBeans es un proyecto exitoso de código abierto con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo[12]. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

Al día de hoy hay disponibles dos productos: el NetBeans IDE y NetBeans Platform.

NetBeans IDE es un entorno de desarrollo, es decir, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java - pero puede servir para cualquier otro lenguaje de programación.

Existe además un número importante de módulos para extender el NetBeans IDE.Tammbién es referenciado como un producto libre y gratuito sin restricciones de uso. También está disponible NetBeans Platform [13]; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

---

[4] TOMCAT. Definiciones. [En Línea]. Fecha de Consulta: 14-10-2008. Disponible en: http://www.tomcat.org/es

4.11 Proceso Unificado

Corresponde a las siglas UP (Proceso Unificado). Es una herramienta utilizada para definir procesos de desarrollo estructurados y con calidad. Sus fases están enmarcadas por modelos, los cuales son utilizados por los actores que intervienen en el procesos de generación de tecnología [14]. Para éstos procedimientos se toman los modelos de Análisis. Diseño e Implementación.

Modelo de Análisis

La siguiente fase es la de construcción del modelo de análisis. Es éste un modelo de objetos del sistema, cuya operación debe cumplir los requisitos especificados por el usuario, pero a nivel conceptual, es decir, sin incluir consideraciones de implementación [15]. Debe ser un modelo completo, es decir, detallado, que permita verificar que su operación cumple los requisitos de usuario, sin dejar lugar a dudas, es decir, debe estar especificado lo que hace, cuando y porqué, si bien no es necesario bajar al detalle de cómo lo hace. Debe ser un modelo completo en el plano estático y en el dinámico.

Este modelo es necesario para profundizar en cada elemento definido en el modelo de requisitos, se inicia un estudio previo a los requisitos funcionales y no funcionales del sistema; todo con el fin de determinar los casos de uso análisis.

Como lo indica el apartado anterior, unos de los objetivos del modelo de análisis es entender el Uso del Sistema: En la reunión con los usuarios potenciales, los desarrolladores se reunieron con los usuarios para descubrir a los actores que inician cada Caso de Uso y a los actores que se beneficiaran de ellos [16] (Un actor puede ser tanto un sistema como una persona). (Diagrama de Casos de Uso)

Modelo de Diseño

El modelo de diseño es la fase más interesante del modelado de software, ya que nos posibilita la concepción del producto, para lograr una visualización correcta del diseño del sistema y que este acorde al ciclo de vida que plantea UP[17].

Cuando el modelo de análisis está en una etapa madura, se inicia con el proceso de construcción del producto, éste se logra mediante la construcción de un modelo de diseño, que nos permite ilustrar todas las características correspondientes al software; para lograr un buen modelo es necesario atender a las técnicas de ingeniería de software vigentes, si observamos, el siguiente diagrama nos permite organizar dicha labor[18].

4.12 ANTECEDENTES

4.12.1 ALFASIS VISUALNET

Sistema Administrativo y de Cartera para Empresas de Seguros.
Cuenta con herramientas para controlar la PRODUCCIÓN (Ventas), Escaneo de las pólizas, control de tesorería (recaudos), la cartera general de primas al asegurado (tomador), la cartera de comisiones que le adeuda las aseguradoras (Conciliación de comisiones), liquidación de comisiones a los comerciales, concesionarios, Control de contratos de Salud, ARP, POS. Cuenta con un módulo de control de negocios, el cual realiza la gestión comercial de la empresa para conocer las cotizaciones presentadas, negocios realizados, negocios pospuestos. Adicionalmente tiene software para: POLIZAS COLECTIVAS, SOAT, SLIP DE COTIZACIÓN y RESUMEN DE SEGUROS, bitácora de SINIESTROS, CORRESPONDENCIA y MENSAJERIA, CONTABILIDAD (presupuestos, centros de costos, saldos de bancos, cuentas por pagar, comprobantes automáticos).

4.12.2 INSURIX®[5]

Es una solución integral para compañías de seguros patrimoniales y de vida, abarca todas las actividades del negocio. Presenta una fuerte orientación a clientes y productores, automatizando todas las etapas de la comercialización de seguros en una organización.

Insurix® es un sistema abierto y parametrizable, que permite definir productos de manera flexible. Al mismo tiempo, abarca la gestión de todos los riesgos: autos, hogar, vida, caución (fianzas), integrales, riesgo operativo, entre otros. Ofrece una visión integral de la compañía, las operaciones, los clientes y permite el control de todas las variables que inciden en el desempeño del negocio.

Insurix® se integra totalmente con el sistema de administración de pólizas, potenciándolo y a la vez posibilita la comercialización de seguros a través de todos los canales comerciales. Para ello cuenta con un módulo generador de productos

[5] INSURIX, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.inworx.com/insurix.html

que soporta 'scoring' y 'profiling'. Este módulo, también asocia las condiciones comerciales y de suscripción según el canal de venta que corresponda.

Insurix® es la solución ideal para la gestión diaria de clientes, control de excepciones y seguimiento de actividades de la compañía. También provee una herramienta para el desarrollo y monitoreo de nuevas campañas comerciales.

Insurix® provee, además, el módulo Smart Connector® que permite la sincronización "en linea" de las operaciones entre brokers y aseguradoras. De esta manera, disminuye la carga de trabajo para ambos.

4.12.3 TAYSOS[6]

Taysos, empresa dedicada a e-solutions, diseñó la solución Kbee con el objetivo de agilizar la gestión de documentos físicos como las pólizas que contratan sus usuarios, resaltando que estos son los documentos más importantes al momento de emitir un seguro en sus diferentes modalidades y coberturas.

Taysos explica cómo en la empresa se facilita las tareas a las compañías financieras desde el primer contacto con el posible cliente hasta el cierre. Detalla cómo el ECM puede mejorar el proceso de adjudicación de pólizas. Aseguran que Kbee puede ayudar a las aseguradoras a optimizar la administración del repositorio de imágenes digitales, implementar un Workflow editorial así como integrar todos los sistemas administrativos y bases de datos.

4.12.4 CDC Software[7]

Las empresas del sector asegurador están repletas de procesos complejos y todavía más complejas redes de relaciones. Por consiguiente todas estas las compañías se enfrentan a retos crecientes en términos de colaboración, coordinación, y gestión de relaciones. Entre sus principales restos está el del gran número de clientes y de agente, su gestión e intentar entre todas ellas las mejores fuentes de rentabilidad.

4.12.5 SAS ABM[8]

---

[6] TAYSOS, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.ecm-spain.com/interior.asp?IdItem=8922

[7] CDC Software, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.cdcsoftware.es/soluciones/Vertical_Aseguradoras.htm

[8] SAS ABM Software, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en:

Línea Directa Aseguradora utiliza la solución ABM- Activity Based Management- (Gestión de Costes por Actividad) para el cálculo del coste de sus procesos internos en cada una de las áreas de su negocio. La solución de , líder mundial en software de Business Intelligence, permite a la aseguradora calcular los costes directos e indirectos generados por la venta y gestión de pólizas, liquidación, recobro, etc.

Gracias a la Gestión de Costes por Actividad, la compañía podrá conocer y tener un mayor control de los costes internos, optimizando tanto los procesos como las actividades de cada una de las áreas de negocio y facilitando la realización de benchmarkings internos.

El Activity-Based Management va más allá de los métodos tradicionales de repartos de costes, proporcionando informes de gestión que permiten tomar decisiones estratégicas basadas en el verdadero conocimiento de qué segmentos de negocio, qué productos o que clientes resultan más ventajosos. La capacidad de realizar simulaciones de negocio permite además a Línea Directa Aseguradora predecir sus necesidades de recursos y sus costes futuros.

4.12.6 INFOSEG, GESTIÓN ADMINISTRATIVA EN SEGUROS[9]

Es un sistema de gestión administrativa complementario al sistema de gestión general de Compañías y Corredurías de Seguros. Resuelve la problemática de gestión documental, comunicaciones y tareas en entornos de Compañías y Corredurías de Seguros. Pensado para cuando el sistema de gestión no incluye dichas funcionalidades o cuando las incluye de forma incompleta

VENTAJAS:

* Diseñado específicamente para el entorno de Seguros
* Integrable en el sistema de gestión general de la empresa
* Utilizable por Compañías y Corredurías de Seguros
* Utilizable en central y oficinas
* Mínimos requerimientos de hardware
* Un solo sistema para todas las necesidades de gestión administrativa

---

http://www.bureaudeprensa.com/es/view.php?bn=bureaudeprensa_software&key=1163501751&pattern=SAS

[9] INFOSEG Software, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.infoseg.com/is_admin.shtml

COMPONENTES:

* Gestión Documental [Is-Doc] Info
* Gestión de Comunicaciones Internas Info
* Gestión de Comunicaciones Externas Info
* Gestión de Mailings Info
* Gestión de Tareas Info
* Control de Accesos, de Usuarios y del Sistema Info

4.12.7 SIRYS - Corredores de Seguros[10]

SIRYS, sistema de información corporativo para compañías de seguros y adaptable a corredores de seguros, cuenta con características propias para administrar la información de clientes de tipo estatal y privado, con funciones para la gestión del seguro desde la captura de la póliza hasta la indemnización del siniestro, desde la facturación de primas hasta el recaudo directo o la gestión de cuentas por cobrar. Controla la cartera de primas a través de la cuenta corriente por cliente. Genera la calificación de cartera y provisión. Contabiliza cada transacción sobre la póliza consolidándolas con base en el PUC contable.

Las funciones especializadas del cargo de un usuario se pueden configurar para incrementar la productividad y la facilidad de operación, garantizando la seguridad en la administración de la información del cliente. Cuenta con un sistema de alertas que facilita a la aseguradora y al cliente el conocimiento anticipado de cualquier situación que afecte directamente el estado de la información.

4.12.8 TE-SIS SEGUROS[11]

TE-SIS Seguros es un programa de seguros que pretende dar una cobertura eficaz y flexible a la mayoría de las necesidades de tratamiento de datos que surgen en el día a día de los profesionales del Sector Mediador. El programa se adapta perfectamente a cualquier Correduría, Sucursal, Agencia, Agente, Agente independiente o Colaborador, sea cual sea su estructura y tamaño.

---

[10] INFOSEG Software, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.infoseg.com

[11] Te-SIS Seguros, Soluciones para compañías de seguros. Fecha de Consulta: 07-08-2008. Página principal. Disponible en: http://www.te-sis.net/

TE-SIS Seguros es el programa para gestión de corredurías que cautivará desde la primera toma de contacto al profesional más exigente.

4.12.9 CEA SOFTWARE AGENTES DE SEGUROS[12]

Archivos de Compañías, Ramos, Comisiones, Agentes / Subagentes, clientes, Pólizas, Recibos, Canales de cobro, Generación de recibos de Cartera Facturación y Liquidación a compañías, así como liquidación de comisiones a los Subagentes, seguimiento de siniestros, bancos, etc.

Y además a un precio asequible, tanto para el Agente cuya cartera de Seguros aun no sea muy elevada, pero que está interesado en tenerla perfectamente controlada, como para aquel Agente que tenga su propia Agencia o Correduría y trabaje con varias Compañías e incluso tenga Subagentes . Con el programa Segusenci puede tener cubiertas todas sus necesidades sin efectuar un desembolso importante en una aplicación informática.

4.12.10 SAS[13]

Esta aplicación, está diseñada para el profesional, que necesita tener controladas todas las acciones realizada con su trabajo diario. Controla todos los aspectos, desde los clientes, agentes, compañías, partes de siniestros, pólizas, recibos.

Diseño libre de los formatos impresos, copias de seguridad, archivo de agentes y Subagentes, listados de agentes, Cartera de clientes, archivo de vehículos de clientes, archivo de marcas de vehículos, tipos de riesgos, listado de datos bancarios, emisión de etiquetas para correspondencia, emisión de pólizas, , listado de pólizas por vencimiento, entrada de partes de siniestros, control de siniestros.

4.12.11 CEIBA SOFTWARE HOUSE[14]

Servicios Prestados

---

[12] CEA SOFTWARE AGENTES DE SEGUROS, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.infoseg.com/is_admin.shtml

[13] SAS, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.utilidadesgratis.com/software-agentes-de-seguros.html

[14] CEIBA, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.ceibasoft.com

- Ciclo de ingeniería de software completo para el Sistema de EVALUACIÓN MÉDICA (Java sobre ORACLE).

- Ciclo de ingeniería de software completo para el Sistema de SALVAMENTO DE AUTOS AUTOSURA (Java sobre ORACLE).

- Ciclo de ingeniería de software completo para el proyecto PLATAFORMA DE SERVICIOS. (Java sobre ORACLE).

- Ciclo de ingeniería de software completo para el proyecto BITÁCOTA (Java sobre ORACLE).

- Ciclo de ingeniería de software completo para el proyecto ASISTENCIA (Java sobre ORACLE).

- Ciclo de ingeniería de software completo para el proyecto MATRIZ DE RIESGOS (Java sobre ORACLE).

4.12.12 INTERHELPER[15]

Software diseñado para Intermediarios de Seguros con el cual pueden mantener un completo control de sus clientes y de todos los productos que estos han adquirido, los movimientos de dinero de todas las transacciones involucradas en una venta de seguros de cualquier índole y sus renovaciones. Además, se constituye como una herramienta útil para realizar estrategias de mercadeo, atención a clientes, control de metas y resultados, administración de comisiones, y otras ventajas. El InterHelper se adapta perfectamente a cualquier Corredor, Sucursal, Agencia, Agente, Agente independiente o Colaborador, sea cual sea su estructura y tamaño.

4.12.13 ORBIS SEGUROS[16]

ORBIS Seguros es un nuevo y revolucionario sistema que permite la comercialización de seguros de asistencia en viaje directamente desde las aplicaciones ORBIS de Gestión Comercial para Agencias de Viajes.

---

[15] INTERHELPER, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.interhelper.net/

[16] ORBIS SEGUROS, Soluciones para compañías de seguros. Fecha de Consulta: 11-09-2008. Página principal. Disponible en: http://www.pipeline.es/productos/orbis_seguros.htm

Mediante ORBIS Seguros, los agentes pueden realizar la venta del seguro de viaje en apenas unos segundos, directamente desde el propio expediente. ORBIS Seguros muestra las opciones disponibles y realiza, en tiempo real, la contratación del seguro y la generación de la póliza a través de Internet. Todo el proceso es automático, quedando todos los datos integrados en la propia ficha de venta y en el resto de procesos de gestión y contabilidad de la agencia.

5. METODOLOGÍA

El desarrollo del proyecto se guió por la metodología propuesta por lineamientos de la Facultad de Ingeniería, y propuesta por el presidente de la tesis, mientras el lenguaje para el proceso de Ingeniería del Software es el Lenguaje Unificado de Modelado, UML. El desarrollo se implementa usando Java, bajo el uso y los lineamientos del IDE NetBeans. A continuación se realiza una breve descripción de estos elementos:

Lenguaje Unificado de Modelado (UML): Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; aún cuando todavía no es un estándar oficial, está apoyado en gran manera por el OMG* Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables[17].

5.1 TIPO DE TRABAJO

Un proyecto de tipo informático como lo es el Sistema de Información Integrado para el Gilberto Arenas y CIA LTDA está orientada al desarrollo completo y satisfactorio de sistemas de información, lo cual, depende de las metodologías utilizadas, de la interacción de los usuarios responsables del desarrollo y de las herramientas utilizadas para el proceso.

5.2 PROCEDIMIENTO

Para la elaboración de este sistema de información se plantean las siguientes fases de desarrollo:

5.2.1 FASE A: Requisitos y análisis

[17]OBJECT MANAGEMENT GROUP. Introduction to OMG's Unified Modeling Language™ (UML®). Fecha de Consulta: 03-08-2007. Inicio > Introduction to UML. Disponible en: http://www.omg.org/gettingstarted/what_is_uml.htm

En esta fase es utilizado el Lenguaje de Modelado Unificado UML para poder definir todo el diseño tanto de la aplicación como de la Base de datos. Se realizaron las siguientes actividades:

- ACTIVIDAD A.1: Recolección de la Información
- ACTIVIDAD A.2: Planeación
- ACTIVIDAD A.3: Levantamiento de información
- ACTIVIDAD A.5: Análisis
- ACTIVIDAD A.6: Documentación del análisis
- ACTIVIDAD A.7: Construcción de diagramas
- ACTIVIDAD A.8: Validación de diagramas

5.2.2 FASE B: Análisis y diseño de la base de datos

- ACTIVIDAD B.1: Análisis de la BD, y la normalización de la Base de Datos.
- ACTIVIDAD B.2: Creación de todos los elementos de la Base de Datos: tablas, vistas, consultas, entre otros.
- ACTIVIDAD B.3: Se realizaron de pruebas de funcionamiento para la BD.

5.2.3 FASE C: Instalación de los servicios

Se buscaron y se instalaron los servicios necesarios tanto para el desarrollador como para el usuario final de la aplicación.

- ACTIVIDAD C.1: Determinar cuáles son las mejores aplicaciones y cual nos brindaba un mejor desempeño para este tipo de aplicaciones.

- ACTIVIDAD C.2: Instalación y configuraron cada uno de los servicios que se escogieron para el desarrollo.

Los servicios y aplicaciones instaladas son: JAVA, TOMCAT, POSTGRES, los drivers de conexión.

5.2.4 FASE D: DESARROLLO DEL MÓDULO DE CLIENTES

- ACTIVIDAD D.1: Se Validación de requisitos

- ACTIVIDAD D.2: Codificación de interfaces administrativas: insertar, eliminar, consultar para el módulo de gestión de Clientes

- ACTIVIDAD D.3: Crear los JSP para el módulo de Gestión de Clientes.

5.2.5 FASE E: Desarrollo del módulo de Tesorería

- ACTIVIDAD E.1: Codificar las cadenas de conexión del módulo.
- ACTIVIDAD E.2: Codificación de interfaces administrativas: insertar, eliminar, consultar para el módulo de Gestión de servicios.
- ACTIVIDAD E.3: Crear los JSP para el módulo de Gestión de tesorería.

5.2.6 FASE F: Desarrollo del módulo de Gestión de hoja de ruta

- ACTIVIDAD F.1: Codificar las cadenas de conexión del módulo.
- ACTIVIDAD F.2: Codificación de interfaces administrativas: insertar, eliminar, consultar para el módulo de gestión de rutas.
- ACTIVIDAD F.3: Crear los JSP para el módulo de Gestión de hoja de ruta

5.2.6 FASE G: Desarrollo del módulo de cotizaciones

- ACTIVIDAD G.1: Se codificaron Codificar las cadenas de conexión del módulo.
- ACTIVIDAD G.2: Codificación de interfaces administrativas: insertar, eliminar, consultar para el módulo de Gestión de Crédito y Cartera.
- ACTIVIDAD G.3: Crear los JSP para el módulo de Gestión de cotizaciones.

5.2.8 FASE H: Desarrollo del módulo de pagos en línea

- ACTIVIDAD H.1: Codificar las cadenas de conexión del módulo.
- ACTIVIDAD H.2: Codificación de interfaces administrativas: insertar, eliminar, consultar para el módulo de Servicio al Cliente
- ACTIVIDAD H.3: Crear los JSP para el módulo de pagos en línea.

5.2.9 FASE I: Desarrollo el módulo de Gestión de mensajería interna

- ACTIVIDAD I.1: Codificar las cadenas de conexión del módulo.
- ACTIVIDAD I.2: Se Codificación de interfaces administrativas: insertar, eliminar, consultar para el módulo de Gestión de Deducciones
- ACTIVIDAD I.3: Crear los JSP para el módulo de Gestión de mensajería interna.

5.2.10 FASE J: Desarrollo e implementación de para emitir informes requeridos por la empresa.

- ACTIVIDAD J.1: Diseñar informes bajo plataforma WEB.
- ACTIVIDAD J.2: Implementar informes.

5.2.11 FASE K: Realización de las pruebas necesarias para evaluar el sistema.

- ACTIVIDAD K.1 Pruebas de caja blanca al sistema.
- ACTIVIDAD K.2: Pruebas de caja negra al sistema.

6. RESULTADOS

6.1 DESCRIPCIÓN DE RESULTADOS

Como objetivo principal se planteó analizar, diseñar y desarrollar un sistema WEB para la gestión de la información en la aseguradora Gilberto Arenas y CIA LTDA; donde se contextualizó en los módulos para la gestión del conocimiento del cliente (CRM), control de tesorería y cartera, la hoja de ruta (control de negocios) y cotizaciones en línea. Todo con el fin de suplir las necesidades básicas de procesamiento de la información y bajo plataformas Web.

Tomando los lineamientos generales de los requisitos y teniendo en cuenta los términos generales en los objetivos, se cumplió a cabalidad con las expectativas planteas inicialmente; el proyecto satisface a la compañía con un producto de calidad, que permita mejorar sus procesos, tanto en su gestión interna como para los procedimientos con los clientes.

El módulo para la gestión del conocimiento del cliente (CRM), ayuda a mejorar la comunicación con los clientes, logrando como resultado la posibilidad de aumentar la gestión comercial de la compañía.

Tanto para el control de tesorería y cartera, como para la hoja de ruta (control de negocios), es de vital importancia lograr un impacto positivo en el manejo la información, aumentando el control y permitiendo hacer auditoría.

El módulo para cotizaciones en línea, debe permitir presentar las propuestas a los clientes en forma más ágil y oportuna, logrando aumentar la capacidad de atención a nuevos clientes.

Por último, el proyecto deberá integrarse, permitiendo comunicarse por medio de una interfaz al programa contable existente en la empresa.

Como resultado general, se logró una aplicación orientada a Web, la cual, usó como plataforma la herramienta de desarrollo especificada en el marco teórico, NetBeans, su estructura interna fue implementada bajo la orientación a objetos.

El desarrollo orientado a objetos se logró mediante al utilización de JavaBeans*, todo con el fin de implementar el producto en tecnologías que ayuden a mejorar los procesos.

Para el ingreso a la página principal, tal y como se observa en la figura 1, se construyó una página Web, la cual permite ilustrar algún tipo de información referente a la compañía.

Figura 1: Página principal

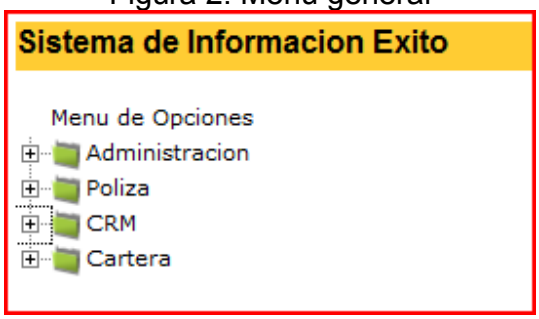
Fuente: La aplicación

La estructura general de la aplicación esta manejada por un menú configurable, el cual utilizó un script usado universalmente, el cual tiene características de seguridad y de manejo ágil.

* Los JavaBeans son un modelo de componentes creado por Sun Microsystems para la construcción de aplicaciones en Java. Se usan para encapsular varios objetos en un único objeto, para hacer uso de un sólo objeto en lugar de varios más simples.

Dicho resultado se contextualiza en la siguiente Figura, la cual define su característica técnica.

Figura 2: Menú general



Fuente: La aplicación

Para el ingreso de la información a las bases de datos, se definieron elementos de interfaz con características específicas de fácil manejo. En la Figura 3 se ilustra cómo se crearon éstos elementos.

Figura 3: Ingreso

Fuente: La aplicación

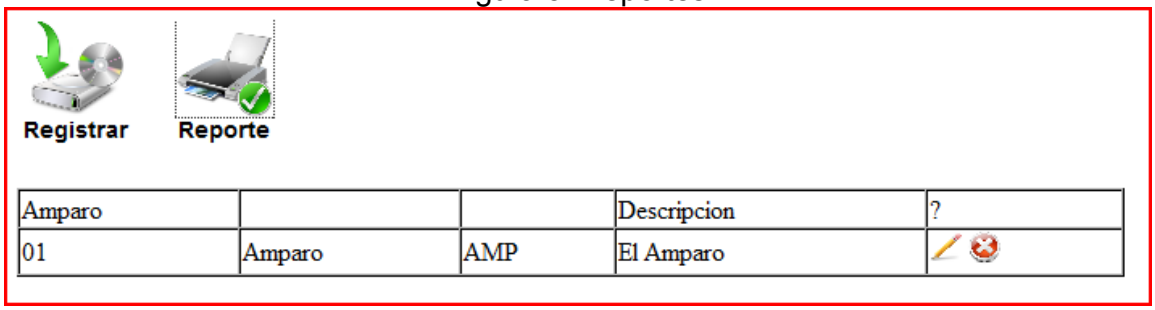
Otro aspecto fundamental, y de funcionamiento hace referencia a la consulta de la información, ésta se logra mediante un menú que aparece en el Figura 4.

Figura 4: Menú

Fuente: La aplicación

La anterior figura, hace referencia al menú resultante, el cual tiene un botón para reportes, el cual mostrará la siguiente ventana(Figura 5):

Figura 5: Reportes



Fuente: La aplicación

La figura anterior permite la edición y eliminación de registros.

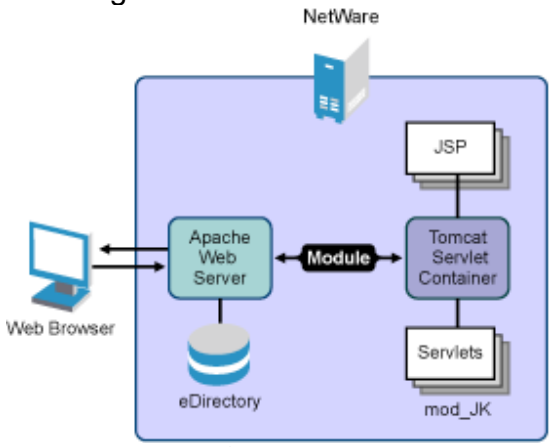
6.2 DISCUSIÓN DE RESULTADOS

Los resultados en términos generales están orientados a la Web, la Figura abajo descrita, se toma como ejemplo un servidor Netware, todo con el fin de especificar su multiplataforma, ya que se puede implantar en servidores Windows, Unix y Netware.

Su funcionamiento radica en solicitudes de los web browsers, éste realiza peticiones al servidor, el cual procesa la solicitud y envía el resultado al browser. Dicho servidor procesa lo elementos escritos en JSP, logrando la generación de

servlets, los cuales son elementos muy seguros; en el caso especifico para la compañía analizaremos la gestión de un Ajustador.

Figura 6: Servidor TOMCAT

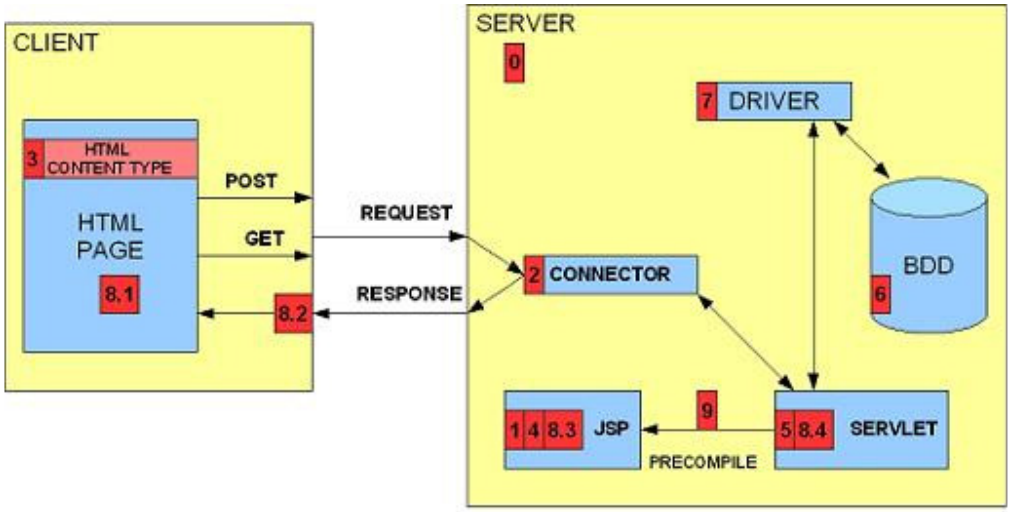


Fuente: Sun Microsystem

Las características anteriores no son comunes en los antecedentes, puesto que se encuentran aplicaciones basadas a escritorio, y desarrolladas con estándares de mayor antigüedad.

Para el acceso a las bases de datos, se logró la implementación de un esquema cliente – servidor, dónde se seleccionó el motor Postgres para la gestión de la información, en la Figura 7, se observan algunos de los elementos más importantes y relevantes de implantación del proyecto

Figura 7: Base de datos



Fuente: Sun Microsystem

El desarrollo de la aplicación se llevó a cabo tomando las características generales de JavaBeans, a continuación se defines los detalles de construcción para lograr los objetivos planteados.

7. CONCLUSIONES

Haciendo síntesis sobre los resultados más importantes y significativos, se concluye el proyecto, el cual, logró la construcción de un sistema bajo plataformas Web, permitiendo cumplir con los objetivos planteados en la Empresa. A continuación se enmarcan en contexto general, los puntos sobre los cuales se tiene consideración más relevante.

- Un CRM genérico es una gran adaptación de los sistemas de información; teniendo en cuenta su alta complejidad, es necesario aclarar que el desarrollo para manejos de clientes del presente proyecto, es una aplicación implementada a la medida, la cual busca cumplir con las necesidades de la aseguradora. Como se observa en las recomendaciones, se plantea la necesidad de continuar con un proceso que busque en la aseguradora integrar un CRM genérico.

- Se concluye que la aplicación de una metodología de desarrollo, con miras a la consecución de un producto eficiente y de calidad, forma parte importante. Estas características técnicas corresponden a los procesos y procedimientos específicos para lograr la obtención de los resultados. En términos generales, el uso de herramientas de análisis y diseño, es un actor fundamental para la organización en el desarrollo del producto, el cual, fue ordenado de forma clásica mediante el Proceso Unificado.

- En conclusión, las plataformas Web, como solución en sistemas de información para empresas son fundamentales, todo debido a que las tendencias actuales en las Tecnologías de la Información tienen como objetivo principal, la distribución de recursos. En el desarrollo del proyecto, se tomaron los lineamientos iníciales para desarrollo distribuido, es decir, su estructura técnica esa orientada a plataformas Web, lo que permitió un mejor acceso a los medio e interactuar con los clientes de forma ágil y oportuna.

- Para Gilberto Arenas y CIA LTDA, es importante la comunicación con los clientes; haciendo referencia en términos generales, existe una tendencia de alza en el uso de dicha tecnología. En respuesta a esta nueva demanda, las empresas están implantando diversas arquitecturas destinadas a obtener toda la información necesaria de sus clientes, convirtiendo estos datos en una parte importante y de gran valor par a estas corporaciones. Las estructuras tradicionales se quedan obsoletas y no cuentan con las herramientas adaptadas a estas necesidades concretas. Multitud de

organizaciones ya están desarrollando sistemas para facilitar la toma de decisiones, centrando sus actividades en hacer operativos y eficaces estos procesos basados en soluciones Customer Relationship Management (CRM), y para el caso del desarrollo del proyecto, no es la excepción.

- El control de hoja de ruta, es un elemento del sistema que aumenta ostensiblemente la toma de decisiones, es decir, cuando se tiene un control sobre las negociaciones de los productos, se logra realizar mayores controles sobre la calidad del servicio. La calidad del servicio es uno de los objetivos principales de la empresa Gilberto Arenas y CIA LTDA.

8. RECOMENDACIONES

- Se plantea como recomendación, la continuación, mediante otros trabajos de desarrollo tecnológico, la implementación de los módulos de:

    - Contabilidad
    - Presupuesto
    - Facturación con un ente externo

- Para efectos de un impactos sostenible, es necesario crear un comité evaluador que vele por la continuidad del proyecto, controlando su uso y verificación de todos los procesos en Internet

- Para la eficiencia del proyecto, se plantea mejorar el servidor en cuando a hardware, aumentando su capacidad de procesamiento y de almacenamiento, todo con el fin de lograr mejorar los tiempos de proceso

- Como recomendación de continuidad, se plantea la necesidad de continuar con el módulo que apunta a la construcción de un CRM, puesto que el presente proyecto logró un alcance limitado respecto a las implicaciones reales del tema, permitiendo proponer la continuidad de éste módulo hacia la construcción de un CRM genérico, que pueda se implantado en cualquier entidad.

- Para efectos de seguridad informática, se recomienda la implantación de un servidor con firewall, que permita aumentar la seguridad e integridad de la información.

BIBLIOGRAFÍA

[1] G. Booch, I. Jacobson, J. Rumbaugh. El Lenguaje Unificado de Modelado. Guía del usuario. Addison-Wesley/Díaz de Santos,1999.

[2] Larman, Craig, UML y patrones: introducción al análisis y diseño orientado a objetos Prentice-Hall, 2ª ed. 2002.

[3] Jacobson, Ivar, Booch, Grady and Rumbaugh, James. "*El Proceso Unificado de Modelado*". Addison-Wesley, 2000.

[4] Muller, Pierre-Alain. Modelado de objetos con UML. Madrid. Gestión 2000. 1997. ISBN: 84-8088-226-3

[5] GRIFF, Mayer. SQL y Java: guía para SQLJ, JDBC y tecnologías relacionadas. Madrid. RA-MA. 2002. ISBN: 84-7897-506-3

[6] C. J. Date. Faudón, Sergio Luis María Ruiz. Introducción a los sistemas de bases de datos. Pearson Educación. 2001. ISBN 9684444192. 936 páginas

[7] C. Worsley, Joshua D. Drake. Practical PostgreSQL. O'Reilly. 2002. ISBN 1565928466. 619 páginas

[8] Douglas, Susan. PostgreSQL: a comprehensive guide to building, programming, and administering PostgreSQL databases. Sams Publishing. 2003. ISBN 0735712573. 790 páginas

[9] Moldes, F. Javier. JAVA Práctico. Madrid. ANAYA MULTIMEDIA. 2007. ISBN: 9788441522886

[10] Lemay, Laura. Cadenhead, Rogers. JAVA Para Programadores. Madrid. ANAYA MULTIMEDIA. 2008. ISBN: 9788441522992

[11] Aumaille, Benjamin . J2EE desarrollo de aplicaciones Web. ENI . Madrid. 2002. ISBN 2746019124, 9782746019126. 357 páginas

[12] Tim Boudreau, Jesse Glick, Simeon Greene, Jack Woehr, Vaughn Spurlin. NetBeans: the definitive guide. Los Angeles. O'Reilly. 2003. ISBN 0596002807, 9780596002800. 646 páginas

[13] Keegan, Patrick. Ludovic Champenois. NetBeans IDE Field Guide: Developing Desktop, Web, Enterprise, and Mobile Applications. New York. Prentice Hall

Professional Technical Reference. 2005. ISBN 0131876201, 9780131876200. 397 páginas

[14] Weitzenfeld, Alfredo. Ingeniería de software orientada a objetos con UML, Java e Internet. Cengage Learning Editores. 2005. ISBN 9706861904, 9789706861900. 704 páginas

[15] Cerrada Somolinos, José Antonio. Introducción a la ingeniería del software. Madrid. Editorial Ramón Areces. 2000. ISBN 8480044179, 9788480044172. 334 páginas

[16] Ian Graham, Aminta Yanes Nieves. Métodos orientados a objetos. Madrid. Ediciones Díaz de Santos. 2001. ISBN 0201653559, 9780201653557. 640 páginas

[17] Ivar Jacobson, Grady Booch, James Rumbaugh. El proceso unificado de desarrollo de software. New York. Addison Wesley. 2006.

[18] Mónica Aguilar Garrido. El proceso unificado de desarrollo: Estudio teórico-práctico. Universidad de de Vigo, Escola Superior de Ingeniería Informática. 2003. 100 páginas

[19] Jayson Falkner, Ben Galbraith, Romin Irani. JSP un enfoque práctico. Madrid. ANAYA MULTIMEDIA. 2002.

[20] Jason Brittain, Ian F. Darwin. Tomcat 6.0. La guía definitiva. Madrid. ANAYA MULTIMEDIA. 2008. ISBN: 9788441524316.

[21] Vivek Chopra, Sing Li, Jeff Genender. Apache Tomcat 6. Madrid. ANAYA MULTIMEDIA. 2008. ISBN: 9788441523777

[22] MERCER, D. Fundamentos de Programación en ASP. McGraw–Hill. 2001. Interamericana, p. 573.

[23] BOLLINGER, G. & NATARAJAN, B. JSP: A Beginner's Guide. New York. McGraw Hill. 2001.

[24] MCCARTY, W. PHP 4: A Beginner's Guide. Miami. Osborne. McGraw–Hill. 2001. p. 544.

ANEXOS

ANEXO A
REQUISITOS

CONTENIDO

1. Generalidades del proceso

Para el desarrollo del proyecto, se inicio un proceso verificación y validación con los usuarios, todo con el fin de establecer las necesidades reales del cliente.

Las necesidades reales son lo resultante de un proceso de encuestas, las cuales se logran por medio de actividades con los empleados, se verificó dichos procesos y procedimientos.

En términos generales se adoptó un formato, el cual es un documento prototipo para documentos de requisitos del software. Su Definición técnica está basada conforme con el estándar IEEE Std 830-1998[*].

Las secciones que no se consideren aplicables al sistema descrito podrán de forma justificada indicarse como no aplicables (NA).

Un SRS es inequívoco si, y sólo si, cada requisito declarado tiene sólo una interpretación. Como un mínimo, se requiere que cada característica de la última versión del producto se describa usando un único término. En casos dónde un término en un contexto particular tenga significados múltiples, el término debe ser incluido en un glosario dónde su significado es hecho más específico.

Un SRS[**] es una parte importante del proceso de requisitos del ciclo de vida de software y se usa en el diseño, aplicación, supervisión, comprobación, aprobación y pruebas como está descrito en IEEE Std 1074-1997.

El SRS debe ser inequívoco para aquéllos que lo crean y para aquéllos que lo usan. Sin embargo, estos grupos no tienen a menudo el mismo fondo y por consiguiente no tienden a describir los requisitos del software de la misma manera.

El software tiene como objetivo principal, gestionar la administración en los procesos de la Empresa **Gilberto Arenas y CIA LTDA**, sobre los cuales se requiere un sistema orientado a la Web, todo con el fin de aportar aspectos fundamentales a sus objetivos de expansión.

---

[*] El estandar está basado en IEEE; la página standards.ieee.org/ reading/ ieee/ std_public/ description/ se/ 830-1998_desc.html, no fue tomada como bibliografía para el tema, pero permite ilustrar al lector en forma adecuada, por favor consultar el marco teórico.

[**] Corresponden a las siglas en ingles software requirements specification, el cual hace referencia a especificación de requerimientos de software.

En la introducción de la Especificación de requisitos de software (SRS) se proporciona una vista general de la SRS. Incluye el objetivo, el alcance, las definiciones y acrónimos, las referencias, y la vista general del SRS.*

En términos generales, se hace necesario crear una aplicación Web que cumpla con especificaciones generales en cuanto a seguridad informática, permitiendo el ingreso y la consulta de información referente a los aspectos procedimentales de la Empresa.

1.1 Propósito

En términos generales, el software pretende crear un sistema que permite conocer mejor al cliente, administrando todos los recursos usando en la empresa.

Por otro lado se pretende crear un sistema que permite realizar cotizaciones en línea, permitiendo al usuario conocer en forma más ágil los productos y servicios.

En términos generales se requiere la conexión segura con servidores para datos en línea, los cuales, deberán cumplir con las características generales de dicho sistema.

Propósito del documento:

En términos generales, el trapajo se presupuestó para estudiar los siguientes elementos:

- Analizar, diseñar e implementar un módulo para la gestión del conocimiento del cliente (CRM).
- Analizar, diseñar e implementar un módulo para el control de tesorería y cartera.
- Analizar, diseñar e implementar un módulo para la hoja de ruta (control de negocios).
- Analizar, diseñar e implementar un módulo para cotizaciones en línea.
- Analizar, diseñar e implementar un módulo para pagos en línea dependiendo del perfil.
- Analizar, diseñar e implementar un módulo para gestión de la mensajería interna.
- Analizar, diseñar e implementar un módulo para emitir informes requeridos por la empresa.

* Software requirements specification

- Analizar, diseñar e implementar un módulo administrativo para perfiles y roles del sistema.
- Instalar un sitio web para las consultas en línea.
- Realizar interfaz con el software contable.
- Empalmar los servicios de proveedores de pagos electrónicos con la interfaz del sistema.
- Desarrollar toda la documentación de manuales para el manejo de usuarios tanto especializados como finales.

1.2 Alcance

Para la identificación de los productos a desarrollar, fue necesario crear un sistema de encuestas, las cuales fueron aplicadas a los actores principales del sistema, los cuales arrojaron las siguientes conclusiones:

El proyecto implementa los siguientes módulos:

- Gestión del conocimiento del cliente (CRM).
- Módulo para el control de tesorería y cartera.
- La hoja de ruta (control de negocios).
- Cotizaciones en línea.
- Módulo para pagos en línea dependiendo del perfil.
- Gestión de la mensajería interna.
- Administrativo para perfiles y roles del sistema.
- Empalmar los servicios de proveedores de pagos electrónicos con la interfaz del sistema.

1.3 Personal involucrado

Relación de personas involucradas en el desarrollo del sistema, con información de contacto.

Esta información es útil para que el gestor del proyecto pueda localizar a todos los participantes y recabar la información necesaria para la obtención de requisitos, validaciones de seguimiento, etc.

1.4 Definiciones, acrónimos y abreviaturas

Definición de todos los términos, abreviaturas y acrónimos necesarios para interpretar apropiadamente este documento. En ella se pueden indicar referencias a uno o más apéndices, o a otros documentos.

Browser o navegador: Aplicación utilizada para navegar por el internet, y que despliega las páginas traídas desde el servidor en la pantalla del computador del usuario.

Cliente: Computador y programa computacional que solicita servicios a otro computador en Internet, llamado servidor.

Página Web: Documento del web con información (texto, imágenes, video, audio, etc.) que se presenta en una misma pantalla. Una página Web está en un servidor Web, y es traída al computador del usuario para visualizarla.

JSP: Lenguaje que interactúa con el usuario y la base de datos para trabajar en páginas web.

Servidor: Computador y programa computacional que brinda los servicios solicitados por otro computador llamado cliente.

HTML: Lenguaje de Marcado de Hipertexto, donde se escriben las páginas web a las que se accede a través de navegadores o browsers.

postgreSQL: Software gestor de base de datos de licencia Open Source, es un motor de bases de datos muy robusto, y cuenta con gran aplicación para software orientado a WEB.

JavaScript: Lenguaje de programación que permite la ejecución de código dentro de páginas HTML.

Lenguaje natural: Resulta una técnica muy ambigua para la definición de los requisitos. Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello. A pesar de que son muchos los trabajos que critican su uso, es cierto que nivel práctico se sigue utilizando.

Plantillas o patrones: Esta técnica, recomendada por varios autores, tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea ésta, menos ambigüedad ofrece. Sin embargo, si el nivel de detalle elegido es demasiado

estructurado, el trabajo de rellenar las plantillas y mantenerlas, puede ser demasiado tedioso.

Escenarios: La técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La especificación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia una descripción gráfica en forma de diagramas de flujo. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales de sistema.

Casos de uso: Como técnica de definición de requisitos es como más ampliamente han sido aceptados los casos de uso. Actualmente se ha propuesto como técnica básica del proceso UP (Proceso Unificado). Sin embargo, son varios los autores que defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

1.5 Referencias

Referencia Titulo	Ruta	Fecha	Autor
[001][Documento Institucional]	Suministrador por la Empresa	2006	Directivas de la Empresa

Relación completa de todos los documentos relacionados en la especificación de requisitos de software, identificando de cada documento el titulo, referencia (si procede), fecha y organización que lo proporciona.

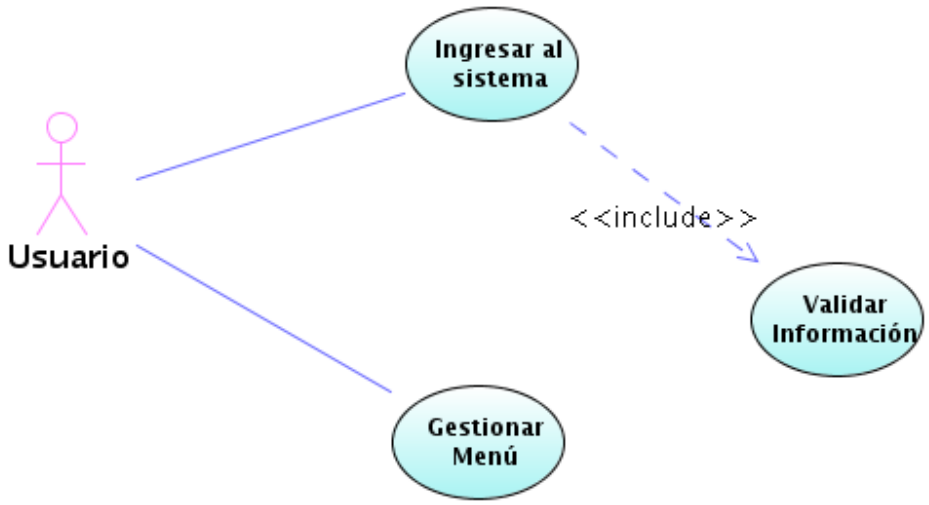
1.6 Resumen

La empresa Gilberto Arenas y CIA LTDA es una agencia asesora de seguros que tiene como domicilio principal la ciudad de Pereira y tiene cobertura completa en el eje cafetero (Manizales – Pereira – Armenia). Su función principales el asesoramiento y venta de todo tipo de seguros, lo que cubre seguros personales como seguros de vida, de salud, vivienda y vehiculares y seguros corporativos los cuales hacen referencia a las pólizas para las instituciones.

El software para implementar está enmarcado en características generales de aplicativos Web, dónde se hace referencia a un producto que puede ser accesado por medio de una intranet, o en caso tal, la red Internet.

El anexo de requisitos, pretende vislumbrar las necedades básicas del software, todo con el fin de establecer los módulos a realizar.

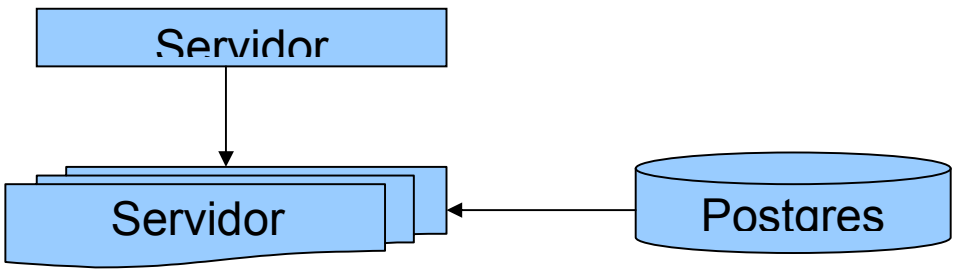
El documento contiene todos los elementos estructurales para definir los requisitos en forma ordenada y comprometida, cada elementos es auditable y validable por los beneficiarios del software.



2. Descripción general

2.1 Perspectiva del producto

El producto desarrollado es principalmente es un sistema que gestiona información general, cotizaciones on-line, gestión de Solicitudes, gestión de comunicaciones con los clientes, gestión de los siniestros, información on-line a los clientes, gestión de seguimientos, gestión de pagos, gestión de cotizaciones, gestión de pagos, gestión de cartera.



El producto debe estar relacionado con un servidor (Tomcat), el cual tiene conexiones con un motor de base de datos que soporte la red.

Por otro lado el sistema podrá ser accesado por los usuarios desde un navegador, el cual tendrá las características mínimas de navegación.

El producto se conectará con el sistema de contabilidad por medio de una interfaz, el cual está representado por un archivo plano.

2.2 Funcionalidad del producto

Debe permitir el ingreso y validación de los usuarios vía Web, manejando perfiles.

Es necesario permitir el ingreso de la información de las pólizas, haciendo referencia a todos los productos que maneja la Empresa.

Se debe hacer un análisis detallado de los procesos y procedimientos en la gestión de clientes, todo con el fin de mejorar la comunicación con los clientes.

Es necesario implementar una interfaz con el software de contabilidad, se deberán crear archivos planos para tales efectos.

2.3 Características de los usuarios

Tipo de usuario	Secretaria
Formación	Secretaría Contable
Habilidades	Habilidad para tratar personas
Actividades	Secretaria Contable

2.4 Restricciones

El desarrollo del software debe ser bajo plataforma segura, el cual funcione bajo multiplataforma. El entorno de desarrollo debe ser NetBeans y el motor de base de datos Postgres.

El código fuente debe estar organizado por medio de paquetes Java, y cada tabla debe estar representada por clases, las cuales serán el elemento de comunicación con los formularios.

Como norma general se adoptan los JavaBeans para la parametrización de los formularios, permitiendo la separación de los modelos de desarrollo, dónde se encuentra las clases y las interfaces por separado.

2.5 Suposiciones y dependencias

El cambio de la herramienta contable, por parte de la Empresa, debe modificar los objetivos de diseño e implantación de la interfaz.

2.6 Evolución previsible del sistema

Para la evolución del producto, es necesario concretar la instalación y configuración de un servidor Web registrado en un dominio que maneje Tomcat con Postgres.

3. Requisitos específicos

En esta fase, se logra obtener una lista detallada y completa de los requisitos que debe cumplir el sistema a desarrollar. El nivel de detalle de los requisitos debe ser el suficiente para que el equipo de desarrollo pueda diseñar un sistema que satisfaga los requisitos y los encargados de las pruebas puedan determinar si éstos se satisfacen.

Cada requisitos, se dispondrán en forma de listas numeradas para su identificación, seguimiento, trazabilidad y validación, tal y como se observa en la lista bajo indicada.

RF1. Analizar, diseñar e implementar un módulo para la gestión del conocimiento del cliente (Desarrollo a la medida según necesidades).

RF2. Analizar, diseñar e implementar un módulo para el control de tesorería y cartera.

RF3. Analizar, diseñar e implementar un módulo para la hoja de ruta (control de negocios).

RF4. Analizar, diseñar e implementar un módulo para cotizaciones en línea.

RF5. Analizar, diseñar e implementar un módulo para pagos en línea dependiendo del perfil.

RF6. Analizar, diseñar e implementar un módulo para gestión de la mensajería interna.

RF7. Analizar, diseñar e implementar un módulo para emitir informes requeridos por la empresa.

RF8. Analizar, diseñar e implementar un módulo administrativo para perfiles y roles del sistema.

RF9. Instalar un sitio web para las consultas en línea.

RF10. Realizar interfaz con el software contable.

RF11. Desarrollar toda la documentación de manuales para el manejo de usuarios tanto especializados como finales.

3.1 Requisitos comunes de los interfaces

Tipo	Descripción
Entrada	Abogado
Entrada	Actividad
Entrada	Afiliado
Entrada	Agente
Entrada	Ajustador
Entrada	Amparo
Entrada	Banco
Entrada	Ciudad
Entrada	Clase Auto
Entrada	Cláusula
Entrada	Cliente
Entrada	Concepto
Entrada	Amparo
Entrada	Deducción
Entrada	Departamento
Entrada	Empresa
Entrada	Familiar
Entrada	Forma Pago
Entrada	Mensajero
Entrada	Marca

Entrada	Moneda
Entrada	País
Entrada	Rama
Entrada	Grupo Rama
Entrada	Sucursal
Entrada	Taller
Entrada	Técnico
Entrada	Tipo Cliente
Entrada	Tipo Negocio
Entrada	Tipo Póliza
Póliza	Agente
Póliza	Amparo
Póliza	Artículo
Póliza	Automóvil
CRM	Tipos Cliente
CRM	Mensaje
CRM	Actividad

3.1.1 Interfaces de usuario

Para describir los requisitos del interfaz de usuario para el producto, se definen algunos aspectos generales. Esto puede estar en la forma de descripciones del texto o pantallas del interfaz. Por ejemplo posiblemente el cliente ha especificado el estilo y los colores del producto. Se describe exacto cómo el producto aparecerá a su usuario previsto.

Lo anterior aclara que el producto deberá contener una página Web inicial, la cual quedará a disposición del administrador, el cual tendrá la responsabilidad de gestionar su contenido.

3.1.2 Interfaces de hardware

Se debe tener un servidor Web que pueda brindar acceso a los computadores mediante una interfaz directa al interior de la empresa, o por medio de un proveedor de hardware.

3.1.3 Interfaces de software

Es necesario generar un archivo plano para luego ser interpretado por el software de contabilidad.

3.1.4 Interfaces de comunicación

También es necesario implementar los procesos necesarios para enlazar la página de la Empresa con el sistema de pagos.

3.2 Requisitos funcionales

En ésta fase, se tiene la definición de acciones fundamentales que debe realizar el software al recibir información, procesarla y producir resultados.

3.2.1 Requisito funcional: Formato Web

Como requisito funcional se detalla el formato Web del proyecto, el cual, es necesario que sea accesado por medio de un navegador.

3.2.2 Requisito funcional: Base de datos segura

Para efectos de distribución Web, es necesario implementar la base de datos en un motor seguro, tal y como se indicó en apartes anteriores, el software es Postgres.

3.2.3 Requisito funcional: Ingreso de pólizas

Las pólizas deben ser ingresadas para cada caso:

- Agente
- Amparo
- Artículo
- Automóvil
- Bien
- Cobro
- Vida

3.3 Requisitos no funcionales

3.3.1 Requisitos de rendimiento

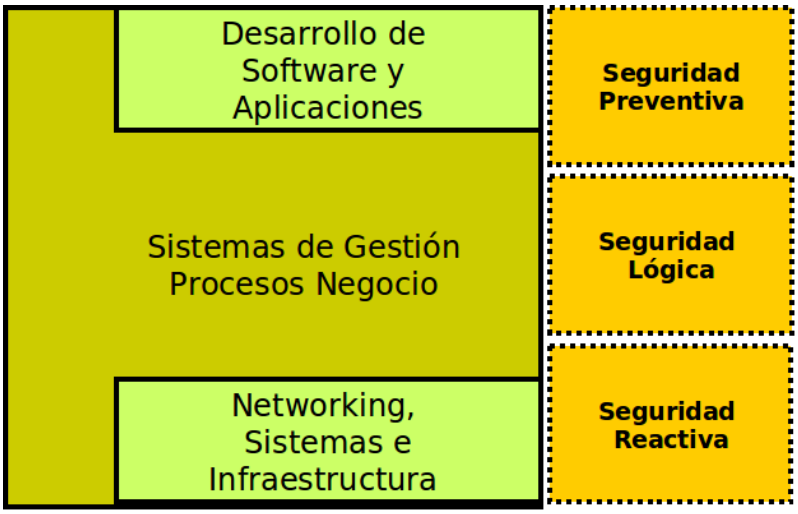
No existen requisitos de rendimientos.

3.3.2 Seguridad

Es posible establecer diferentes permisos a los usuarios de forma que se puede limitar, o incluso denegar, el acceso de un usuario o grupo a los diferentes apartados del software, ingreso de datos, creación de informes, entre otros.

La seguridad del servidor estará a cargo del administrador de la red, el cual, tiene la responsabilidad del uso de un sistema operativo con configuraciones básicas para su operación segura.

Según la figura abajo descrita, el proceso de negocio debe contemplar seguridad preventiva, seguridad lógica y seguridad reactiva.



La seguridad requiere más manejo y riesgo de mitigación, de la que requiere la tecnología. Como un desarrollador, uno primero debe de determinar los riesgos de una aplicación particular. Por ejemplo, el Web site típico de hoy puede ser sujeto de una variedad de riesgos; la desfiguración o la negación distribuida de ataques del servicio.

Una vez que se identifiquen los riesgos, identificar medidas de seguridad apropiadas llega a ser manejable. En particular, al definir los requisitos, es importante considerar cómo la aplicación será utilizada. Con ese conocimiento uno puede decidir, si o no, utilizar características complejas como contabilidad, auditoría, entre otros.

3.3.3 Fiabilidad

La Fiabilidad de software significa que un programa particular debe de seguir funcionando en la presencia de errores. Los errores pueden ser relacionados al diseño, a la implementación, a la programación, o el uso de errores. Así como los sistemas llegan a ser cada vez más complejos, aumenta la probabilidad de errores.

Según la definición de los requisitos para el software, se plantea la necesidad crear un producto fiable, con base a su gestión orientada a la Web.

3.3.4 Disponibilidad

La disponibilidad del servicio prestado por el producto debe ser de tiempo completo, para esto se instalan y configuran servidores Web.

En caso tal, es necesario configurar en el sistema operativo del servidor, los accesos de los usuarios según lo establezcan las políticas de seguridad informática en un futuro.

3.3.5 Mantenibilidad

Para la Identificación del tipo de mantenimiento necesario del sistema y la especificación de quien debe realizar las tareas de mantenimiento, se plantean los siguientes lineamientos:

- El mantenimiento a software, sistemas y hardware, será proporcionado por el personal de la Unidad de apoyo Informático.

- El Jefe de la Unidad de Apoyo Informático será el enlace entre las áreas usuarias del órgano Interno de Control y la Subdirección de Tecnología de la Información o cualquier otra instancia, para adecuar e implantar las modificaciones a los Sistemas Institucionales.

- El personal de la Unidad de Apoyo Informático y Control Documental, atenderá el 95% de las solicitudes de mantenimiento como máximo al día laboral siguiente a su solicitud.

- En caso de que el mantenimiento requiera de cambio de refacciones o de la intervención de un proveedor externo, el plazo para la conclusión del mantenimiento se extenderá el tiempo necesario, informando al usuario del estado en que se encuentra su equipo.

También se debe tener en cuenta la especificación de cuando debe realizarse las tareas de mantenimiento. Por ejemplo, generación de estadísticas de acceso semanal y mensual.

3.3.6 Portabilidad

Para la especificación de atributos que debe presentar el software para facilitar su traslado a otras plataformas u entornos. Se incluyen:

Descripción	Valor
Porcentaje de componentes dependientes del servidor	%100
Porcentaje de código dependiente del servidor	%100
Uso de un determinado lenguaje por su portabilidad	%100
Uso de un determinado compilador o plataforma de desarrollo	Java
Uso de sistemas operativos	Linux Windows UNIX

ANEXO B
ANÁLISIS

CONTENIDO

1. Introducción

El anexo hace referencia al Análisis, éste se define mediante las tecnologías de modelado que ofrece UML, bajo los lineamientos del Proceso Unificado.

Para los procesos y procedimientos de la empresa, el análisis del problema involucra capturar el máximo de información referente a este, obtener la visión del mismo por parte de los involucrados y generar un modelo para cada una de estas visiones. Se deben refinar estos modelos hasta obtener una o varias representaciones (modelos) del problema, los que posibilitan su análisis.

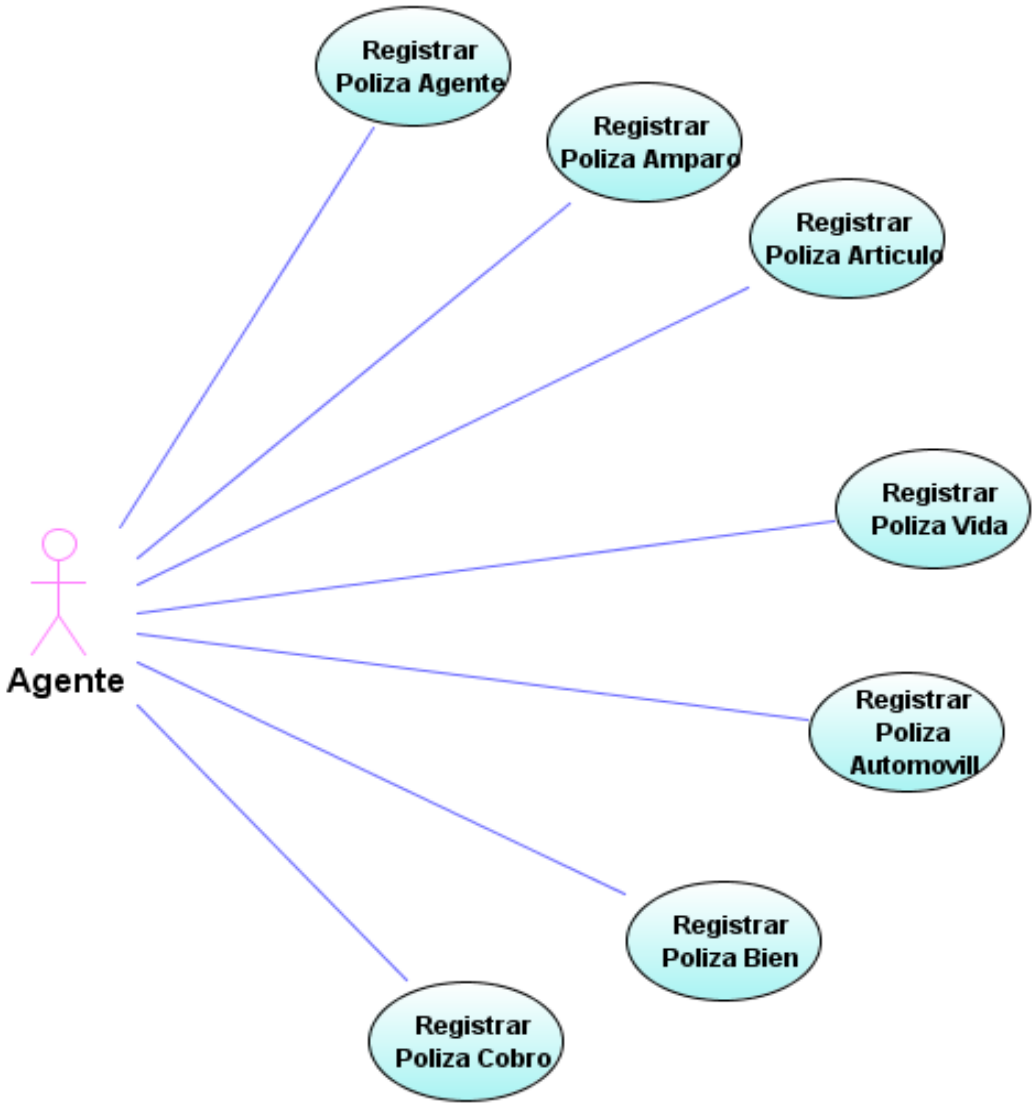
Estos modelos que describen las visiones del problema son los que posibilitarán que se vislumbre alguna solución, tal y como se describe, el componente estático del sistema de software lo constituye la definición de todas sus estructuras, las cuales podrán tomar distintos valores. Por otro lado las estructuras dinámicas de un software lo constituye su comportamiento, es decir, la definición de respuestas (cambios de estado) a ciertos estímulos. El componente funcional es aquel que define las transformaciones de las entradas, objetos del repositorio, eventos, entre otros., en respuestas o salidas del sistema software.

2. Modelo de Análisis

2.1. Diagrama de Casos de Uso

El diseño de los casos de uso están basados en dos aspectos: el primero hace referencia a su descripción gráfica con UML, el cual está basado en actores y los casos de uso, especificando su relación [14].

Figura 1 Caso de Uso Pólizas



Casos de uso: Pólizas para Agentes

Tabla 1: Casos de uso: Pólizas para Agentes

Id caso de uso	**Caso-01**
Nombre	Pólizas para Agentes
Autor	Gerónimo Barreneche Renaud
Versión	Noviembre de 2007
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Registro de Pólizas
Descripción	El registro de pólizas para agentes es un documento comercial, el cual, es ingresado al sistema para su almacenamiento.
Precondición	Ingreso validado al usuario
Flujo Normal	1. Validar ingreso a. Ingresa y valida compañía b. Ingresa y valida rama 2. Registrar en la base de datos 3. Salir del sistema
Postcondiciones	Registro correcto de póliza
Excepciones	En caso de no tener la información completa

Casos de uso: Pólizas para Amparos

Tabla 2 Casos de uso: Pólizas para Amparos

Id caso de uso	**Caso-02**
Nombre	Pólizas para Amparos
Autor	Gerónimo Barreneche Renaud
Versión	Diciembre de 2007
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Registro de Pólizas
Descripción	Cuenta con elementos tales como: Deducciones, Compañías, Monedas y Ramas, dichos elementos son la base fundamental para el registros de las pólizas.
Precondición	Ingreso validado al usuario
Flujo Normal	1. Validar ingreso a. Ingresar datos Generales de pólizas 2. Enviar Datos 3. Navegar por el menú
Postcondiciones	Registro correcto de póliza
Excepciones	En caso de no tener la información completa

Casos de uso: Pólizas para Artículo

Tabla 3 Casos de uso: Pólizas para Artículo

Id caso de uso	**Caso-03**
Nombre	Pólizas para Artículo
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2008
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Registro de Pólizas
Descripción	Los seguros abarcan las pólizas para artículos
Precondición	Ingreso validado al usuario
Flujo Normal	1. Validar ingreso a. Ingresar Compañía b. Ingresar Tipo póliza c. Ingresar Rama d. Ingresar Moneda 2. Enviar Datos 3. Navegar por el menú
Postcondiciones	Registro correcto de póliza
Excepciones	En caso de no tener la información completa

Casos de uso: Pólizas para Automóvil

Tabla 4 Casos de uso: Pólizas para Automóvil

Id caso de uso	**Caso-04**
Nombre	Pólizas para Automóvil
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2008
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Registro de Pólizas
Descripción	La compañía, cuenta con productos para el aseguramiento de vehículos.
Precondición	Ingreso validado al usuario
Flujo Normal	1. Validar ingreso a. Ingresar Compañía b. Ingresar Fecha de Ingreso c. Ingresar Cobertura d. Ingresar Cliente e. Ingresar Tipo Auto 2. Enviar Datos 3. Navegar por el menú
Postcondiciones	Registro correcto de póliza
Excepciones	En caso de no tener la información completa

Casos de uso: Pólizas para Bienes

Tabla 5 Casos de uso: Pólizas para Bienes

Id caso de uso	**Caso-05**
Nombre	Pólizas para Bienes
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2008
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Registro de Pólizas
Descripción	La compañía, cuenta con productos para el aseguramiento de bienes.
Precondición	Ingreso validado al usuario
Flujo Normal	1. Validar ingreso a. Ingresar Compañía b. Ingresar Fecha de Ingreso c. Ingresar Cobertura

	d. Ingresar Cliente e. Ingresar Tipo Auto 2. Enviar Datos 3. Navegar por el menú
Postcondiciones	Registro correcto de póliza
Excepciones	En caso de no tener la información completa

Caso de uso: Administración de información

Cada clase, que en su efecto, tiene una tabla en la base de datos, requiere gestionar la información; con el fin de ilustrar dicho procesimiento, se contruye un caso de uso que aplique a todas las clases.

Figura 2 Caso de uso: Administración

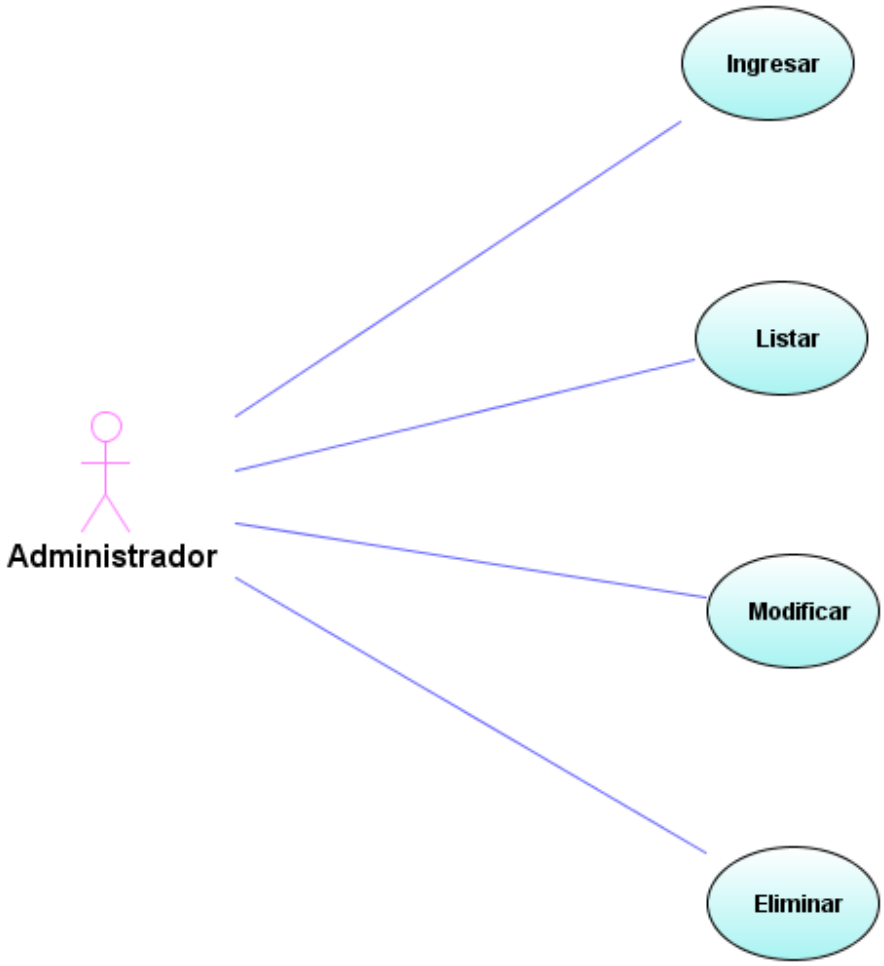


Tabla 6 Caso de uso: Administración

Id caso de uso	Caso-06
Nombre	Administración
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2009

Actores	Administrador
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Mantener la información del sistema de información
Descripción	El caso de uso, da contexto al mantenimiento de la información en sus aspectos básicos.
Precondición	Ser un usuario con privilegios
Flujo Normal	1. Ingresar al sistema 2. Seleccionar opción 2.1. Ingresar 2.2 Consultar 2.3 Modificar 2.4 Eliminar
Postcondiciones	Mantenibilidad de la información
Excepciones	Validación de la información

Caso de uso: control de tesorería y cartera

Figura 3 Caso de uso: control de tesorería y cartera

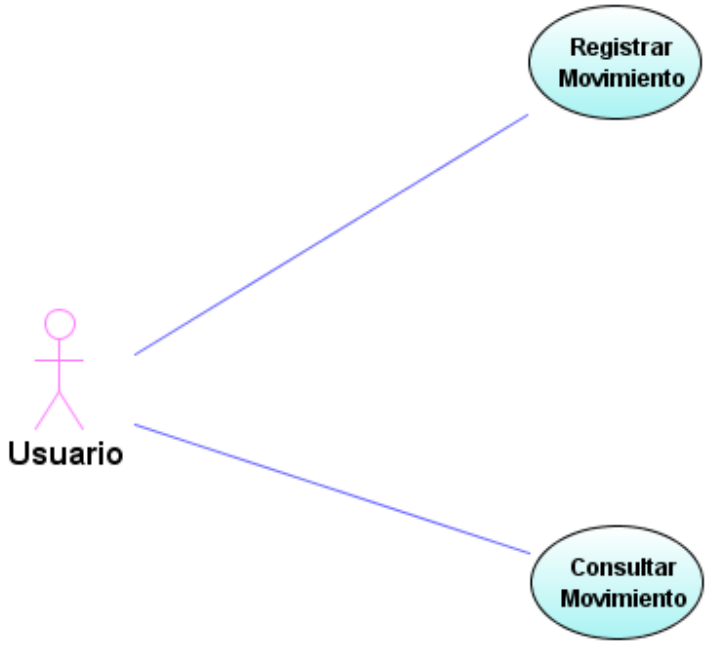


Tabla 7 Caso de uso: control de tesorería y cartera

Id caso de uso	**Caso-07**
Nombre	control de tesorería y cartera
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2009
Actores	Usuario
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Generar un control básico sobre los movimientos de cartera
Descripción	Mediante un formulario, controlar cada movimiento de importancia en la cartera básica de los clientes
Precondición	Ser usuario válido en el sistema
Flujo Normal	1. Ingresar al sistema 2. Ingresar movimientos 3 Consultar movimientos
Postcondiciones	Almacenar y controlar todos los movimientos básicos del cliente
Excepciones	Ninguna

Caso de uso: Gestión de clientes

Figura 4 Caso de uso: Gestión de clientes

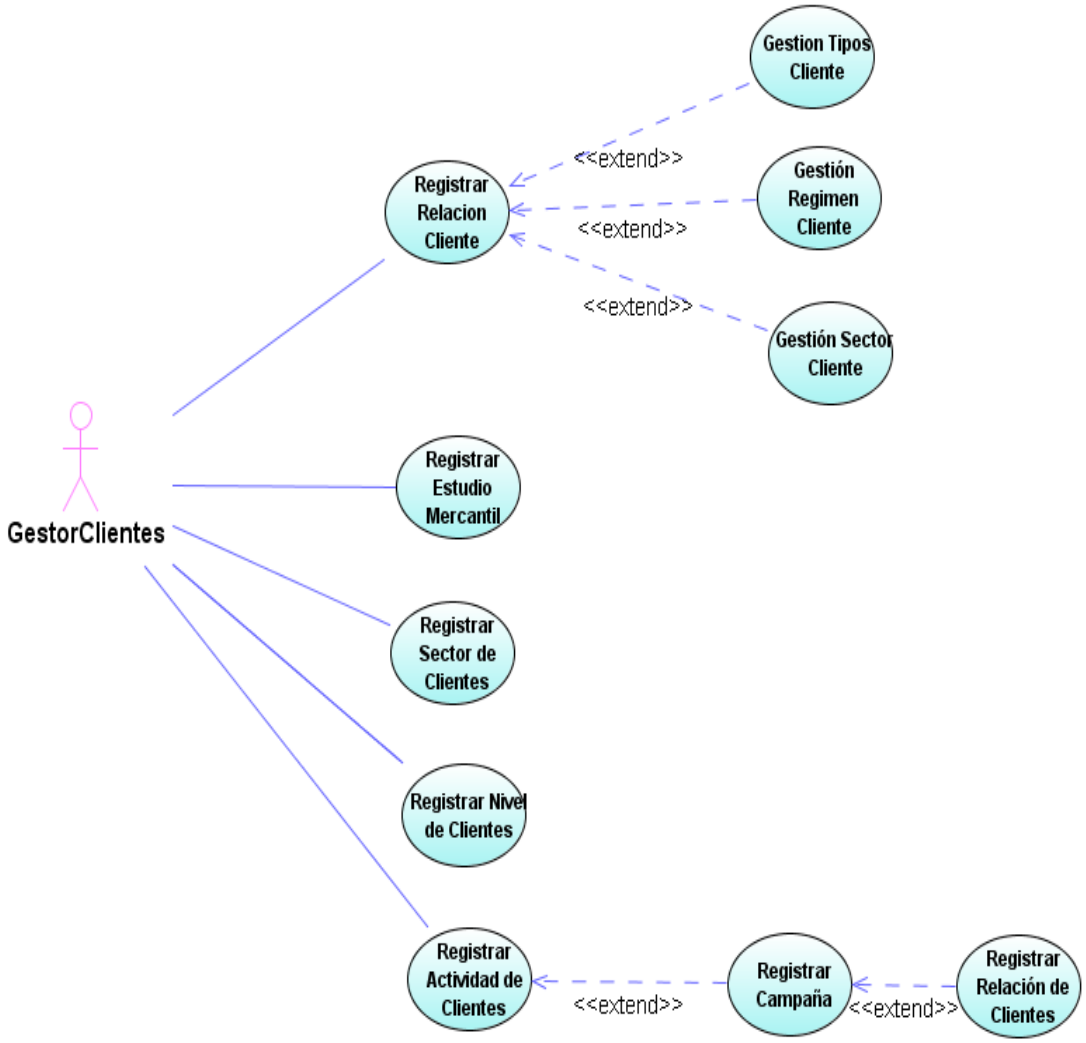


Tabla 8 Caso de uso: Gestión de clientes

Id caso de uso	**Caso-08**
Nombre	Gestión de clientes
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2009
Actores	Gestor Clientes
Fuente	Gilberto Arenas y CIA LTDA

Objetivo	Contexto general de los clientes
Descripción	Contexto general de los clientes
Precondición	Ser usuario del sistema de información
Flujo Normal	1. Ingresar al sistema 2. Gestionar a los sectores 3. Gestionar a los Estudios mercantiles
Postcondiciones	Gestión de información
Excepciones	Ninguna

Caso de uso: cotizaciones en línea

Figura 5 Caso de uso: cotizaciones en línea

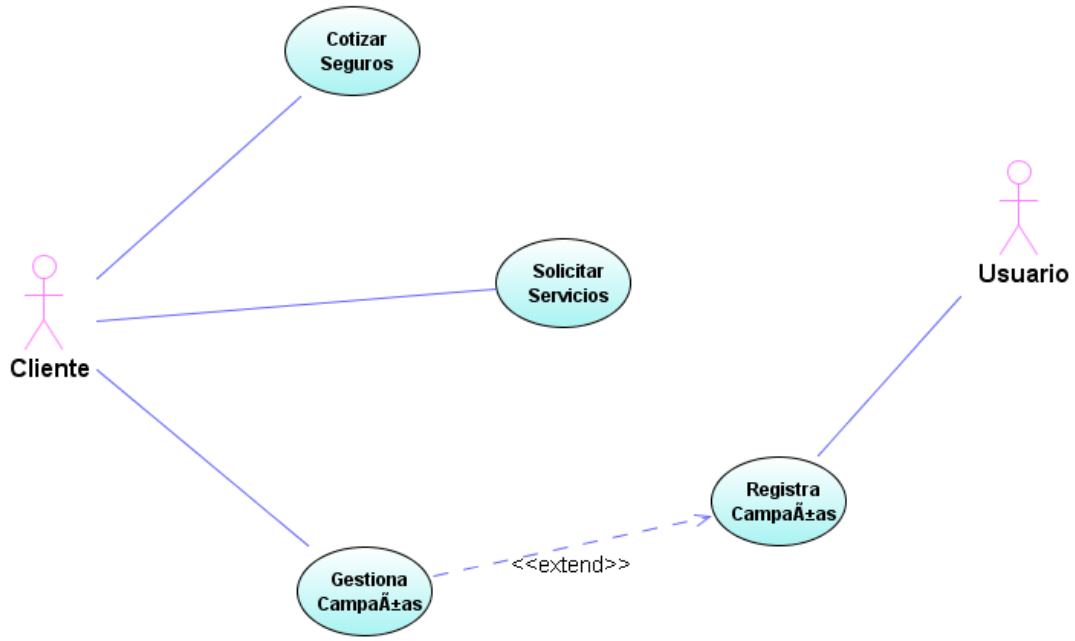


Tabla 9 Caso de uso: cotizaciones en línea

Id caso de uso	**Caso-09**
Nombre	Clientes
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2009
Actores	Gestor Clientes
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Contexto general de los clientes
Descripción	Contexto general de los clientes
Precondición	Ser usuario del sistema de información
Flujo Normal	1. Ingresar al sistema 1.1. Ingresar información básica de cotización 1.2. Guardar la cotización 3. Cerrar el cliente
Postcondiciones	Gestión de información
Excepciones	Ninguna

Caso de uso: Integración

Figura 6 Caso de uso: Integración

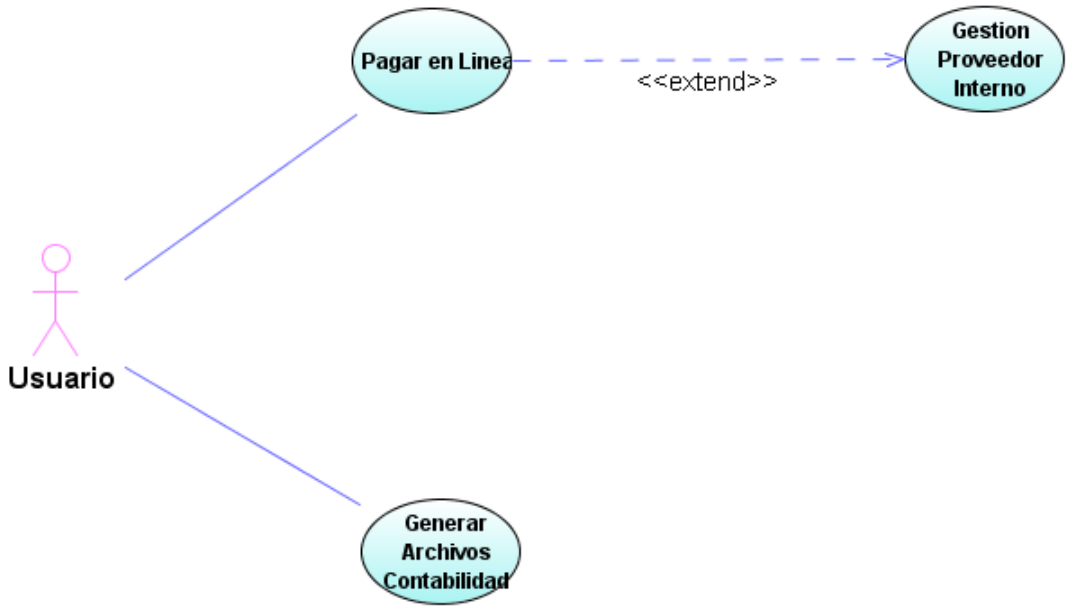


Tabla 10 Caso de uso: Integración

Id caso de uso	**Caso-10**
Nombre	Clientes
Autor	Gerónimo Barreneche Renaud
Versión	Enero de 2009
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Integrar con el sistema contable
Descripción	Integración mediante una interfaz
Precondición	Tener la información verificada en el sistema
Flujo Normal	1. Mostrar interfaz 2. Seleccionar información 3. Generar archivo plano
Postcondiciones	Software contable
Excepciones	Error en la generación del archivo

Caso de uso: la hoja de ruta (control de negocios)

Figura 7 Caso de uso: la hoja de ruta (control de negocios)

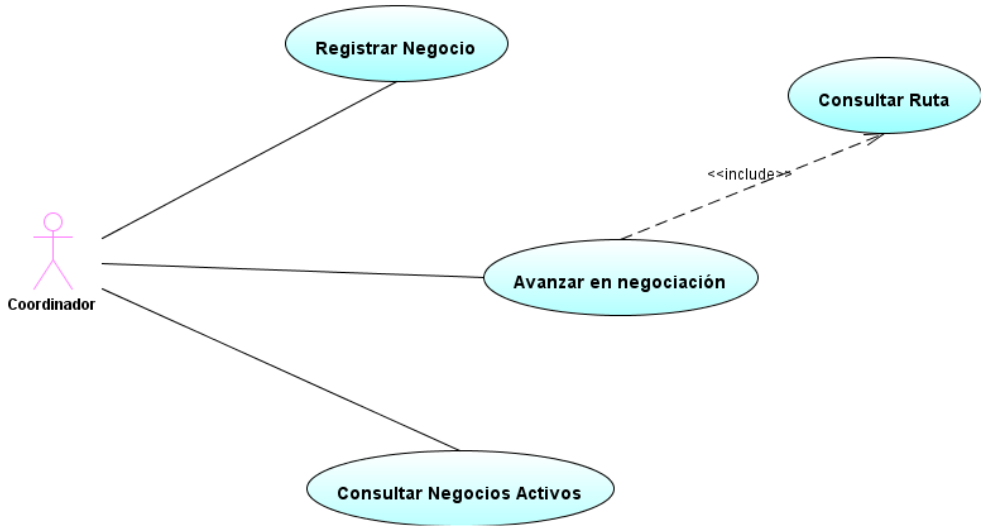


Tabla 11 Caso de uso: la hoja de ruta (control de negocios)

Id caso de uso	**Caso-11**
Nombre	la hoja de ruta (control de negocios)
Autor	Gerónimo Barreneche Renaud
Versión	Noviembre de 2008
Actores	Coordinador
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Controlar las rutas en los negocios
Descripción	Registro de negocios para su seguimiento
Precondición	Tener la información verificada en el sistema
Flujo Normal	1. Registrar Negocio 1.1. Seleccionar Cliente 1.2. Seleciconar Asesor 2. Registrar Negocio
Postcondiciones	Clientes y Asesores
Excepciones	

Caso de uso: pagos en línea

Figura 8 Caso de uso: pagos en línea

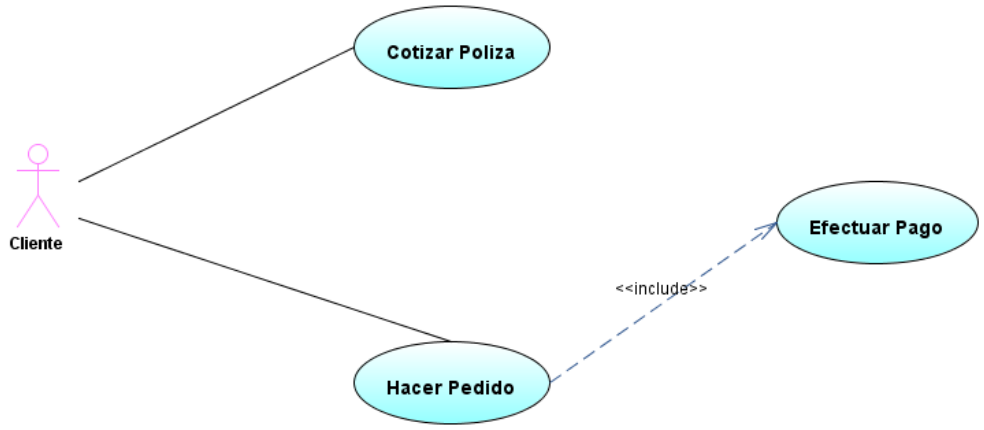


Tabla 12 Caso de uso: pagos en línea

Id caso de uso	**Caso-12**
Nombre	pagos en línea
Autor	Gerónimo Barreneche Renaud
Versión	Noviembre de 2008
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Controlar las rutas en los negocios
Descripción	Registro de negocios para su seguimiento
Precondición	Tener la información verificada en el sistema
Flujo Normal	1. Cotizar Póliza 1.1. Hacer pedido 1.2. Efectuar Pago 2. Verificar con el proveedor de pagos en línea 2.1 Generar archivos de comunicación 2.2 Hacer pruebas de negocios
Postcondiciones	Polizas correctamente registradas
Excepciones	Polizas incompletas

Caso de uso: gestión de la mensajería interna

Figura 9 Caso de uso: gestión de la mensajería interna

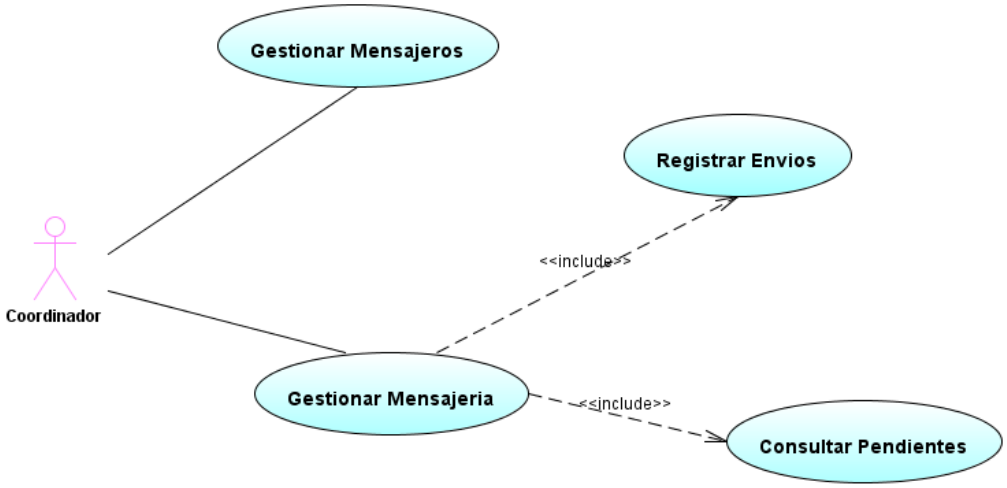


Tabla 13 Caso de uso: gestión de la mensajería interna

Id caso de uso	**Caso-13**
Nombre	gestión de la mensajería interna
Autor	Gerónimo Barreneche Renaud
Versión	Noviembre de 2008
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Controlar las rutas en los negocios
Descripción	Registro de negocios para su seguimiento
Precondición	Tener la información verificada en el sistema
Flujo Normal	1. Gestionar Mensajeros 1.1. Ingresar Mensajeros 1.2. Eliminar Mensajeros 1.3. Consultar Mensajeros 1.4. Modificar Mensajeros 2. Gestionar Mensajería 2.1 Registrar Envios 2.2 Consultar Pendientes
Postcondiciones	Mensajeros registrados en el sistema
Excepciones	

Caso de uso: perfiles y roles del sistema

Figura 9 Caso de uso: perfiles y roles del sistema

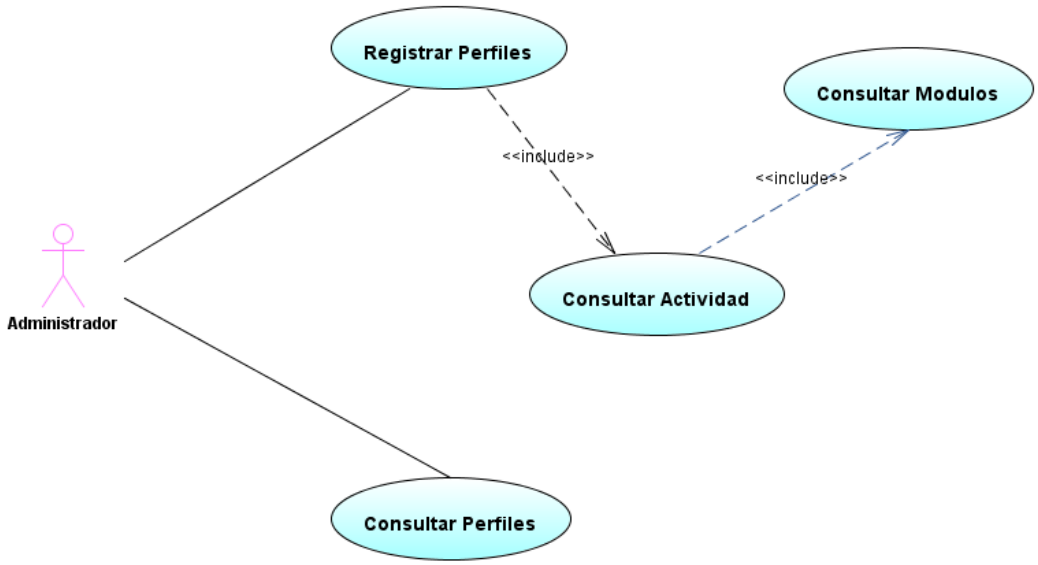


Tabla 14 Caso de uso: perfiles y roles del sistema

Id caso de uso	Caso-14
Nombre	perfiles y roles del sistema
Autor	Gerónimo Barreneche Renaud
Versión	Noviembre de 2008
Actores	Cliente
Fuente	Gilberto Arenas y CIA LTDA
Objetivo	Controlar las rutas en los negocios
Descripción	Registro de negocios para su seguimiento
Precondición	Tener la información verificada en el sistema
Flujo Normal	1. Consultar Perfiles 1.1. Buscar usuario 1.2. Leer configuración 1.3. Leer actividades 1.4. Mostrar Perfil 2. Registrar Perfil 2.1 Seleccionar Modulo 2.1.1 Leer actividades 2.2 Guardar configuración
Postcondiciones	Actividades, módulos y usuarios registrados en el sistema

Excepciones	

2.2. Diagrama de paquetes

El diagrama de paquetes organiza la información de los requerimientos es unidades, en éste caso se derivan los siguientes paquetes.

JAVA:
En éste paquete se incluyen todos los aspectos generales de la herramienta, los cuales son tomados en cuenta como la base fundamental para la plataforma.

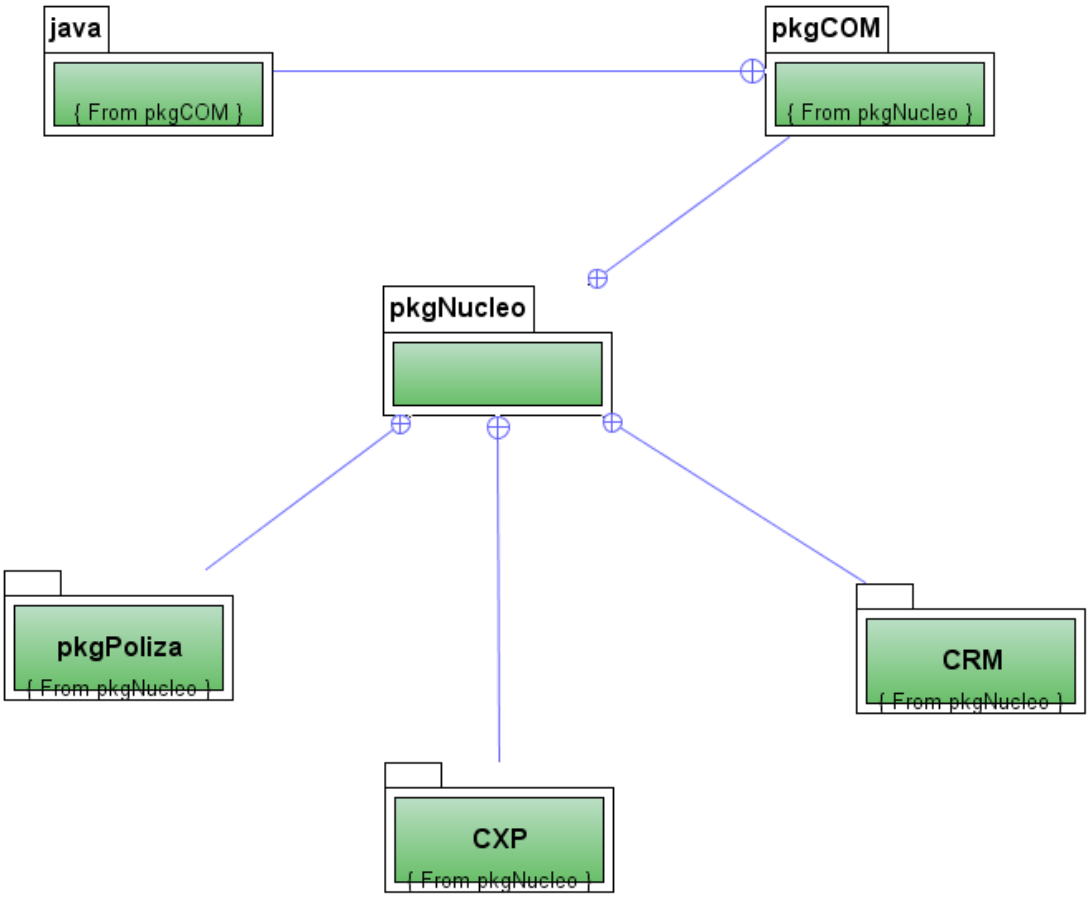
pkgCOM:
En éste elementos, se tienen todas las clases usadas en el sistema, los cuales permiten la comunicación con la base de datos, éste paquete es una herramienta de interfaz.

pkgNucleo:
Las clases que tienen que ver son la administración y la gestión de la información están enmarcadas en éste paquete, el cual, organiza la seguridad de la información.

pkgPoliza:
Dicho paquete alberga las clases referentes a las pólizas, las cuales están debidamente enmarcadas en los requisitos.

pkgCRM:
Paquete para la gestión del cliente, en éste elemento se incluyen toda las clases para cumplir con algunos de los objetivos planetados.

pkgCXP:
Paquete para la gestión de la cartera.



2.3. Diagrama de Clases

Diagrama de clases: perfiles y roles del sistema

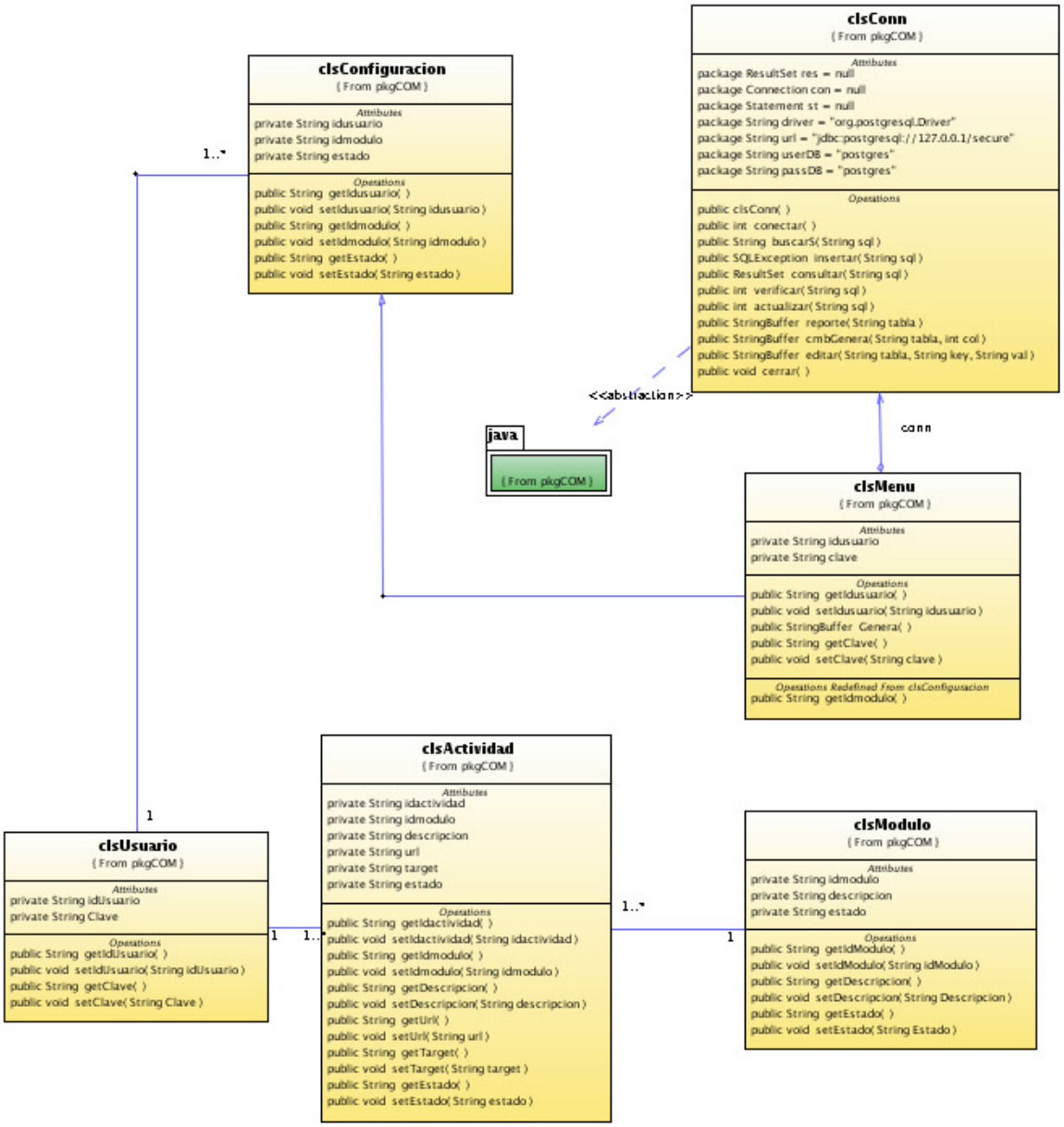


Diagrama de clases: Gestión de clientes (PARTE A)

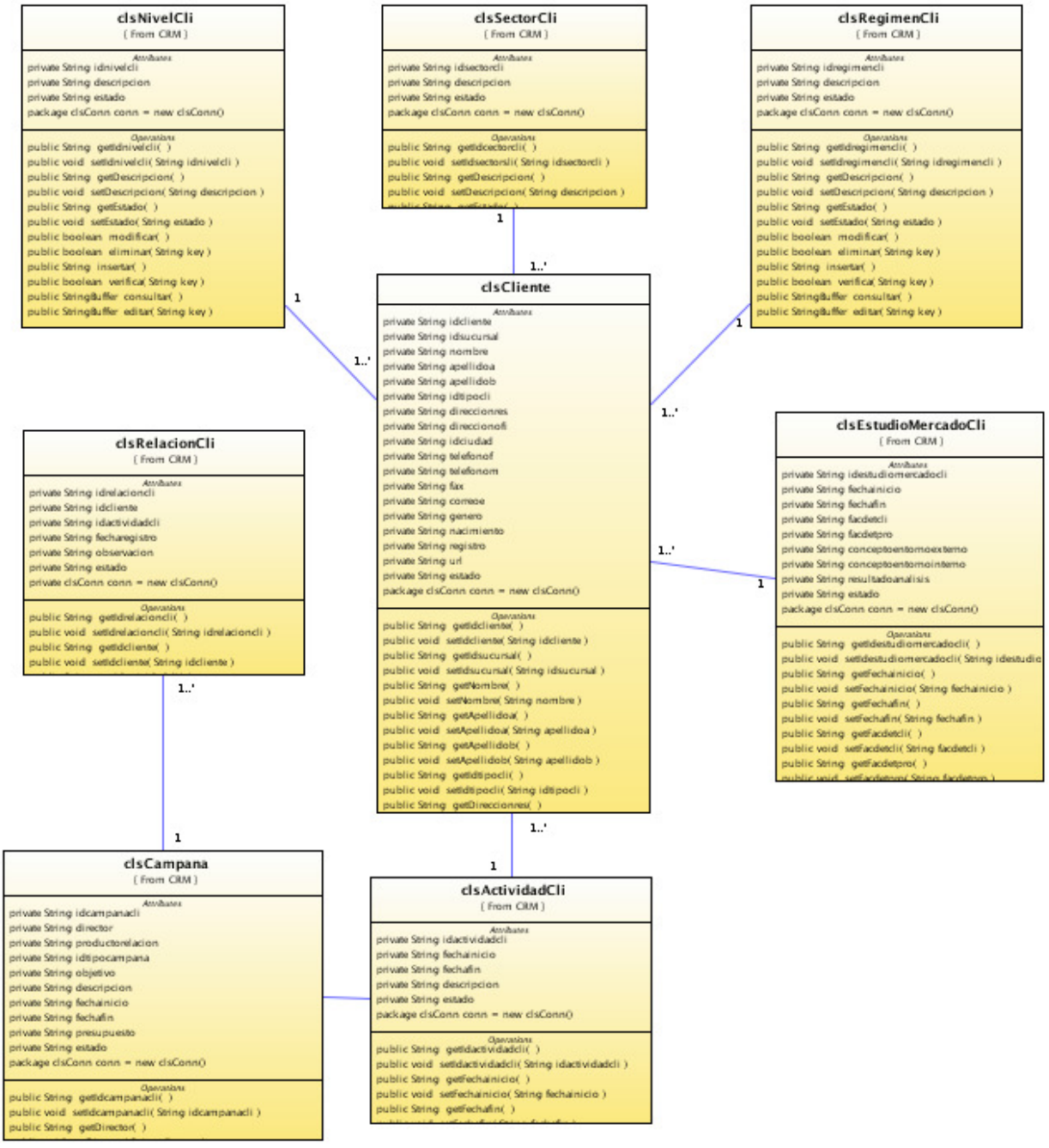


Diagrama de clases: Gestión de clientes (PARTE B)

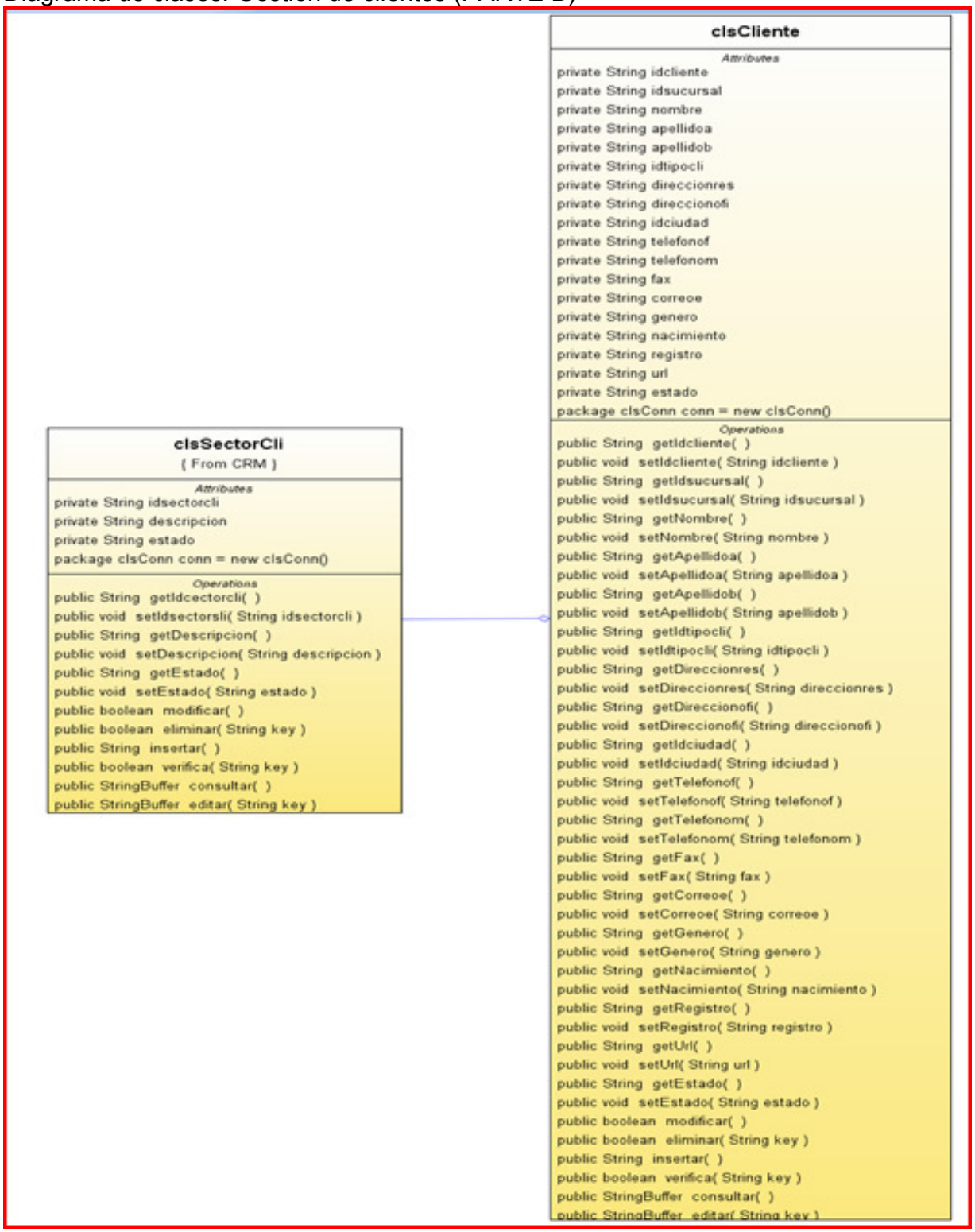


Diagrama de clases: Gestión de clientes (PARTE C)

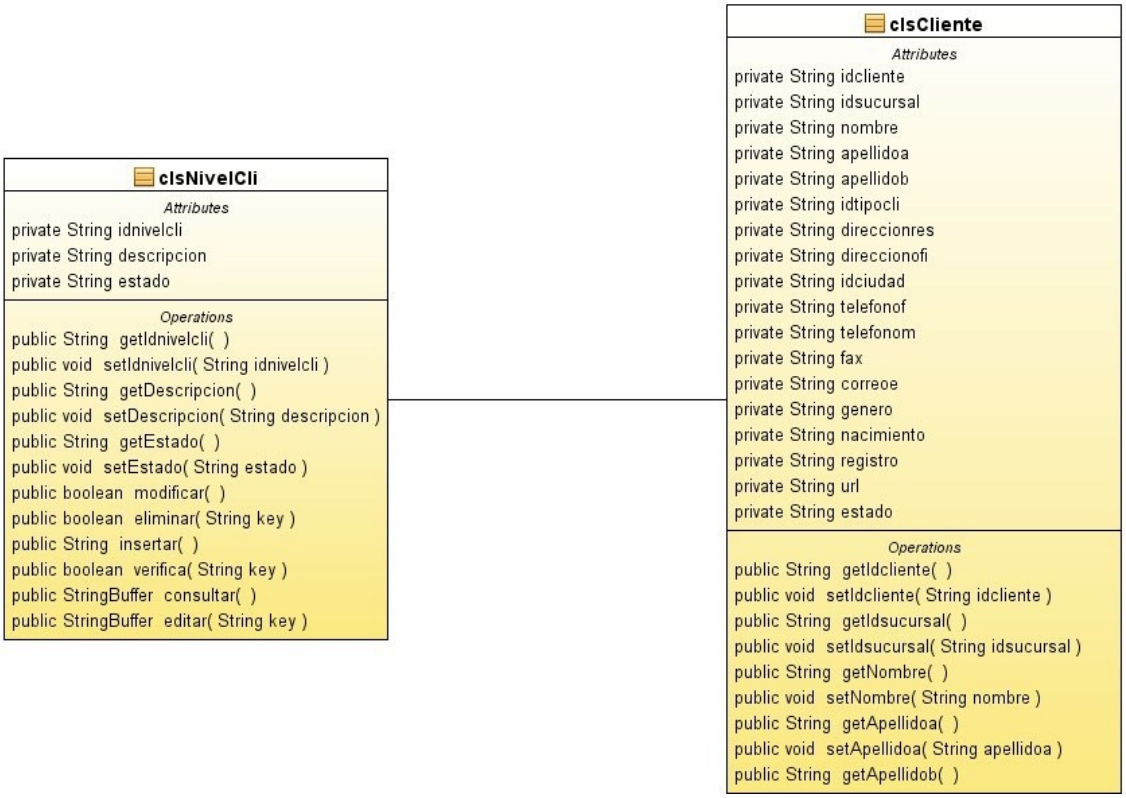


Diagrama de clases: Gestión de clientes (PARTE D)

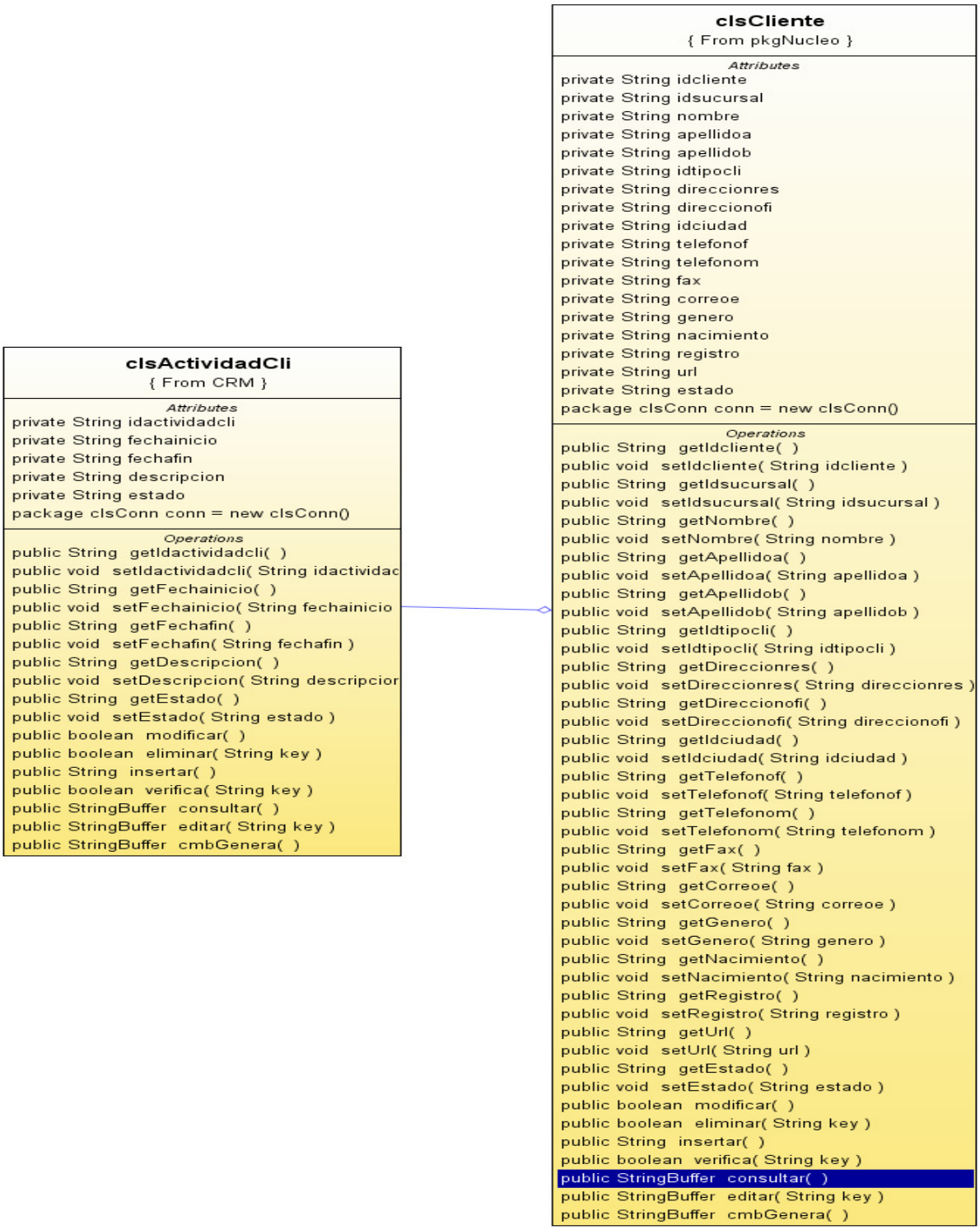


Diagrama de clases: Gestión de clientes (PARTE E)

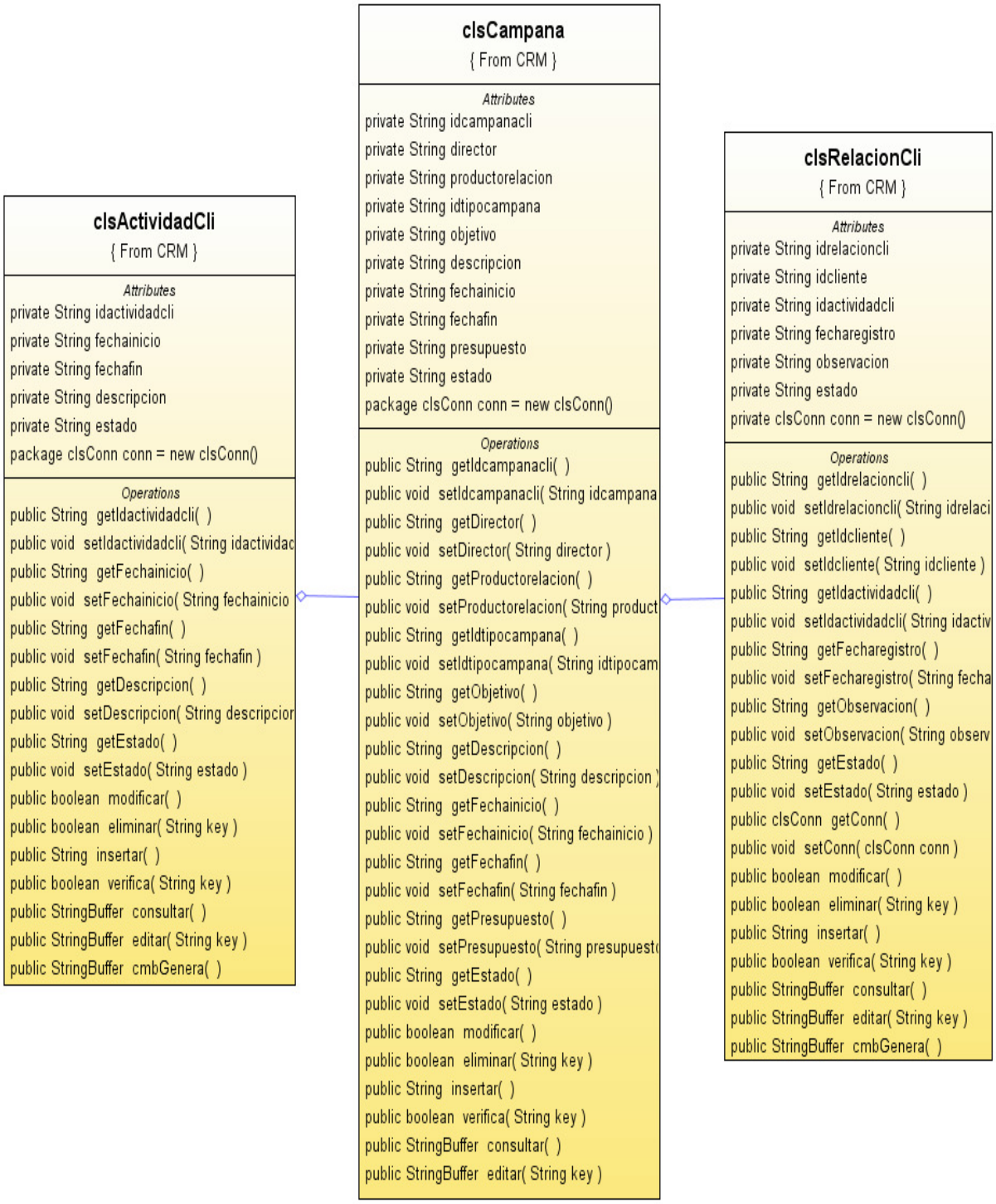


Diagrama de clases: Gestión de clientes (PARTE F)

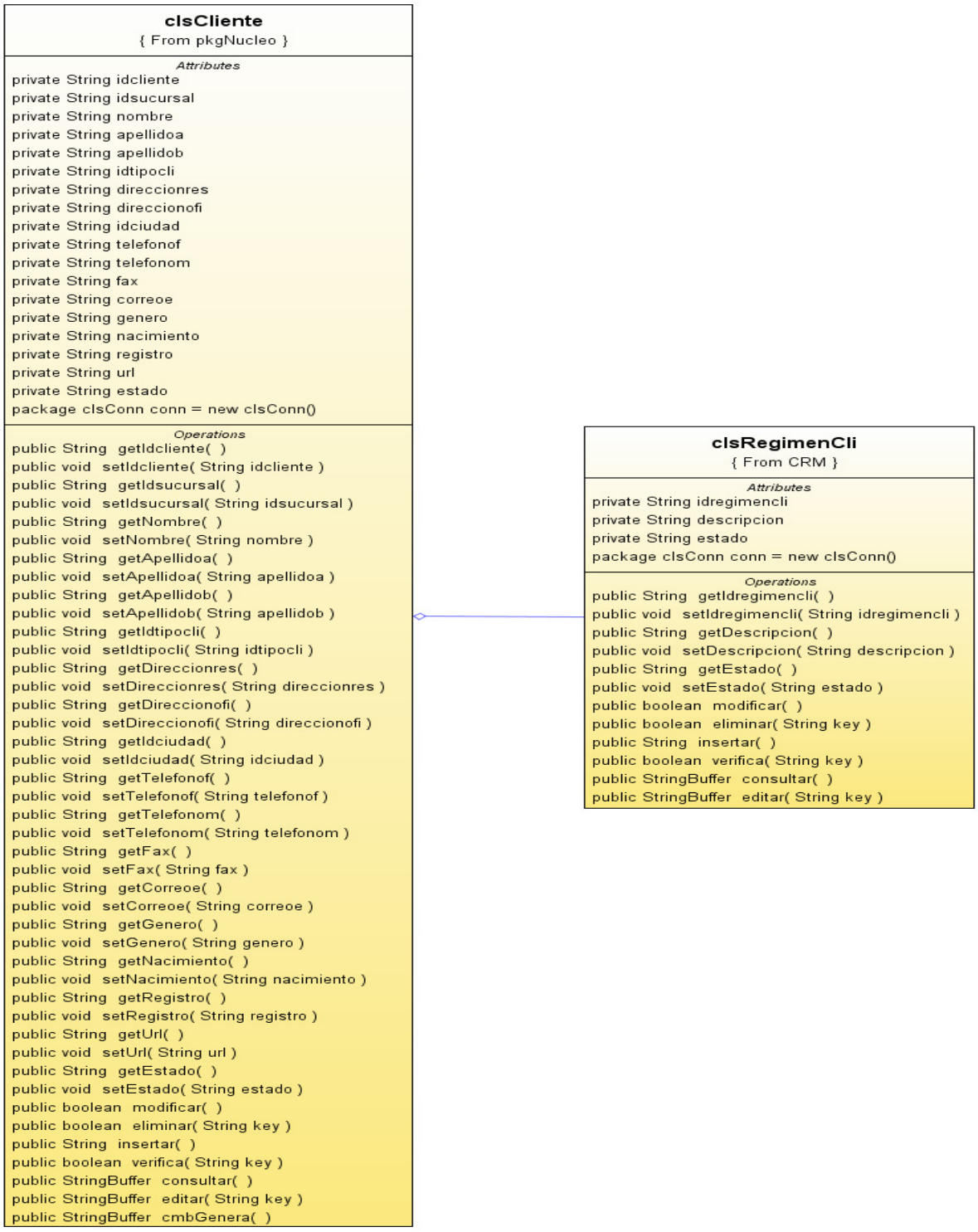


Diagrama de clases: Gestión de clientes (PARTE G)

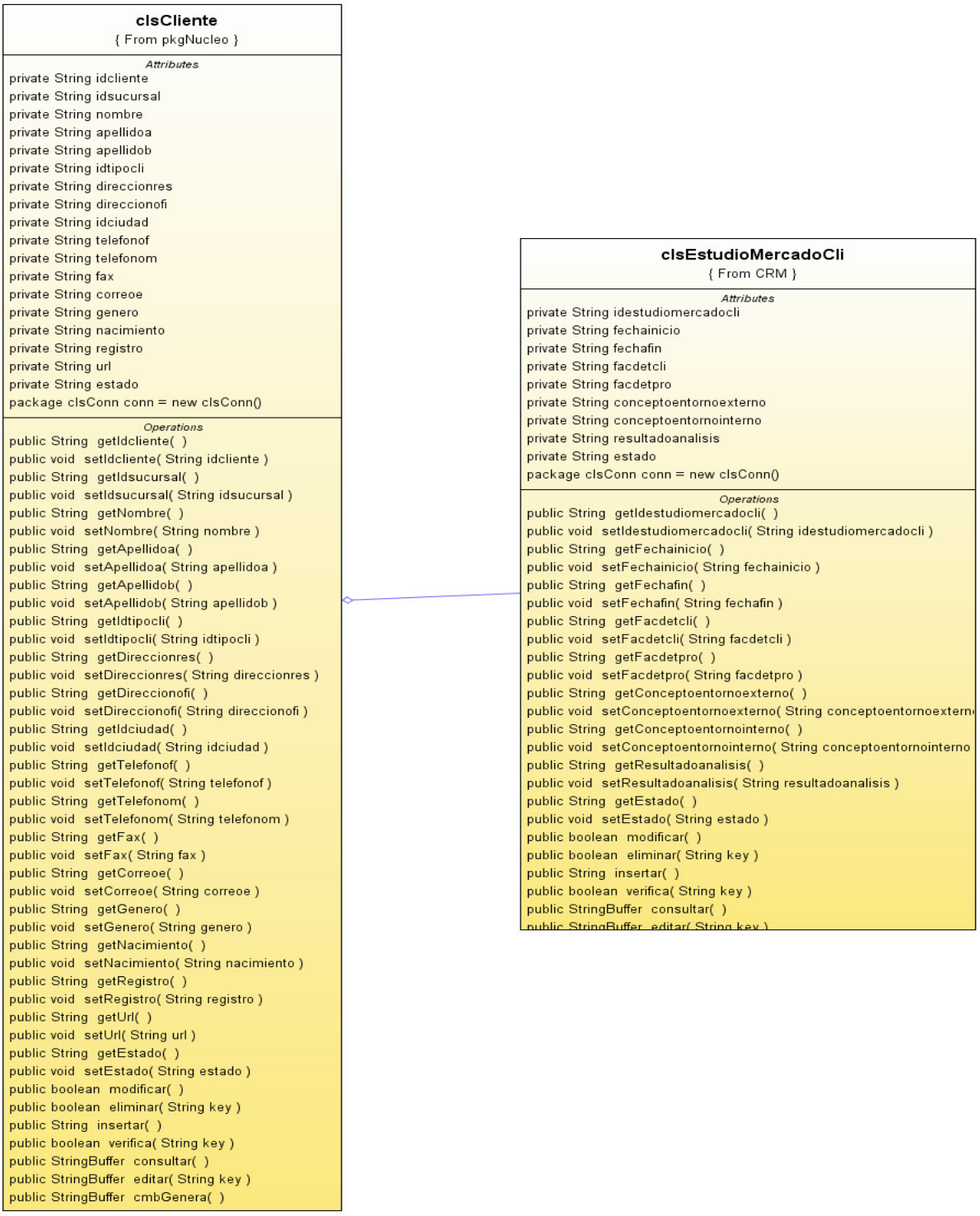


Diagrama de clases: Pólizas Cliente (Parte A)

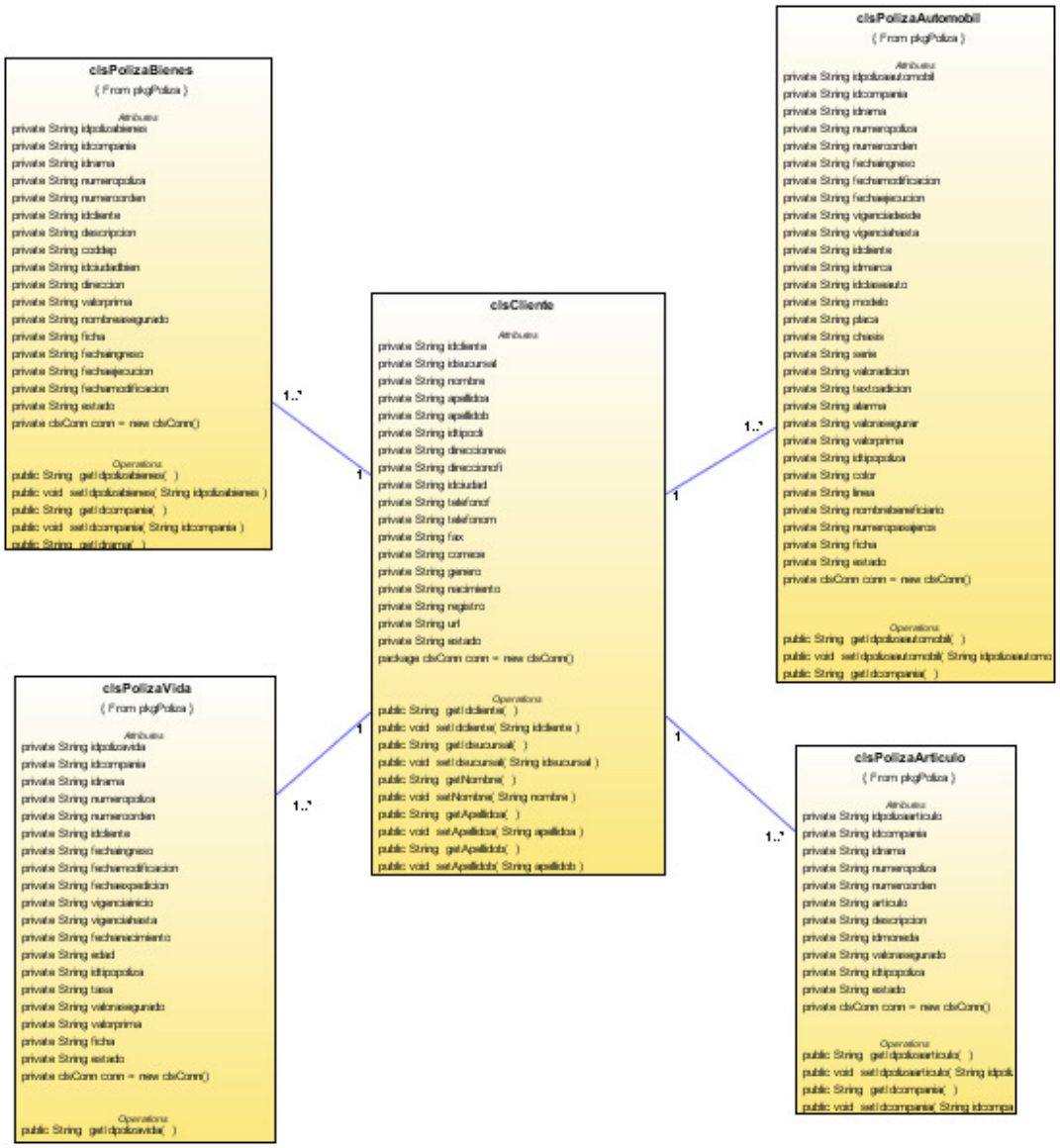


Diagrama de clases: Pólizas Cliente (Parte B)

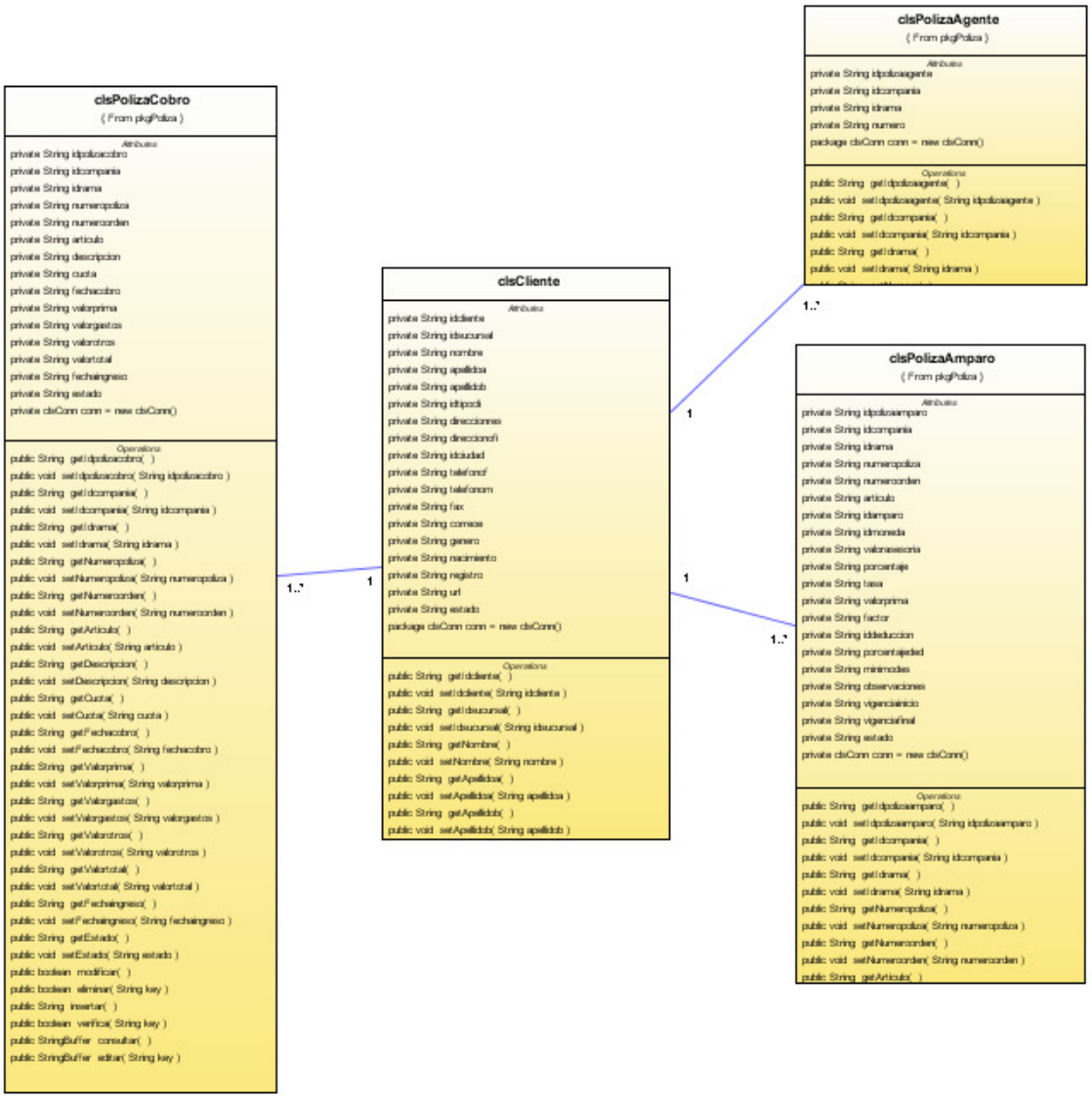


Diagrama de clases: hoja de ruta (control de negocios)

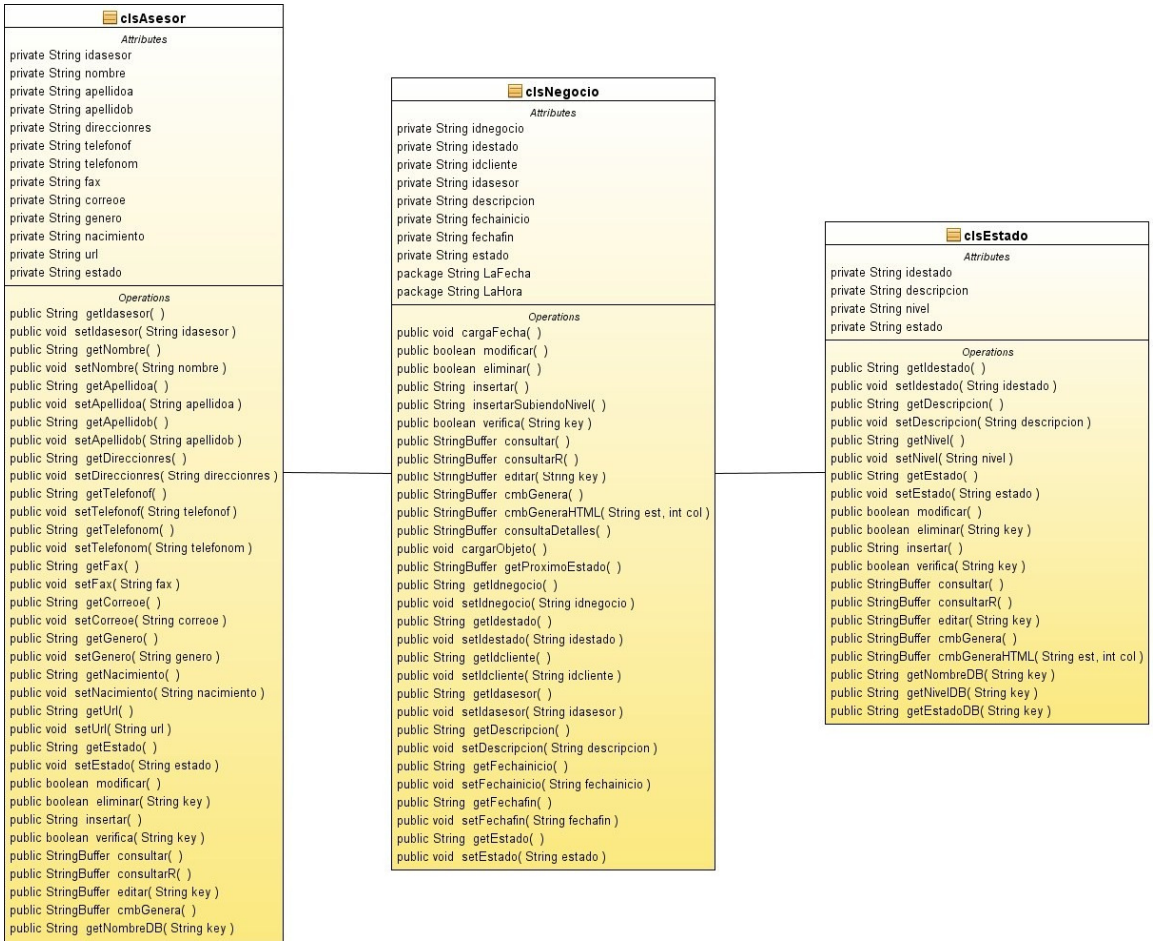


Diagrama de clases: hoja de ruta (control de negocios)

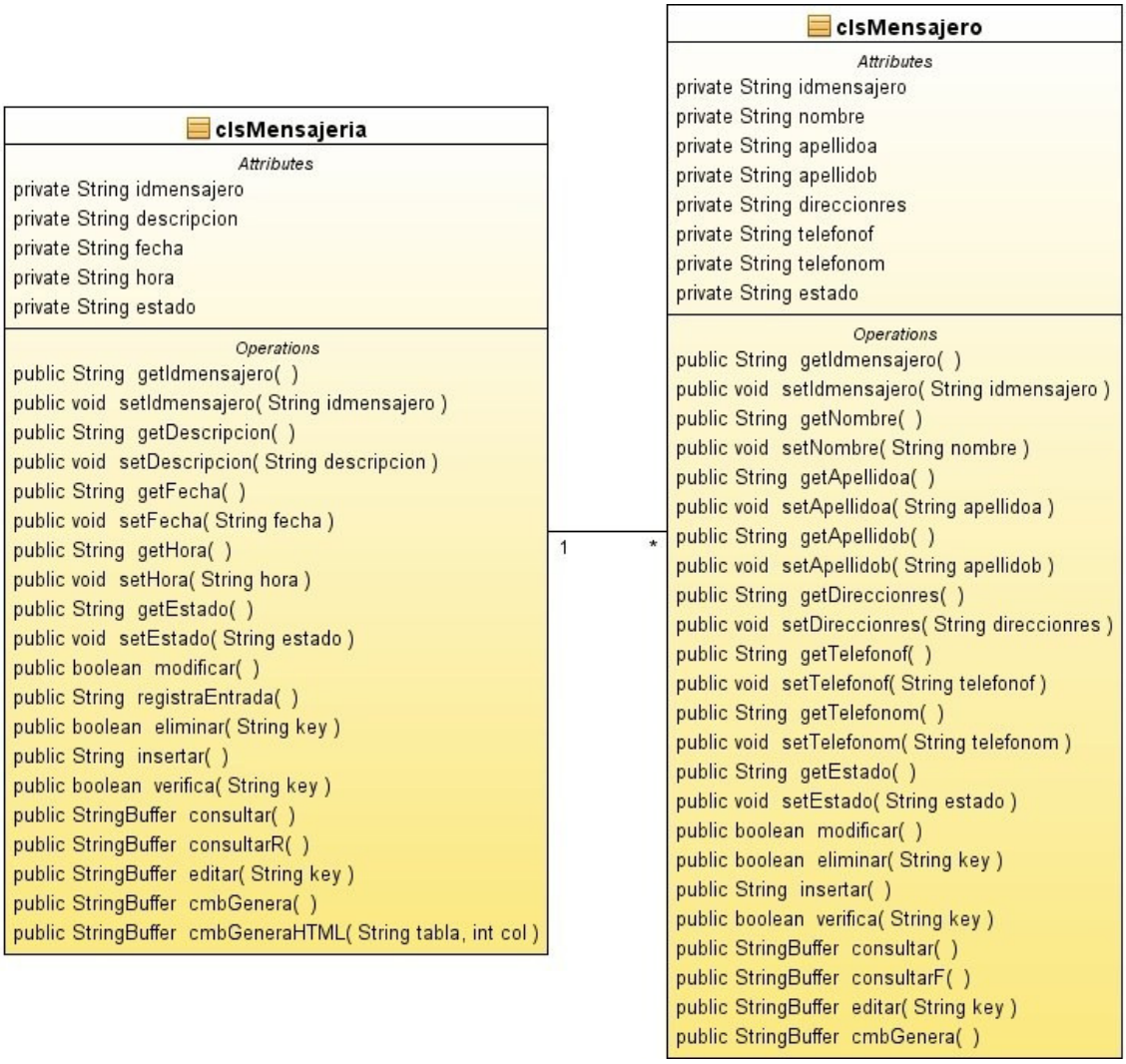


Diagrama de clases: control de tesorería y cartera

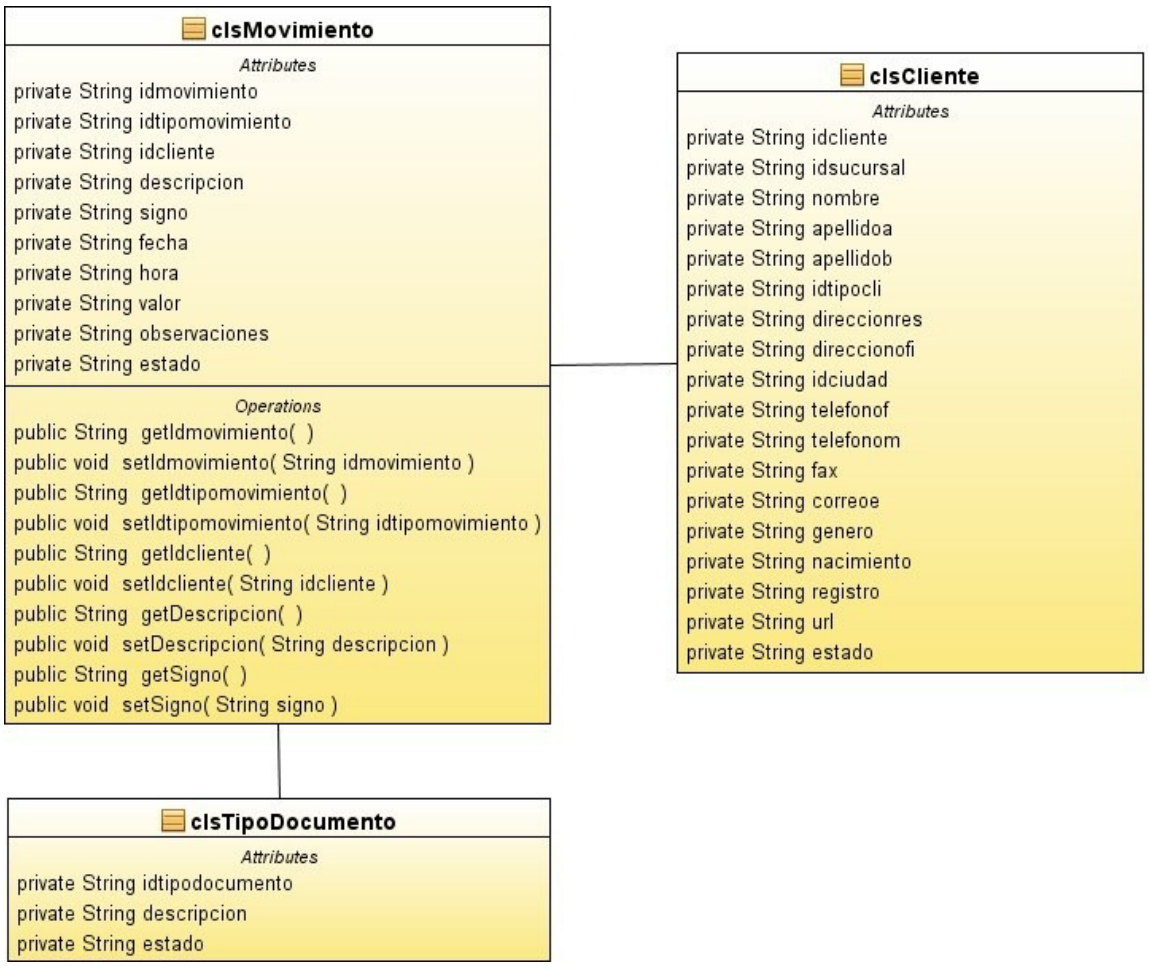
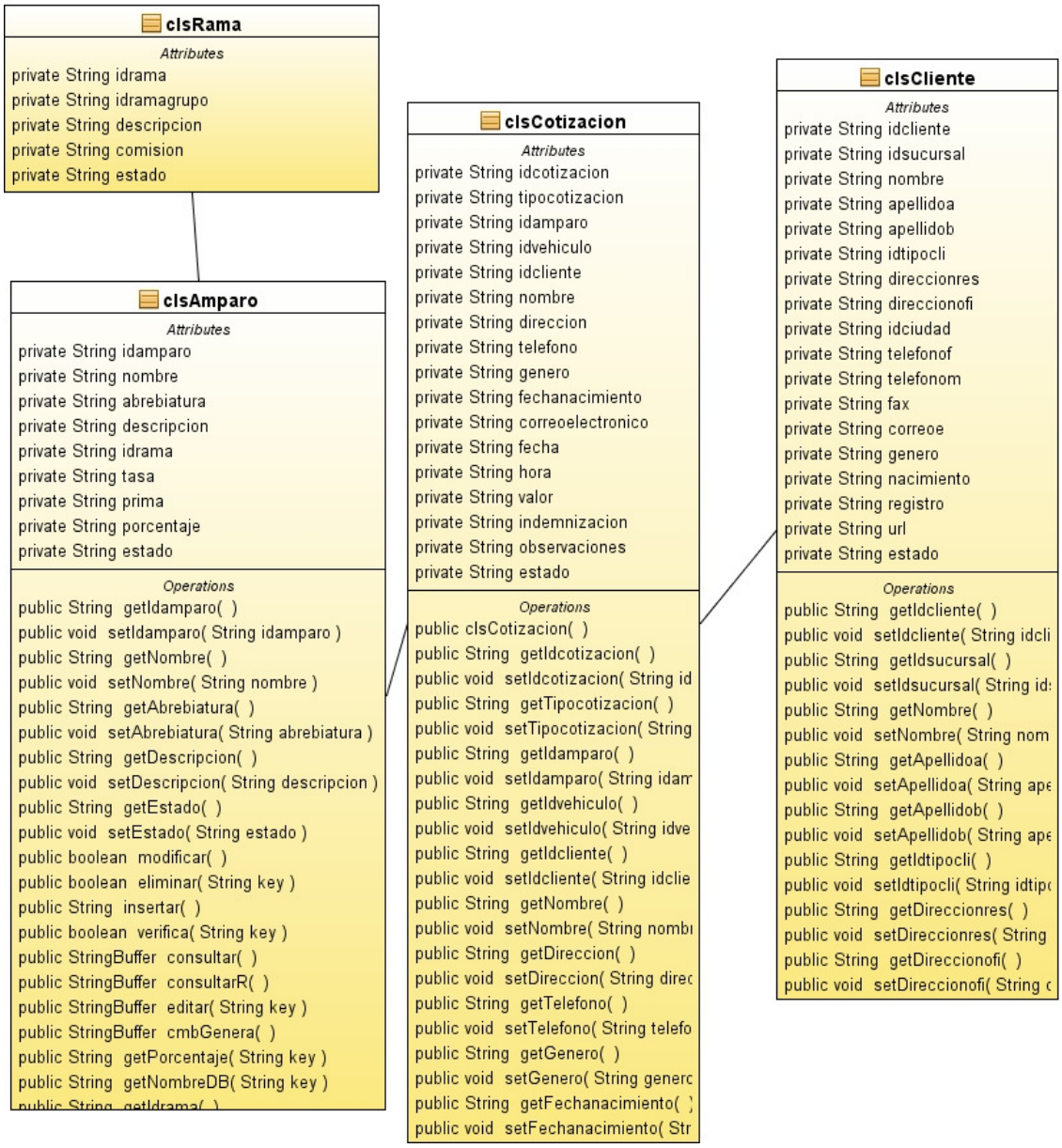


Diagrama de clases: control de tesorería y cartera



2.4. Diccionario de Datos

Tabla: public_tblabogado

Columnas

Nombre	Tipo	Tamaño
idabogado	Texto	10
nombre	Texto	255
apellidoa	Texto	255
apellidob	Texto	255
tarjeta	Texto	45
direccionres	Texto	255
direccionofi	Texto	255
idciudad	Texto	10
telefonof	Texto	56
telefonom	Texto	56
fax	Texto	56
estado	Texto	1

Tabla: public_tblactividad

Columnas

Nombre	Tipo	Tamaño
idactividad	Texto	10
idmodulo	Texto	10
descripcion	Texto	255
url	Texto	60
target	Texto	40
estado	Texto	1

Tabla: public_tblactividadcli

Columnas

Nombre	Tipo	Tamaño
idactividadcli	Texto	10
fechainicio	Texto	12
fechafin	Texto	12
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblactividadcom

Columnas

Nombre	Tipo	Tamaño
idactividadcom	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblafiliado

Columnas

Nombre	Tipo	Tamaño
idafiliado	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblagente

Columnas

Nombre	Tipo	Tamaño
idagente	Texto	10
idtipoagente	Texto	10
nombre	Texto	255
apellidoa	Texto	255

apellidob	Texto	255
estado	Texto	1

Tabla: public_tblajustador

Columnas

Nombre	Tipo	Tamaño
idajustador	Texto	10
nombre	Texto	255
apellidoa	Texto	255
apellidob	Texto	255
tarjeta	Texto	60
direccionres	Texto	255
direccionofi	Texto	255
idciudad	Texto	10
telefonof	Texto	16
telefonom	Texto	16
fax	Texto	40
estado	Texto	1

Tabla: public_tblamparo

Columnas

Nombre	Tipo	Tamaño
idamparo	Texto	10
nombre	Texto	255
abrebiatura	Texto	6
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblbanco

Columnas

Nombre	Tipo	Tamaño
idbanco	Texto	10

nit	Texto	16
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblcampanacli

Columnas

Nombre	Tipo	Tamaño
idcampanacli	Texto	10
director	Texto	255
productorelacion	Texto	255
idtipocampana	Texto	10
objetivo	Texto	255
descripcion	Texto	255
fechainicio	Texto	8
fechafin	Texto	8
presupuesto	Texto	255
estado	Texto	1

Tabla: public_tblciudad

Columnas

Nombre	Tipo	Tamaño
idciudad	Texto	10
iddepartamento	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblclaseauto

Columnas

Nombre	Tipo	Tamaño
idclaseauto	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblclausula

Columnas

Nombre	Tipo	Tamaño
idclausula	Texto	10
nombre	Texto	255
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblcliente

Columnas

Nombre	Tipo	Tamaño
idcliente	Texto	10
idsucursal	Texto	10
nombre	Texto	255
apellidoa	Texto	255
apellidob	Texto	255
idtipocli	Texto	10
direccionres	Texto	255
direccionofi	Texto	255
idciudad	Texto	10
telefonof	Texto	12
telefonom	Texto	12
fax	Texto	12
correoe	Texto	70
genero	Texto	1
nacimiento	Texto	12
registro	Texto	2
url	Texto	20
estado	Texto	1

Tabla: public_tblcompania

Columnas

Nombre	Tipo	Tamaño
idcompania	Texto	10
razon	Texto	4
direccion	Texto	255
telefono	Texto	255
idciudad	Texto	10
correo	Texto	40
web	Texto	40
estado	Texto	1

Tabla: public_tblconceptoafe

Columnas

Nombre	Tipo	Tamaño
idconceptoafe	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblconfiguracion

Columnas

Nombre	Tipo	Tamaño
idusuario	Texto	10
idmodulo	Texto	255
estado	Texto	1

Tabla: public_tbldeduccion

Columnas

Nombre	Tipo	Tamaño
iddeduccion	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tbldepartamento

Columnas

Nombre	Tipo	Tamaño
iddepartamento	Texto	10
idpais	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblempresa

Columnas

Nombre	Tipo	Tamaño
idempresa	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblestudiomercadocli

Columnas

Nombre	Tipo	Tamaño
idestudiomercadocli	Texto	10
fechainicio	Texto	12
fechafin	Texto	12
facdetcli	Texto	12
facdetpro	Texto	12
conceptoentornoexterno	Texto	255
conceptoentornointerno	Texto	255
resultadoanalisis	Texto	255
estado	Texto	1

Tabla: public_tblfamiliar

Columnas

Nombre	Tipo	Tamaño
idfamiliar	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblformapago

Columnas

Nombre	Tipo	Tamaño
idformapago	Texto	10
descripcion	Texto	255
numerocuenta	Texto	255
estado	Texto	1

Tabla: public_tblgusuario

Columnas

Nombre	Tipo	Tamaño
idusuario	Texto	10
idgusuario	Texto	10
estado	Texto	1

Tabla: public_tblmarca

Columnas

Nombre	Tipo	Tamaño
idmarca	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblmensajecli

Columnas

Nombre	Tipo	Tamaño

idmensajecli	Texto	10
idcliente	Texto	10
asunto	Texto	50
textomensaje	Texto	255
fechamensaje	Texto	255
estado	Texto	1

Tabla: public_tblmensajero

Columnas

Nombre	Tipo	Tamaño
idmensajero	Texto	10
nombre	Texto	255
apellidoa	Texto	255
apellidob	Texto	255
direccionres	Texto	255
telefonof	Texto	12
telefonom	Texto	56
estado	Texto	1

Tabla: public_tblmodulo

Columnas

Nombre	Tipo	Tamaño
idmodulo	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblmoneda

Columnas

Nombre	Tipo	Tamaño
idmoneda	Texto	10
descripcion	Texto	255

estado	Texto	1

Tabla: public_tblmovimiento

Columnas

Nombre	Tipo	Tamaño
idmovimiento	Texto	10
idtipomovimiento	Texto	10
idcliente	Texto	10
descripcion	Texto	255
signo	Texto	255
fecha	Texto	12
hora	Texto	8
valor	Entero largo	4
observaciones	Texto	255
estado	Texto	1

Tabla: public_tblnivelcli

Columnas

Nombre	Tipo	Tamaño
idnivelcli	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblpais

Columnas

Nombre	Tipo	Tamaño
idpais	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblpolizaagente

Columnas

Nombre	Tipo	Tamaño
idpolizaagente	Texto	10
idcompania	Texto	10
idrama	Texto	10
numero	Texto	255
estado	Texto	1

Tabla: public_tblpolizaamparo

Columnas

Nombre	Tipo	Tamaño
idpolizaamparo	Texto	10
idcompania	Texto	10
idrama	Texto	10
numeropoliza	Texto	255
numeroorden	Texto	255
articulo	Texto	255
idamparo	Texto	10
idmoneda	Texto	255
valorasesoria	Doble	8
porcentaje	Decimal	16
tasa	Decimal	16
valorprima	Doble	8
factor	Entero largo	4
iddeduccion	Texto	10
porcentajeded	Decimal	16
minimodes	Doble	8
observaciones	Texto	255
vigenciainicio	Texto	12
vigenciafinal	Texto	12
estado	Texto	1

Tabla: public_tblpolizaarticulo

Columnas

Nombre	Tipo	Tamaño
idpolizaarticulo	Texto	10
idcompania	Texto	10
idrama	Texto	10
numeropoliza	Texto	255
numeroorden	Texto	255
articulo	Entero largo	4
descripcion	Texto	255
idmoneda	Texto	255
valorasegurado	Doble	8
idtipopoliza	Texto	255
estado	Texto	1

Tabla: public_tblpolizaautomobil

Columnas

Nombre	Tipo	Tamaño
idpolizaautomobil	Texto	10
idcompania	Texto	10
idrama	Texto	10
numeropoliza	Texto	6
numeroorden	Texto	6
fechaingreso	Texto	12
fechamodificacion	Texto	12
fechaejecucion	Texto	12
vigenciadesde	Texto	255
vigenciahasta	Texto	255
idcliente	Texto	255
idmarca	Texto	10
idclaseauto	Texto	10
modelo	Texto	255
placa	Texto	255
chasis	Texto	255
serie	Texto	255

valoradicion	Entero largo	4
textoadicion	Texto	255
alarma	Texto	255
valorasegurar	Entero largo	4
valorprima	Entero largo	4
idtipopoliza	Texto	255
color	Texto	255
linea	Texto	255
nombrebeneficiario	Texto	255
numeropasajeros	Entero largo	4
ficha	Texto	255
estado	Texto	1

Tabla: public_tblpolizabienes

Columnas

Nombre	Tipo	Tamaño
idpolizabienes	Texto	10
idcompania	Texto	10
idrama	Texto	10
numeropoliza	Texto	6
numeroorden	Texto	6
idcliente	Texto	10
descripcion	Texto	255
coddep	Entero largo	4
idciudadbien	Texto	255
direccion	Texto	255
valorprima	Doble	8
nombreasegurado	Texto	255
ficha	Texto	255
fechaingreso	Texto	12
fechaejecucion	Texto	12
fechamodificacion	Texto	12
estado	Texto	1

Tabla: public_tblpolizacobro

Columnas

Nombre	Tipo	Tamaño
idpolizacobro	Texto	10
idcompania	Texto	10
idrama	Texto	10
numeropoliza	Texto	8
numeroorden	Texto	8
articulo	Entero largo	4
descripcion	Texto	255
cuota	Entero largo	4
fechacobro	Texto	255
valorprima	Entero largo	4
valorgastos	Entero largo	4
valorotros	Entero largo	4
valortotal	Entero largo	4
fechaingreso	Texto	255
estado	Texto	1

Tabla: public_tblpolizavida

Columnas

Nombre	Tipo	Tamaño
idpolizavida	Texto	10
idcompania	Texto	10
idrama	Texto	10
numeropoliza	Texto	8
numeroorden	Texto	8
idcliente	Texto	10
fechaingreso	Texto	12
fechamodificacion	Texto	12
fechaexpedicion	Texto	12
vigenciainicio	Texto	12
vigenciahasta	Texto	12

fechanacimiento	Texto	12
edad	Entero largo	4
idtipopoliza	Texto	10
tasa	Entero largo	4
valoraseguredo	Entero largo	4
valorprima	Entero largo	4
ficha	Texto	10
estado	Texto	1

Tabla: public_tblrama

Columnas

Nombre	Tipo	Tamaño
idrama	Texto	10
idramagrupo	Texto	10
descripcion	Texto	255
comision	Entero largo	4
estado	Texto	1

Tabla: public_tblramagrupo

Columnas

Nombre	Tipo	Tamaño
idramagrupo	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblregimencli

Columnas

Nombre	Tipo	Tamaño
idregimencli	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblrelacioncli

Columnas

Nombre	Tipo	Tamaño
idrelacioncli	Texto	10
idcliente	Texto	10
idactividadcli	Texto	10
fecharegistro	Texto	12
observacion	Texto	255
estado	Texto	1

Tabla: public_tblsectorcli

Columnas

Nombre	Tipo	Tamaño
idsectorcli	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblsis

Columnas

Nombre	Tipo	Tamaño
idsis	Texto	10
des	Texto	255
combo	Texto	1
columnacampo	Texto	255
tabla	Texto	255

Tabla: public_tblsubactividad

Columnas

Nombre	Tipo	Tamaño

idsubactividad	Texto	10
idactividad	Texto	10
descripcion	Texto	255
url	Texto	56
target	Texto	255
estado	Texto	1

Tabla: public_tblsucursal

Columnas

Nombre	Tipo	Tamaño
idsucursal	Texto	10
idciudad	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tbltaller

Columnas

Nombre	Tipo	Tamaño
idtaller	Texto	10
idciudad	Texto	10
nombre	Texto	255
direccion	Texto	255
telefonof	Texto	50
telefonc	Texto	50
fax	Texto	255
correoe	Texto	50
estado	Texto	1

Tabla: public_tbltecnico

Columnas

Nombre	Tipo	Tamaño

idtecnico	Texto	10
idsucursal	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tbltipocli

Columnas

Nombre	Tipo	Tamaño
idtipocli	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tbltipodocumento

Columnas

Nombre	Tipo	Tamaño
idtipodocumento	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tbltiponegocio

Columnas

Nombre	Tipo	Tamaño
idtiponegocio	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tbltipopoliza

Columnas

Nombre	Tipo	Tamaño

idtipopoliza	Texto	10
descripcion	Texto	255
estado	Texto	1

Tabla: public_tblusuario

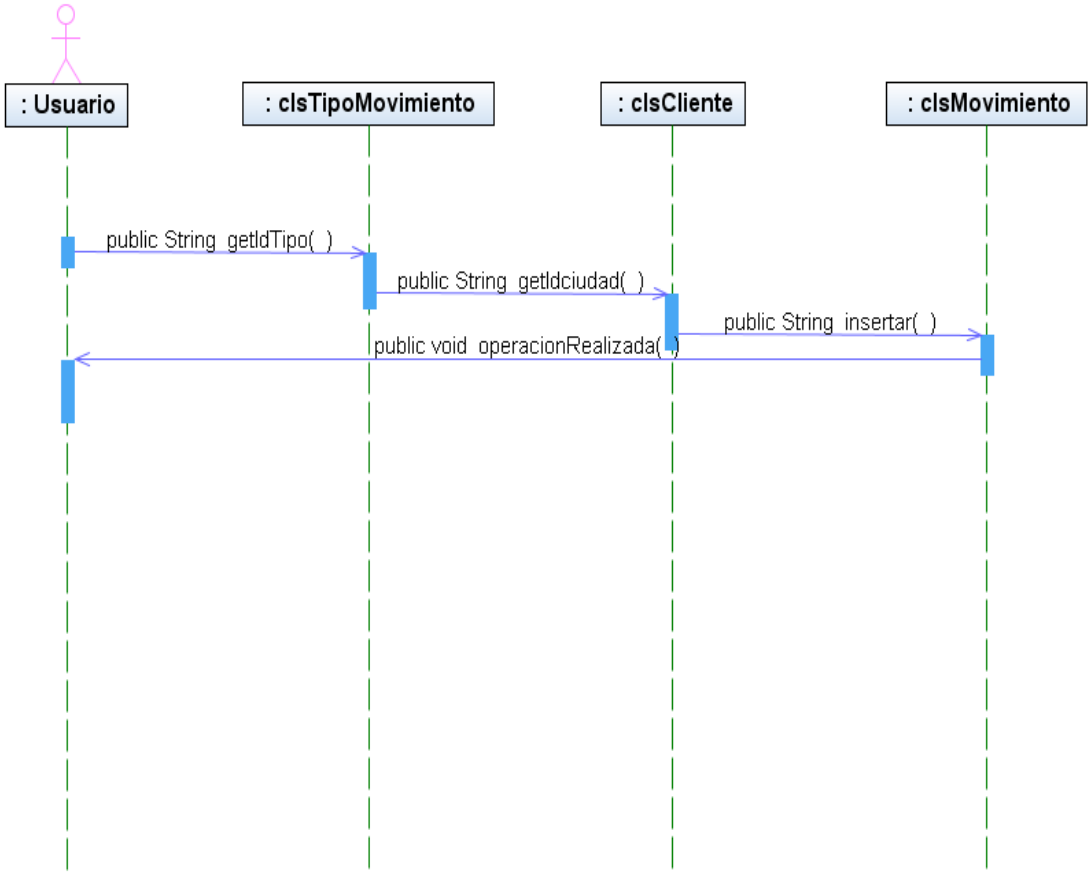
Columnas

Nombre	Tipo	Tamaño
idusuario	Texto	10
descripcion	Texto	255
pass	Texto	56
observaciones	Texto	255
estado	Texto	1

2.5. Diagrama de Secuencia

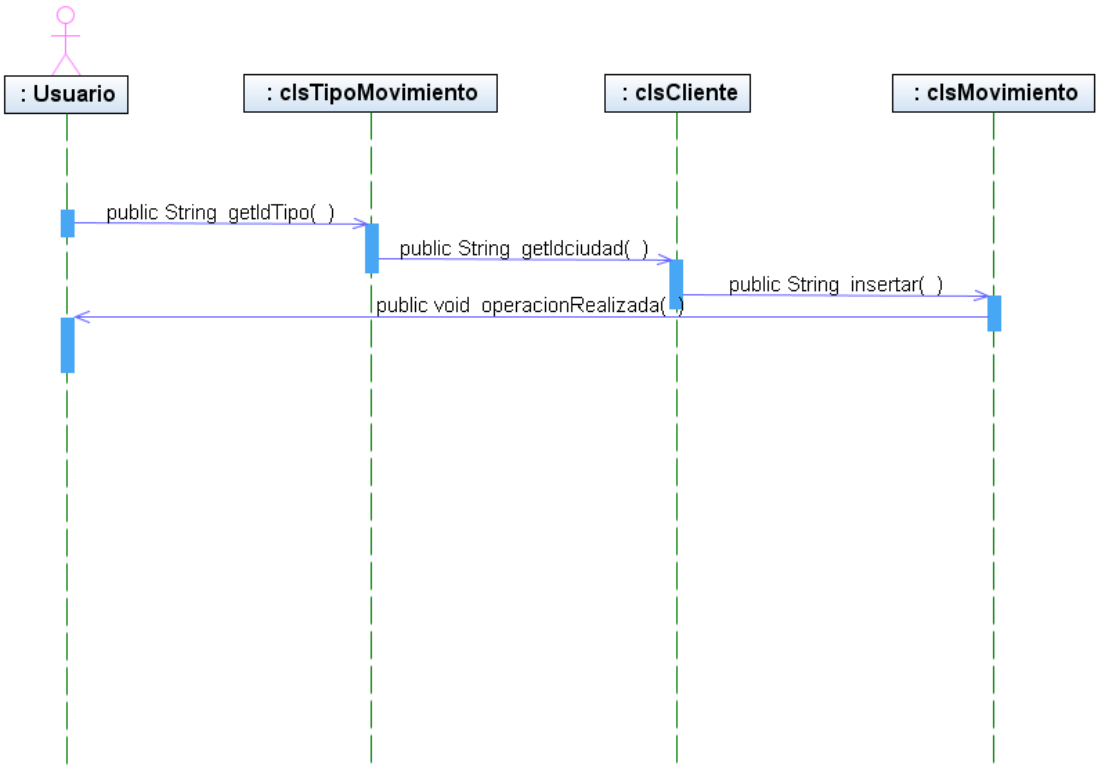
Registro de Cartera

Figura 7 Diagrama de Secuencia: Registro de Cartera



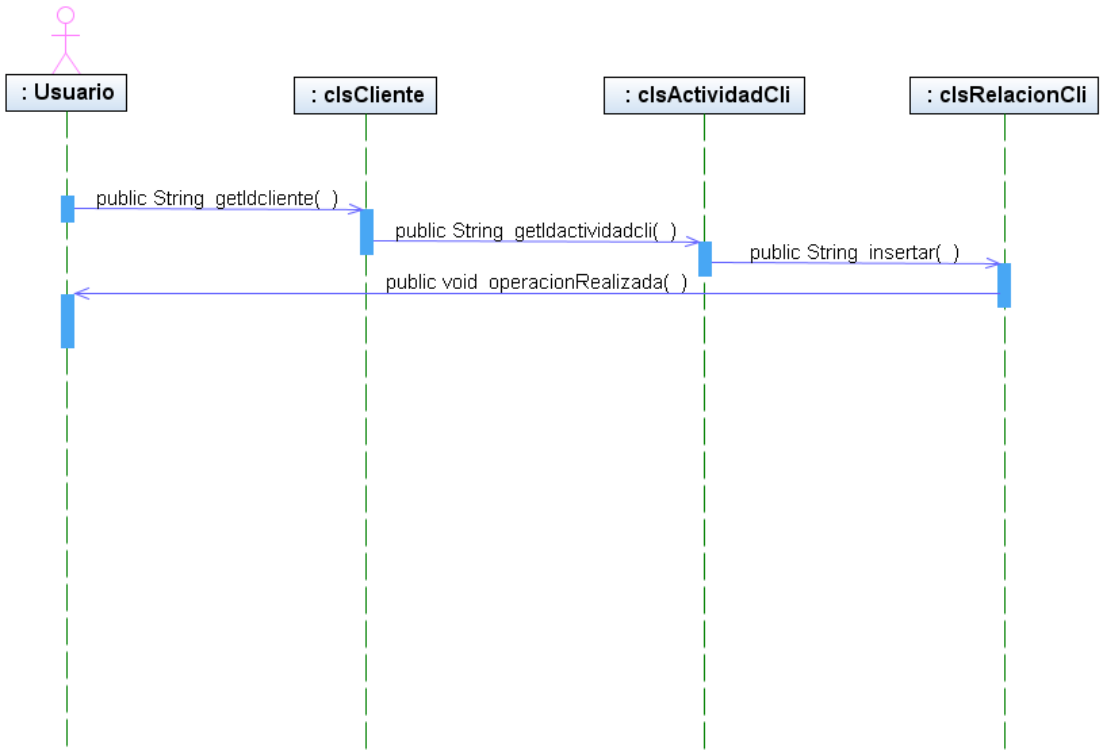
Estudio de Mercado (CRM)

Figura 8 Diagrama de Secuencia: Estudio de Mercado (CRM)



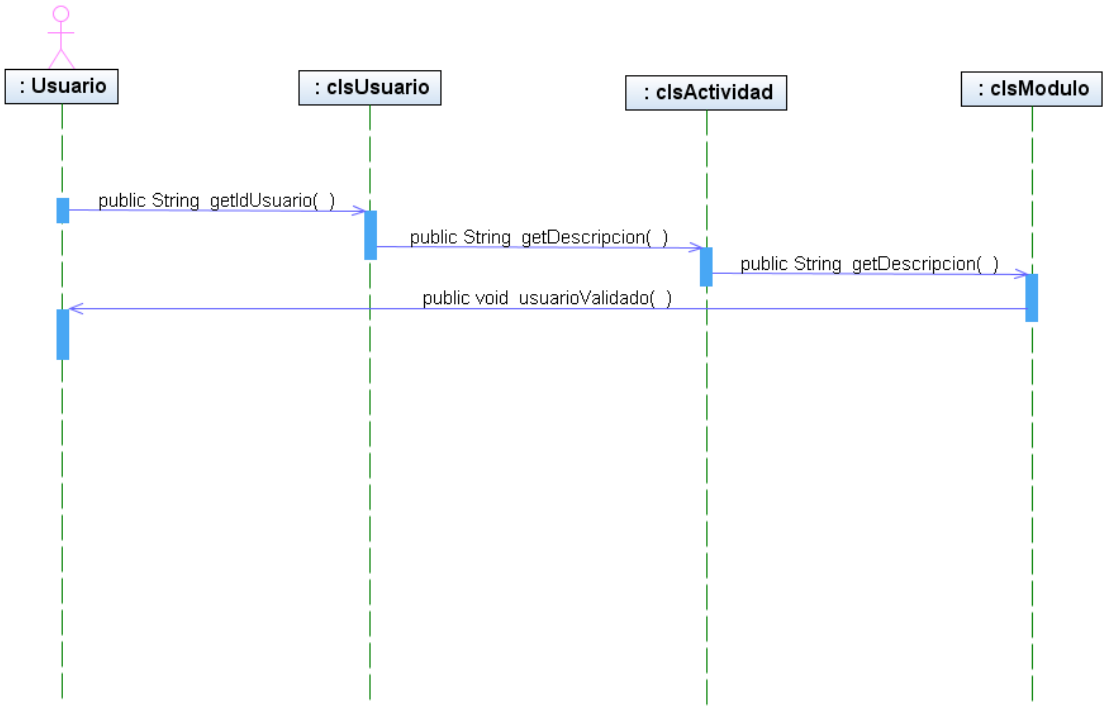
Registro Relación

Figura 9 Diagrama de Secuencia: Registro Relación



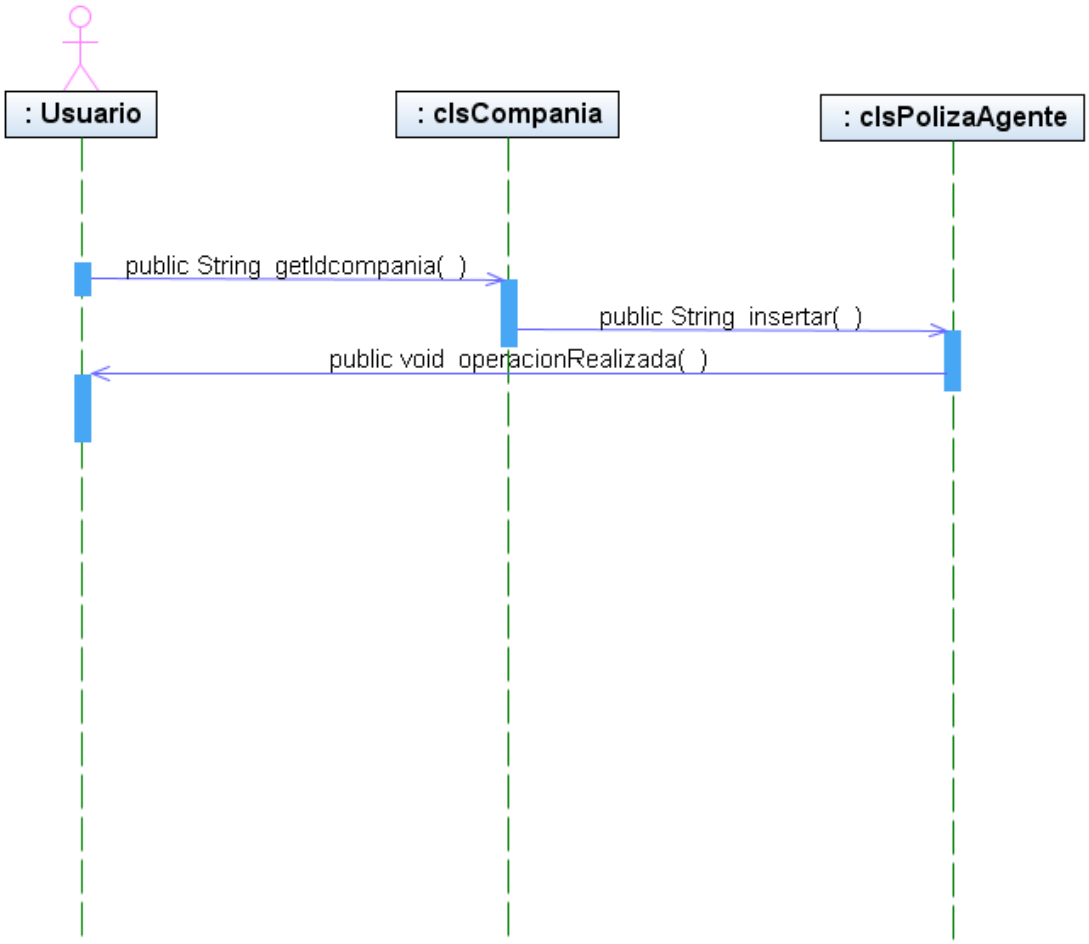
Ingreso al sistema

Figura 10 Diagrama de Secuencia: Ingreso al sistema



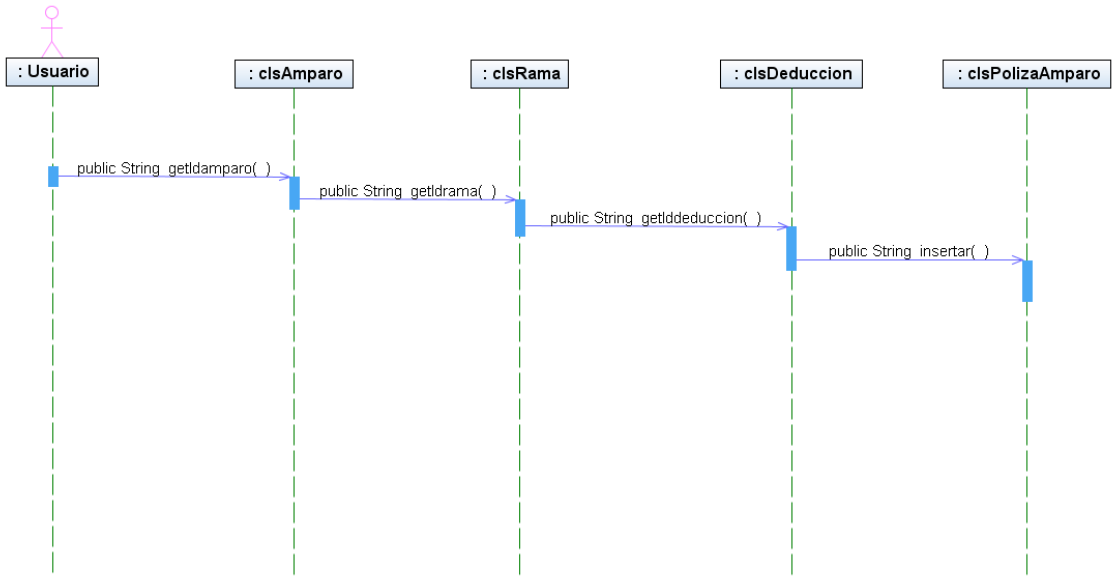
Póliza Agente

Figura 11 Diagrama de Secuencia: Póliza Agente



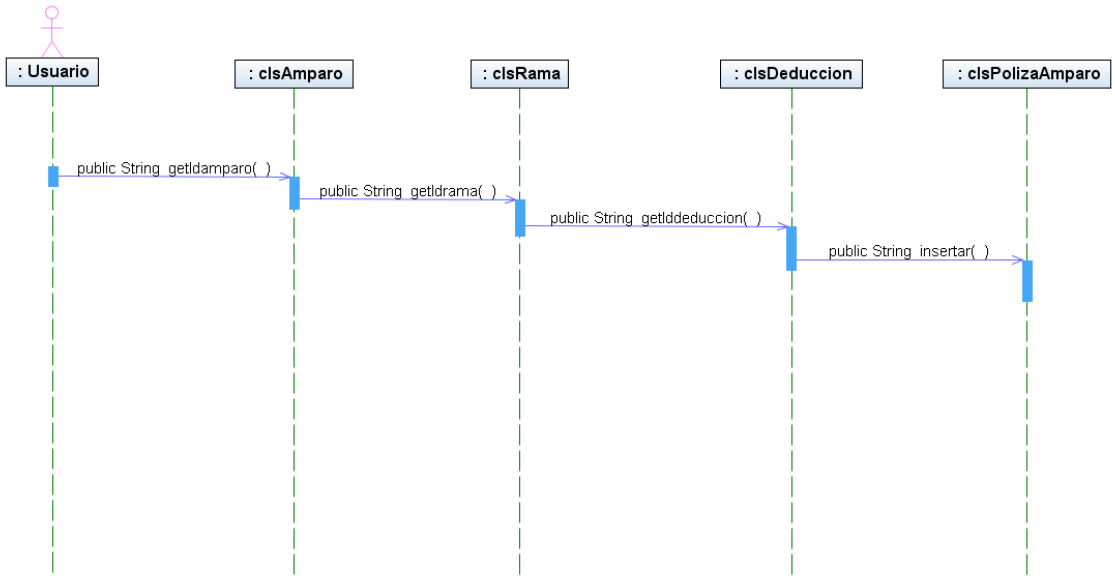
Póliza Amparo

Figura 12 Diagrama de Secuencia: Póliza Amparo



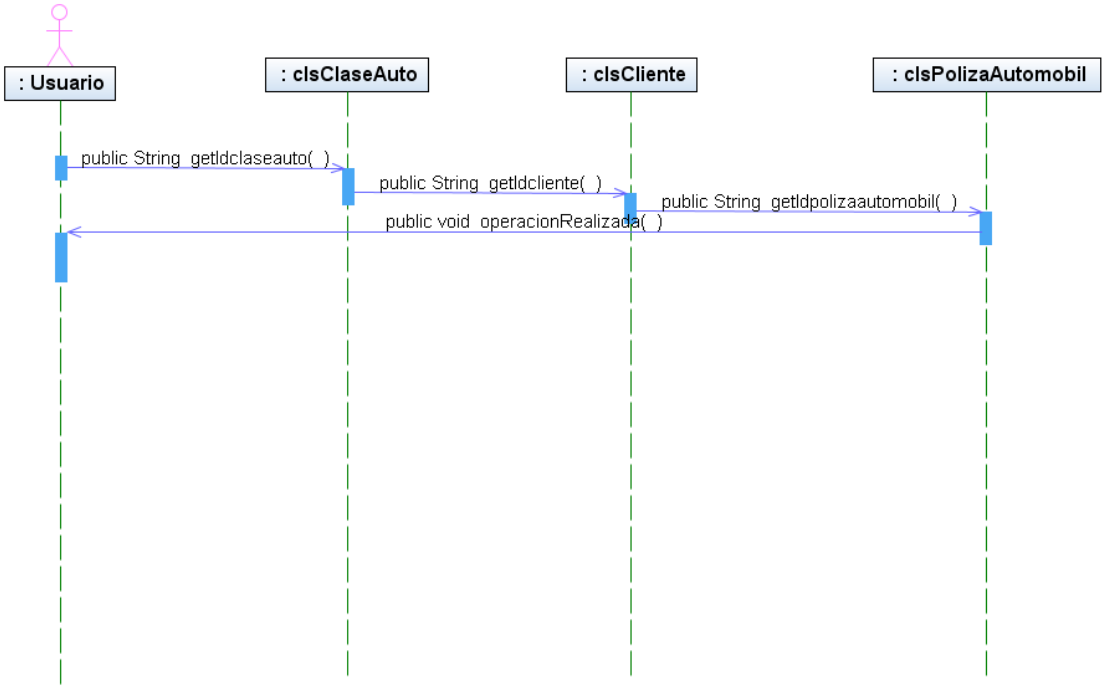
Póliza Artículo

Figura 13 Diagrama de Secuencia: Póliza Artículo



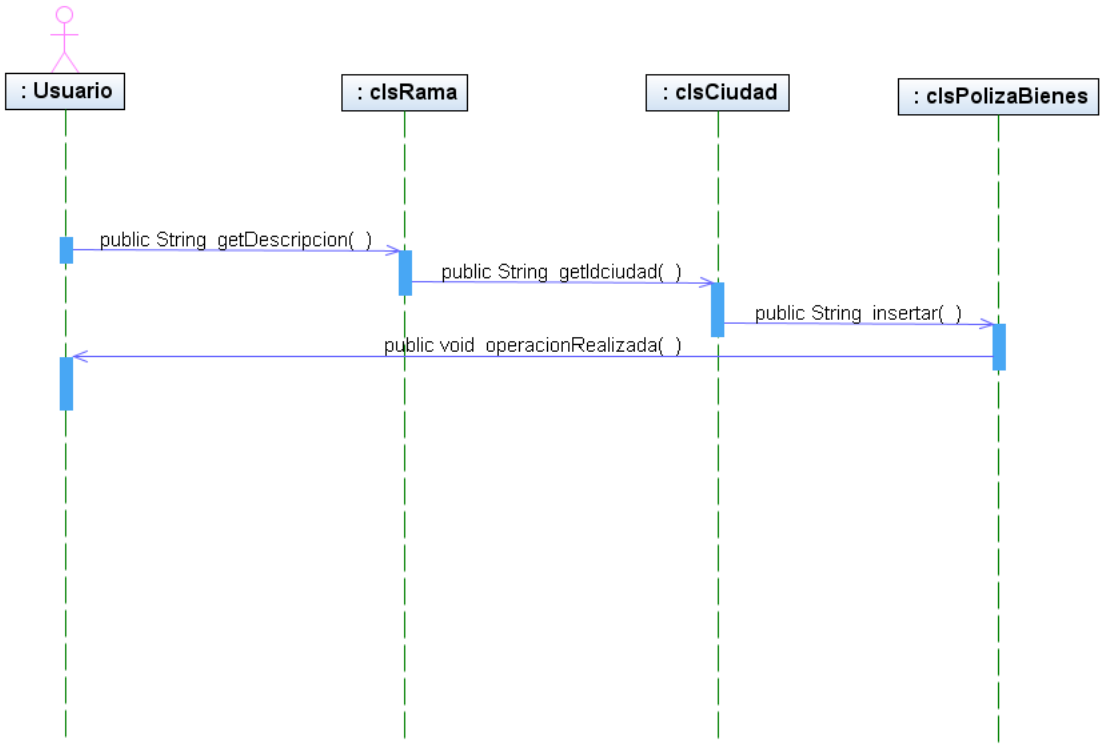
Póliza Automóvil

Figura 14 Diagrama de Secuencia: Póliza Automóvil



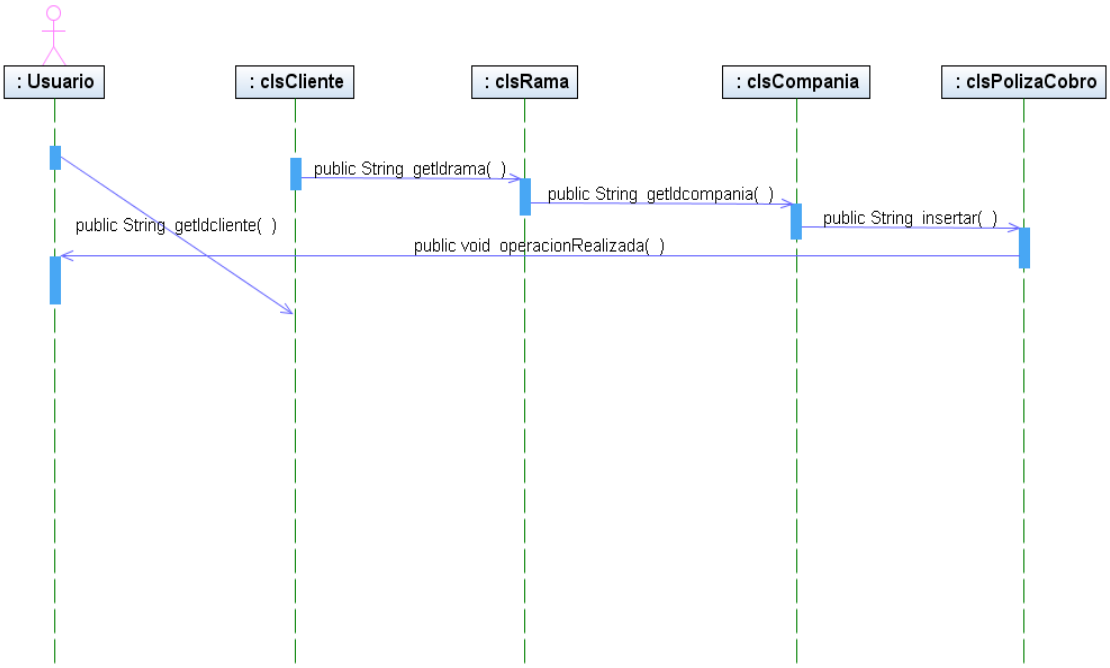
Póliza Bien

Figura 15 Diagrama de Secuencia: Póliza Bien



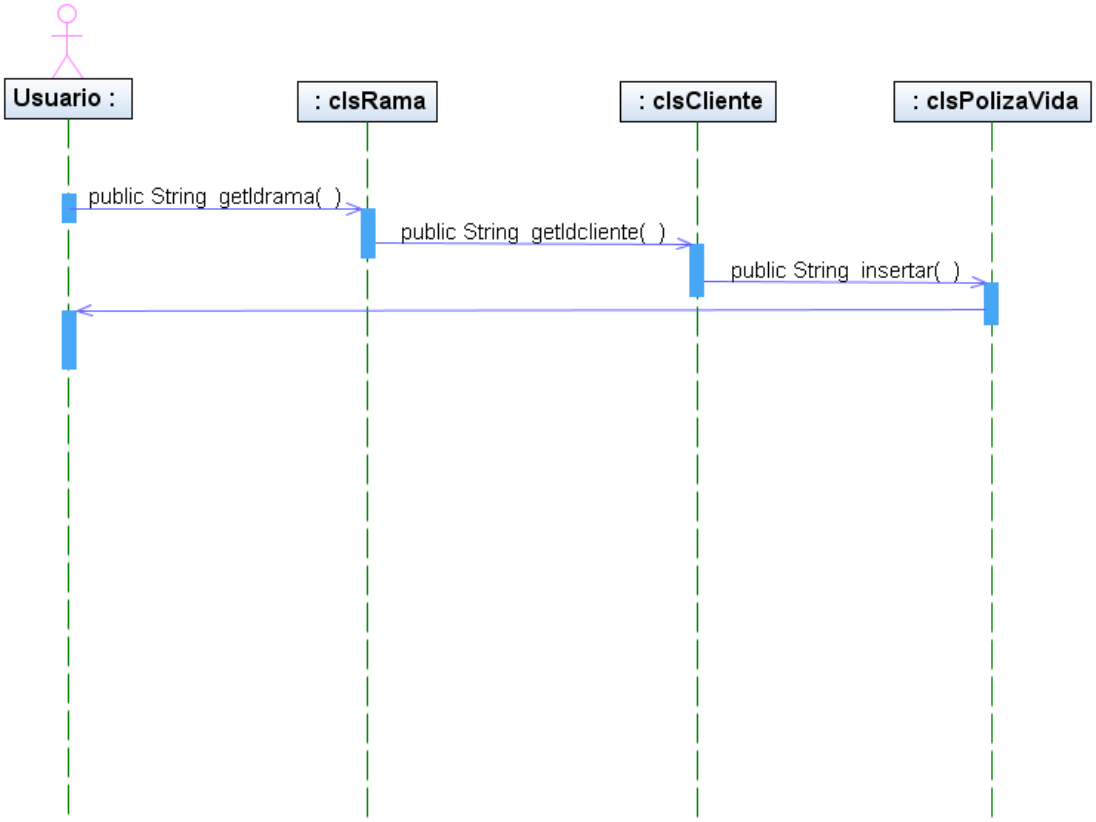
Póliza Cobro

Figura 16 Diagrama de Secuencia: Póliza Cobro



Póliza Vida

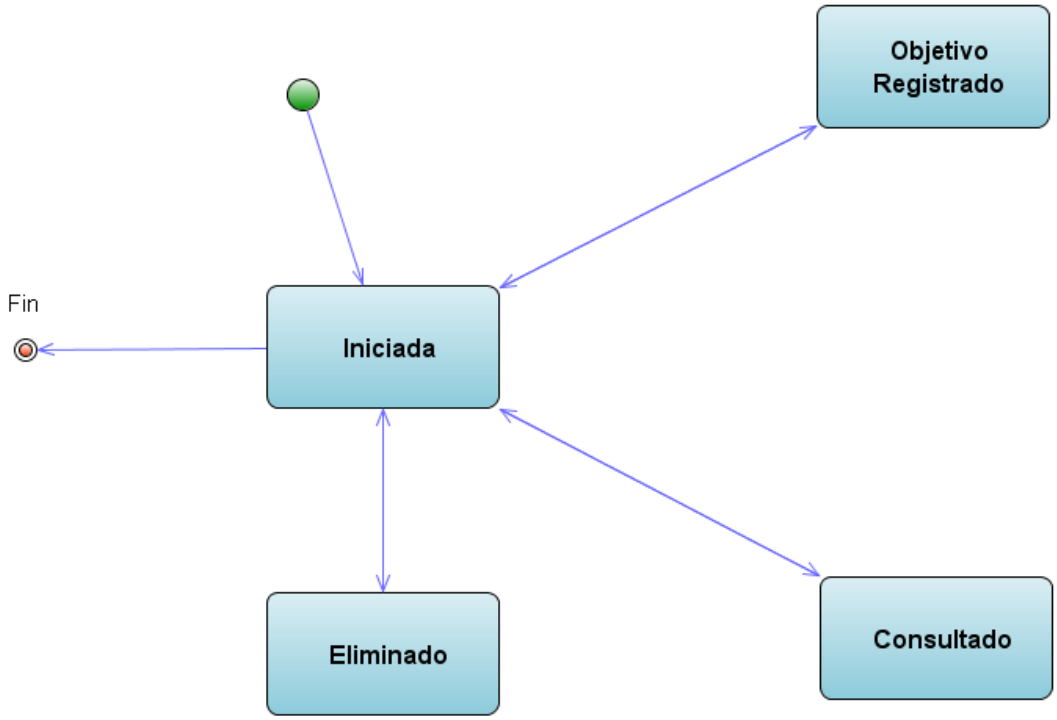
Figura 17 Diagrama de Secuencia: Póliza Vida



2.6. Diagrama de Estados

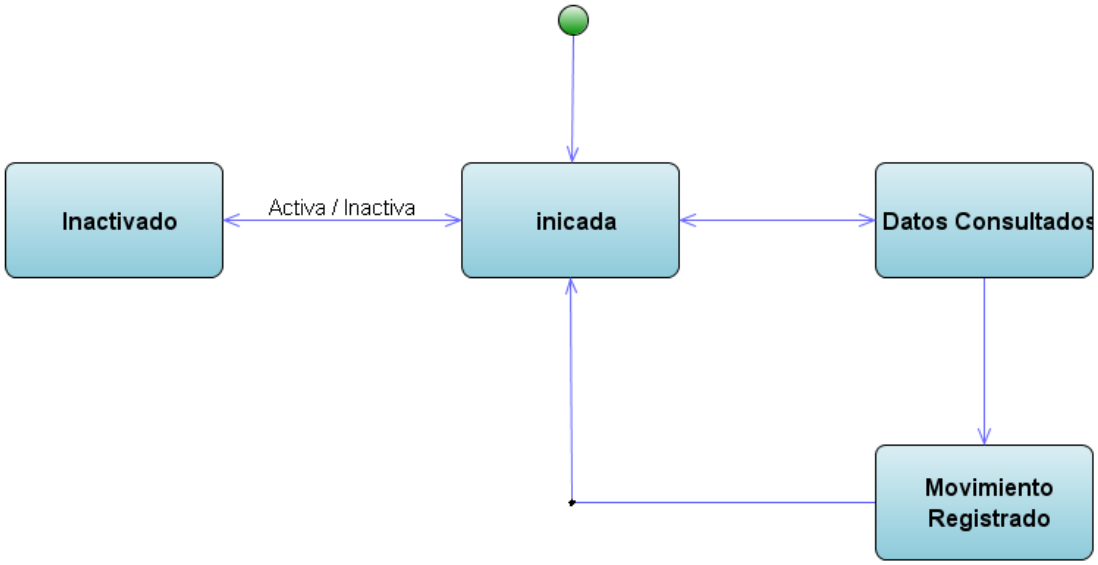
Clase Campaña

Figura 18 Diagrama de Estados: Clase Campaña



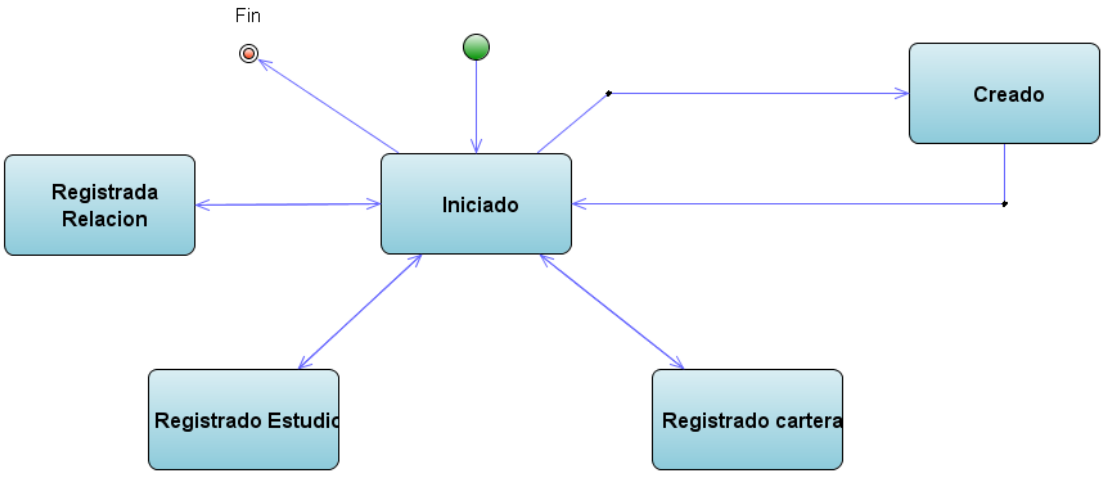
Clase Cartera

Figura 19 Diagrama de Estados: Clase Cartera



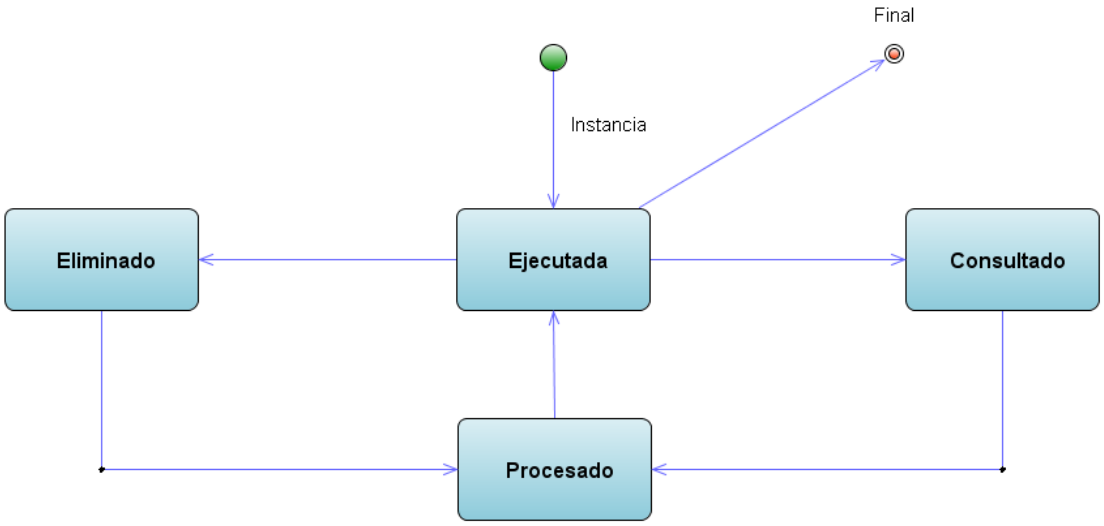
Clase Cliente

Figura 20 Diagrama de Estados: Clase Cliente



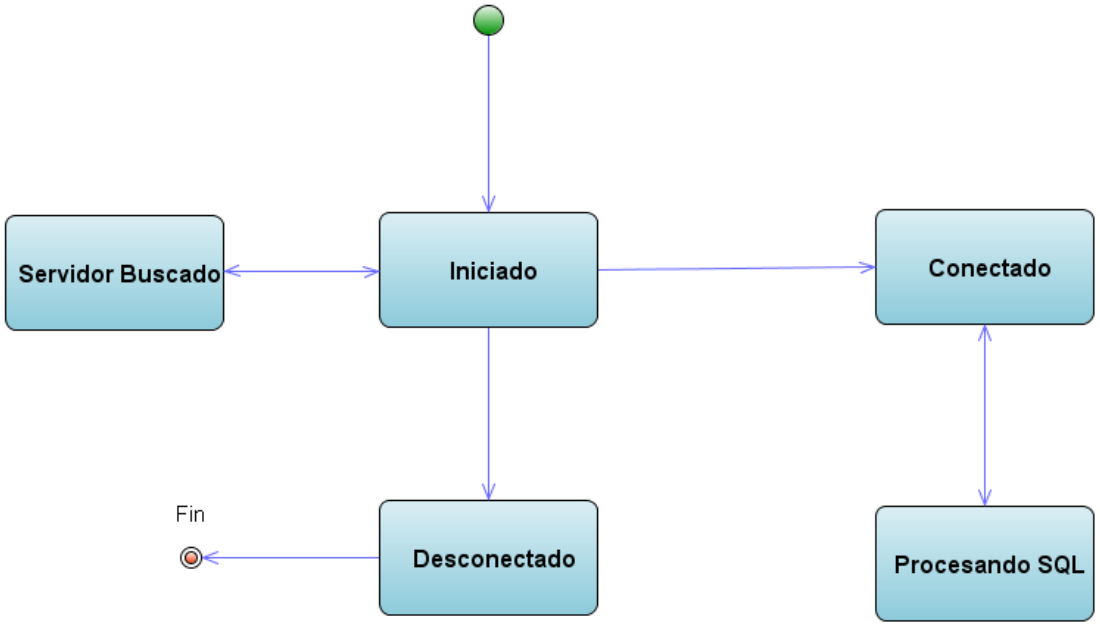
Clase Actividad

Figura 21 Diagrama de Estados: Clase Actividad



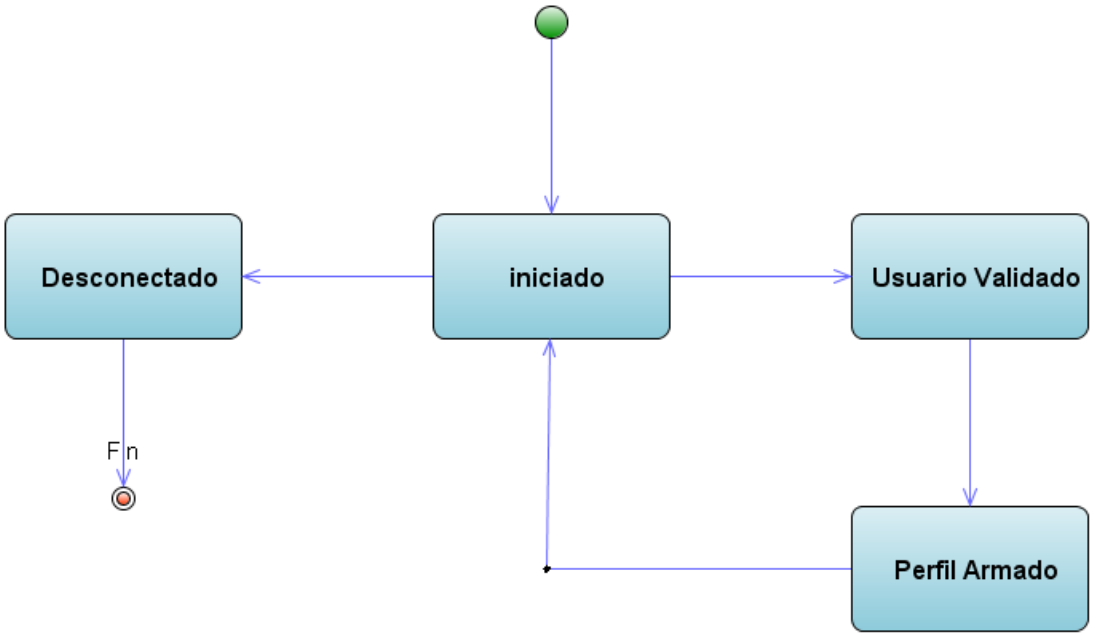
Clase Conn

Figura 22 Diagrama de Estados: Clase Conn



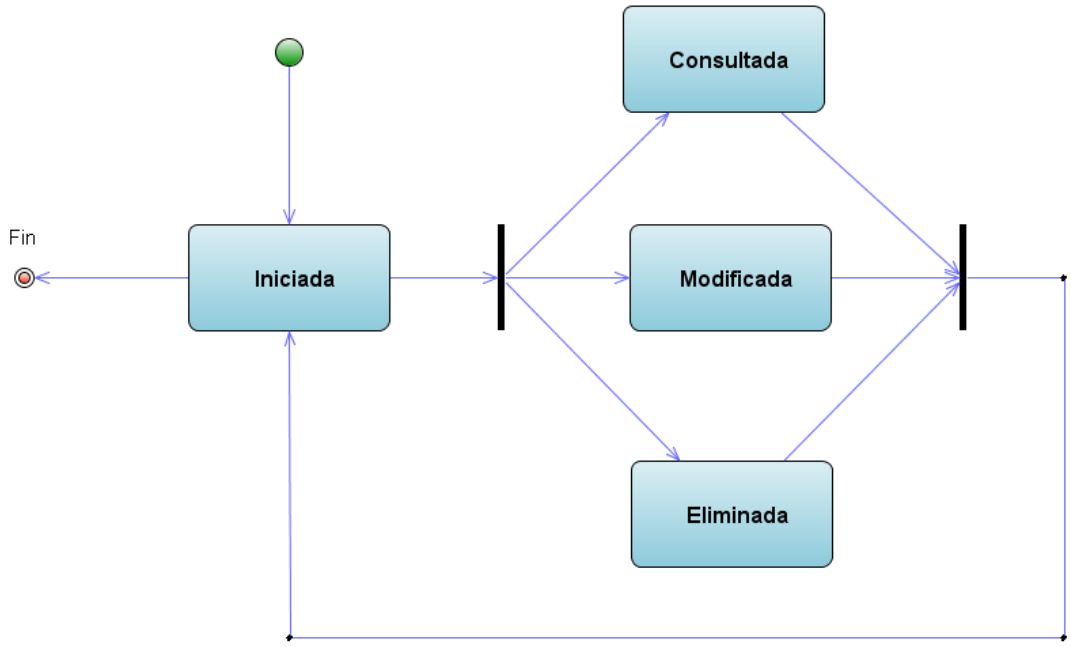
Clase Menú

Figura 23 Diagrama de Estados: Clase Menú



Clase Póliza

Figura 24 Diagrama de Estados: Clase Póliza



2.7. Diagrama de Colaboración

Diagrama de Colaboración: Registro de Cartera

Figura 25 Diagrama de Colaboración: Registro de Cartera

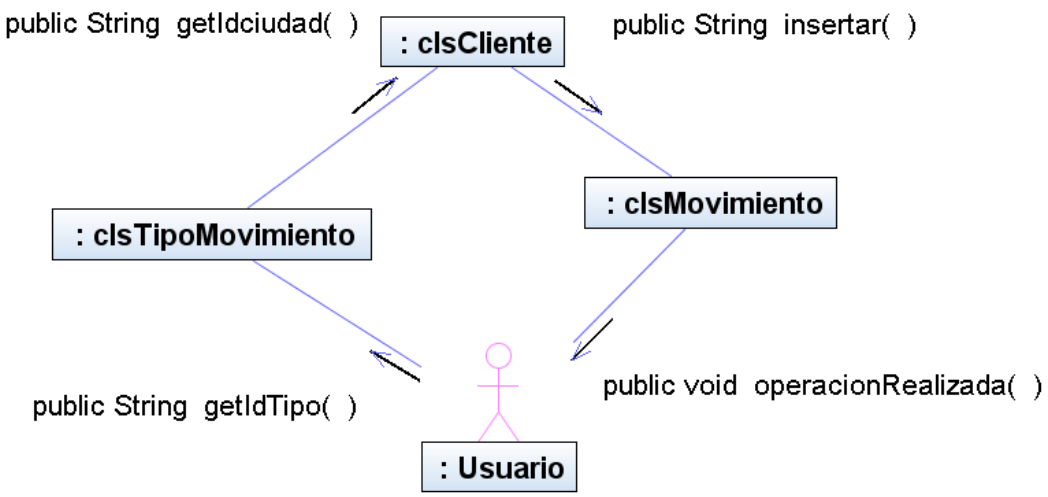


Diagrama de Colaboración: Estudio de Mercado (CRM)

Figura 26 Diagrama de Colaboración: Estudio de Mercado (CRM)

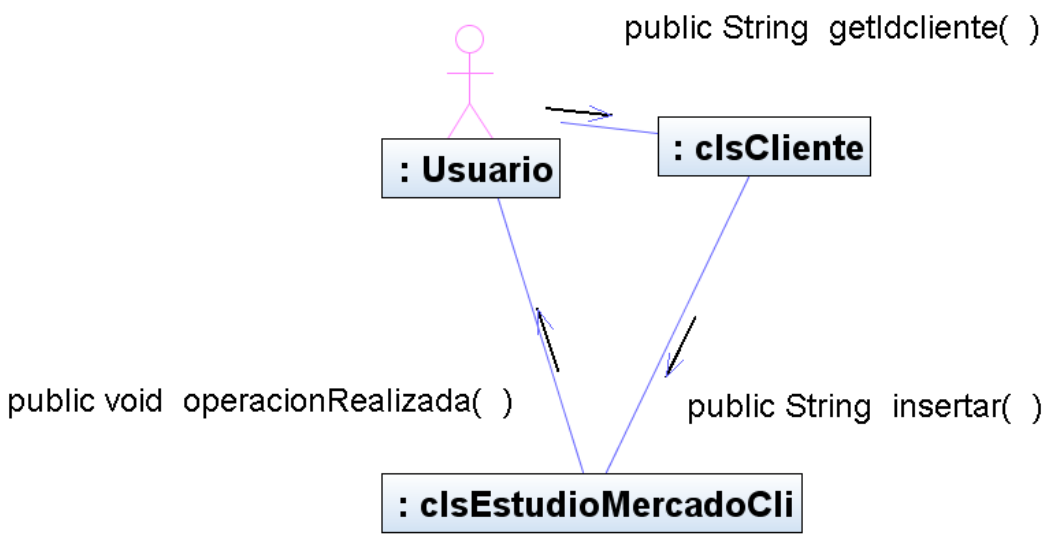


Diagrama de Colaboración: Registro Relación

Figura 27 Diagrama de Colaboración: Registro Relación

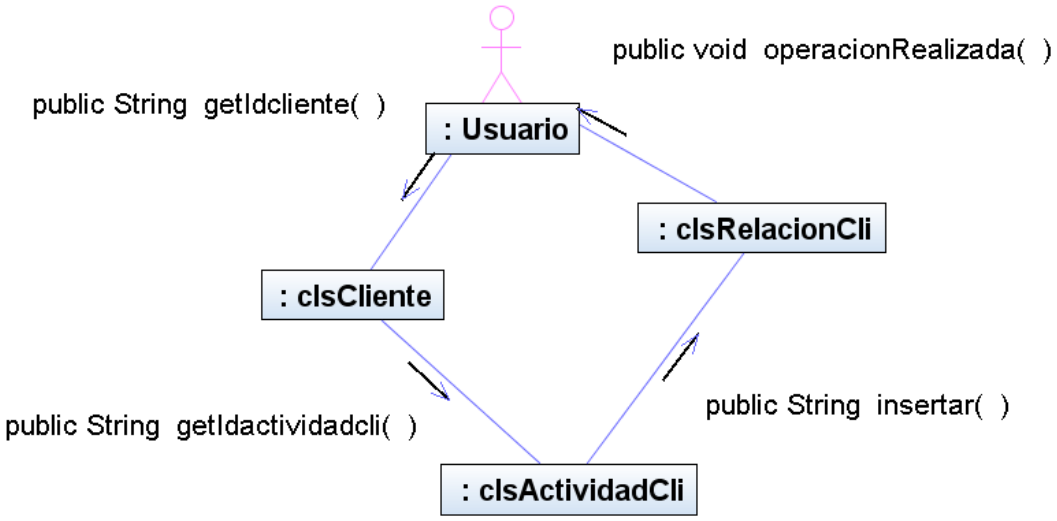


Diagrama de Colaboración: Ingreso al sistema

Figura 28 Diagrama de Colaboración: Ingreso al sistema

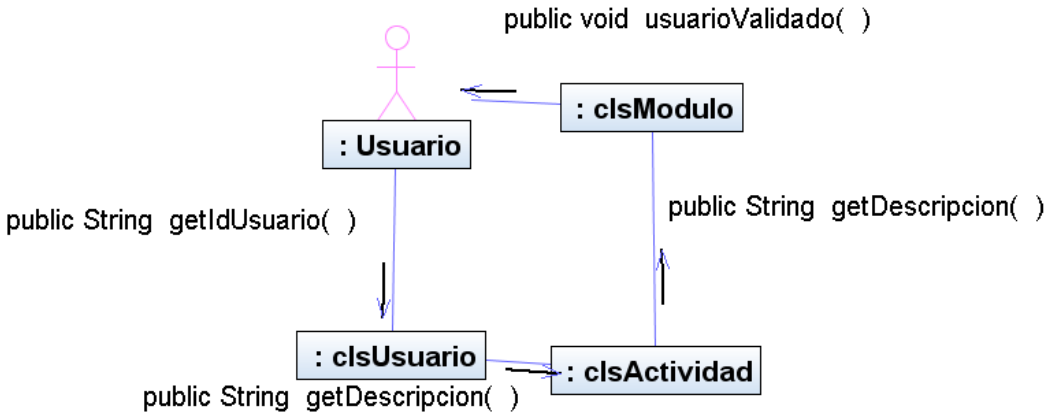


Diagrama de Colaboración: Póliza Agente

Figura 29 Diagrama de Colaboración: Póliza Agente

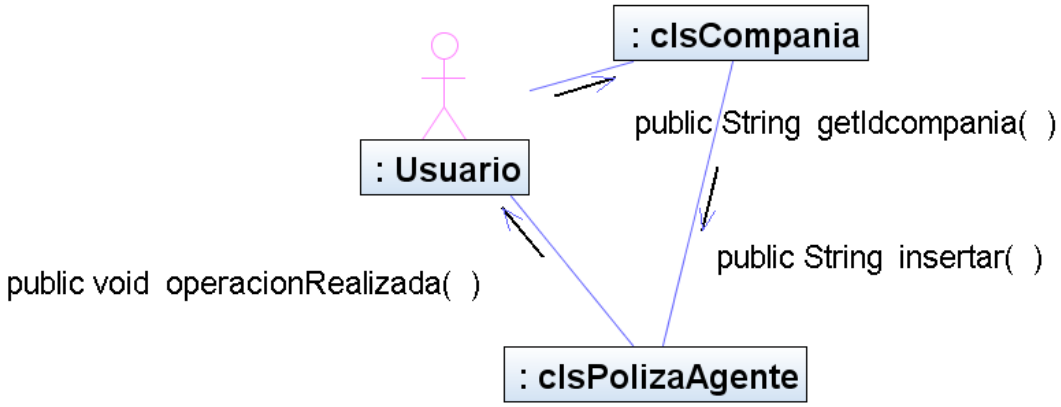


Diagrama de Colaboración: Póliza Amparo

Figura 30 Diagrama de Colaboración: Póliza Amparo

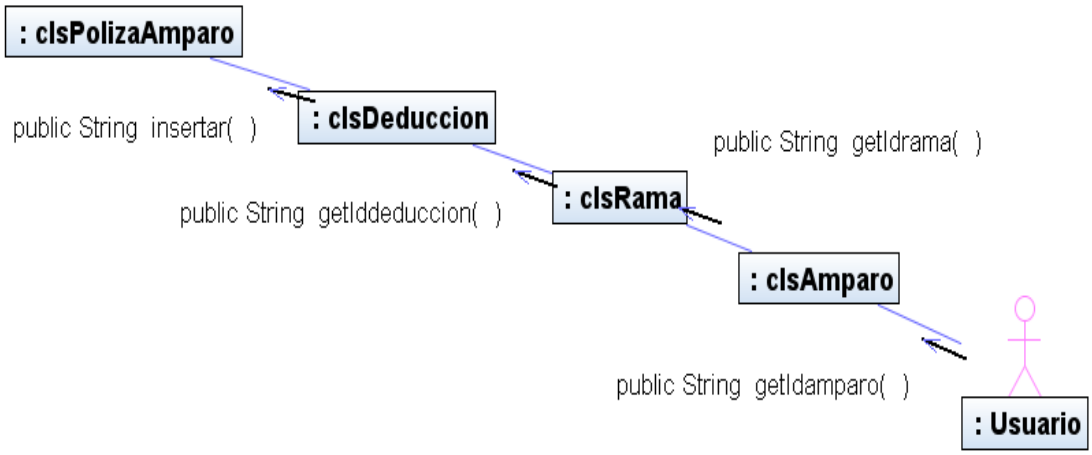


Diagrama de Colaboración: Póliza Artículo

Figura 31 Diagrama de Colaboración: Póliza Artículo

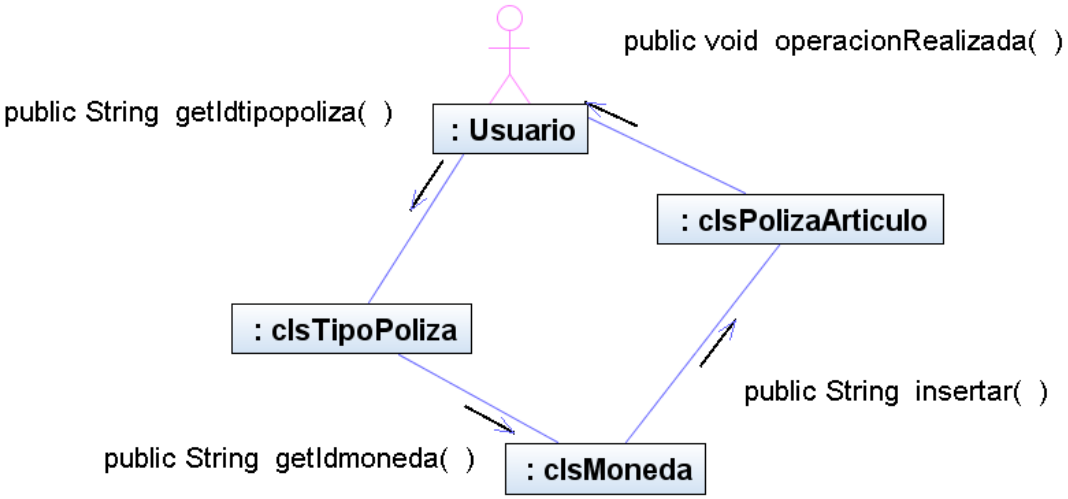


Diagrama de Colaboración: Póliza Automóvil

Figura 32 Diagrama de Colaboración: Póliza Automóvil

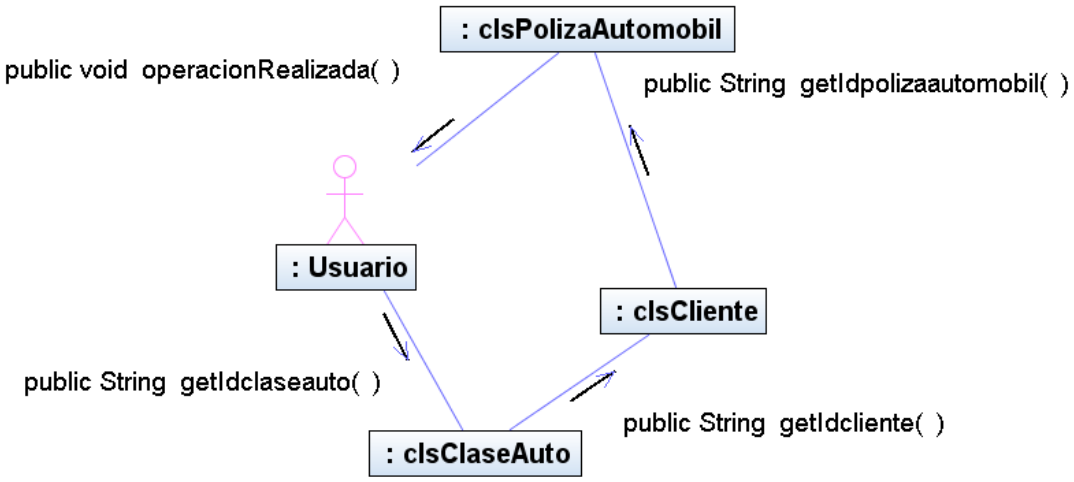


Diagrama de Colaboración: Póliza Bien

Figura 33 Diagrama de Colaboración: Póliza Bien

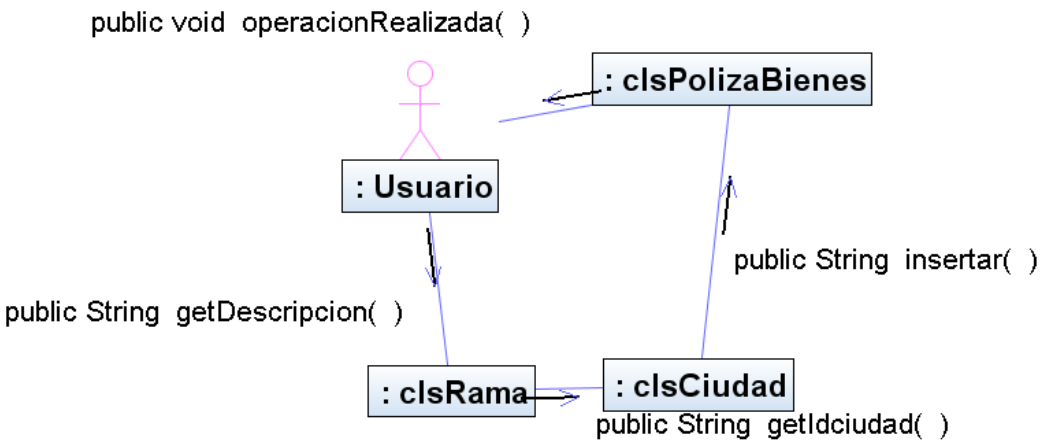


Diagrama de Colaboración: Póliza Cobro

Figura 34 Diagrama de Colaboración: Póliza Cobro

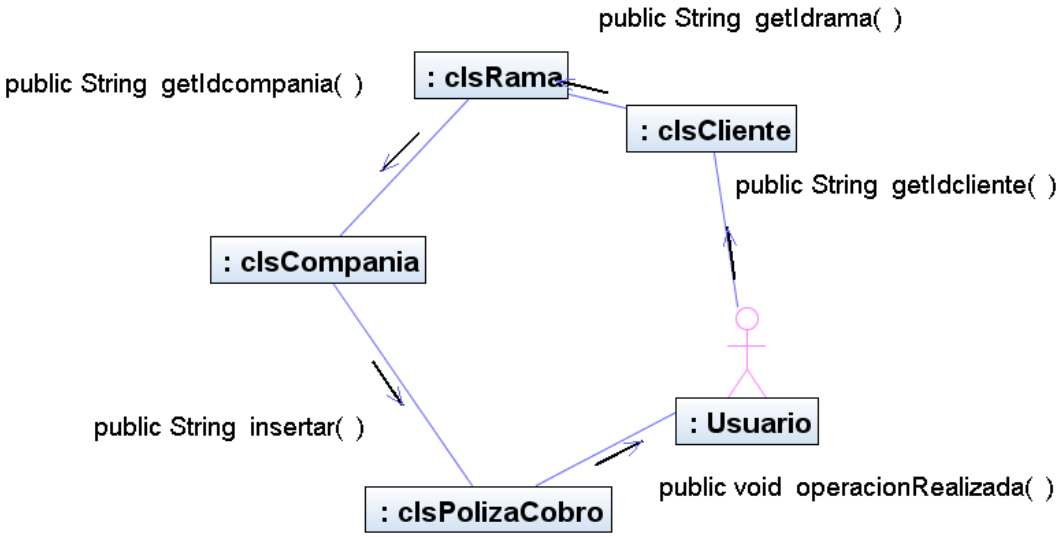
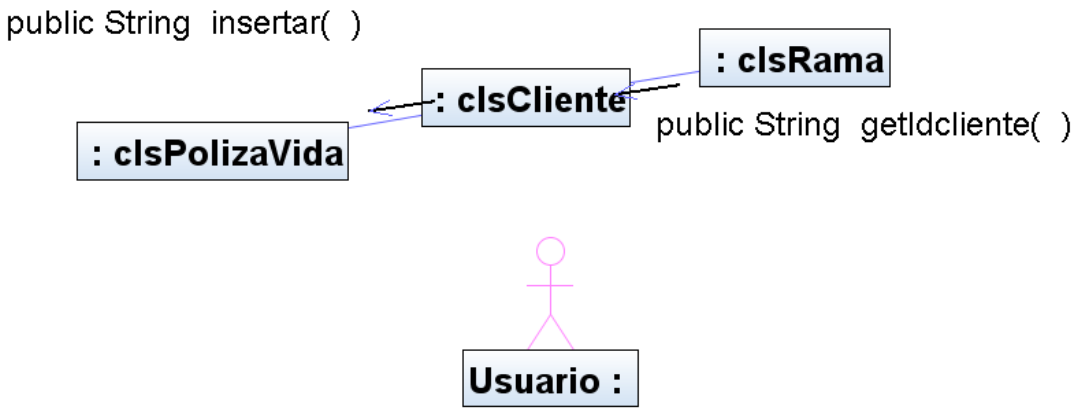


Diagrama de Colaboración: Póliza Vida

Figura 35 Diagrama de Colaboración: Póliza Vida



ANEXO C
DISEÑO

CONTENIDO

1. Requerimientos Técnicos del software

1.1. Software del Servidor

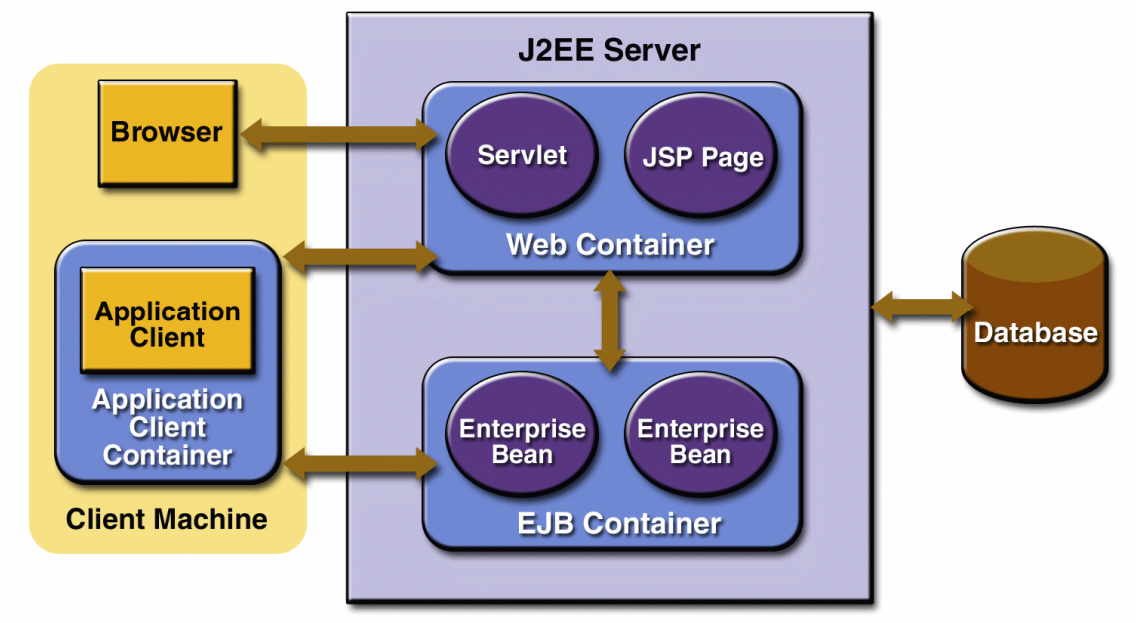
Para la implantación de las características analizadas, es necesario integrar en un mismo software el servidor web (servidor de contenido estático HTML), el servidor de aplicaciones (un plugin para Servlets/JSP), por lo cual se usa como servidor de web, la plataforma Tomcat, con lo que el cliente sólo necesitará usar el navegador.

Como definición fundamental para el proyecto de seguros, se considera un servidor de aplicaciones, el cual funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Fundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems.

La operación del Servidor Web será configurado para servir el contenido estático del portal, y que tome a Tomcat como un plugin para Servlets/JSP. Para poder integrar el servidor web al servidor de aplicaciones el servidor web necesita realizar unas operaciones específicas, ya que además de esperar peticiones de un cliente http tiene que añadir un contenedor de servlets. Entones realiza: Cargar la librería del adaptador del contenedor de servlets e inicializarlo (antes de servir peticiones). Cuando llega una petición, necesita chequear para ver si una cierta petición pertenece a un servlet, si es así necesita permitir que el adaptador tome el control y lo maneje. Por otro lado el adaptador necesita saber qué peticiones va a servir, usualmente basándose en algún patrón de la URL requerida, y dónde dirigir estas peticiones.

1.2 Recursos del hardware

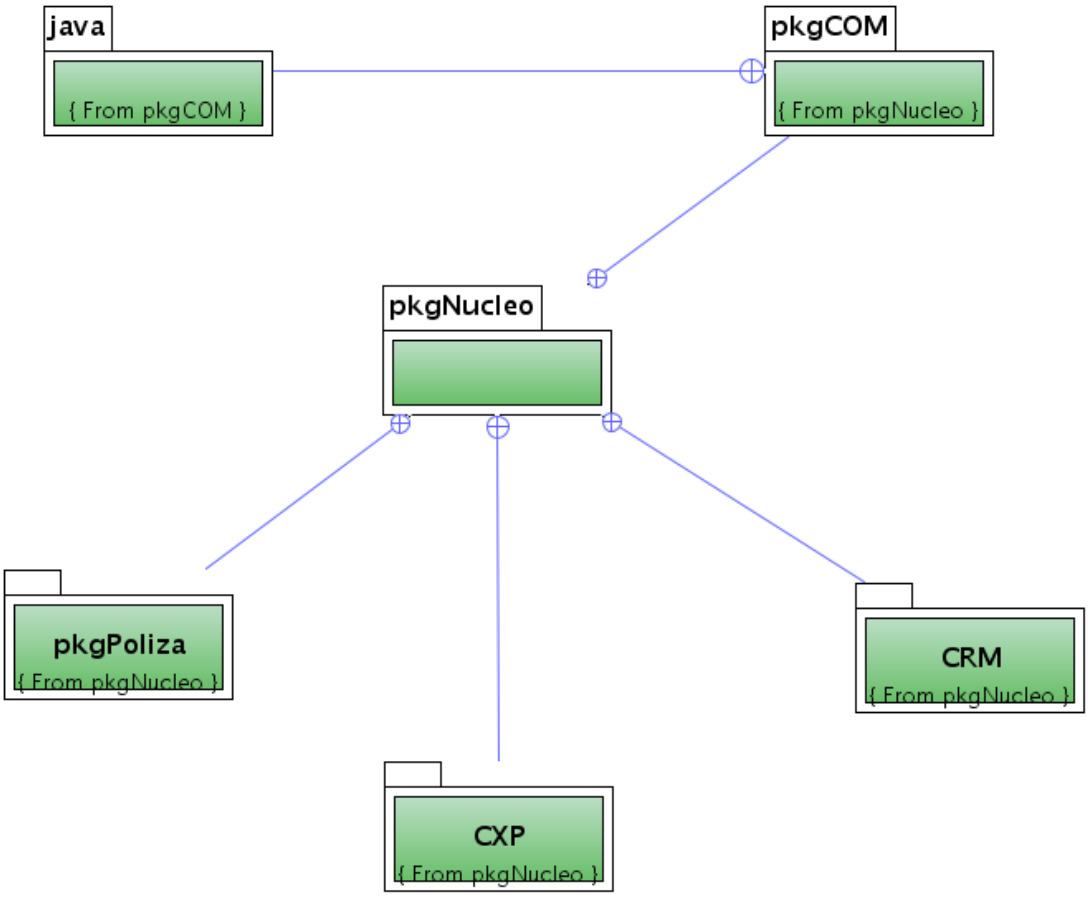
El diseño de la estructura de la red se basa en tener la base de datos con sus correspondientes gestores de base de datos Postgres. El acceso a las bases de datos por medio de los servidores estará controlado por un corta fuego, ya que la información que está guardada y gestionada en la base de datos no tiene que ser fácilmente accesible por nadie, porque se trata de información confidencial.



La arquitectura observada en la figura arriba descrita se basa en la arquitectura de funcionamiento J2EE, dónde todos los datos se guardan en las bases de datos, el procesado de la información en los servidores, y a través de Internet es cuando se presenta al usuario en forma de portal web. Por tanto en los servidores solamente se almacenarán los datos más utilizados por nuestros usuarios del portal.

2. Diseño del sistema

2.1. Identificación de Subsistemas - Diagrama de Paquetes



2.2. Medición de Datos y Proyección

Para la medición de datos, es necesario hacer referencia al término general del problema en bases de datos, el cual es dado por medio de un conjunto S que pertenece a U, dónde se desea recuperar los elementos de S que sean similares a uno dado, donde la similitud entre elementos es modelada mediante una función de distancia positiva d. El conjunto U denota el universo de objetos válidos y S, un subconjunto de U, denota la base de datos en donde buscamos. Básicamente, existen dos tipos de búsquedas de interés en espacios métricos:

Búsqueda por rango: recuperar todos los elementos de S a distancia r de un elemento q dado.

Búsqueda de los k vecinos más cercanos: dado q, recuperar los k elementos más cercanos a q.

Para el proyecto desarrollado, la evaluación de la función d, suele ser una operación costosa y así en la mayoría de los casos la cantidad de evaluaciones de

distancias necesarias para resolver la búsqueda, se usa como medida de complejidad. Integrar objetos de un espacio métrico en un ambiente de bases de datos requiere principalmente poder resolver consultas por similitud eficientemente. En aplicaciones reales es posible tener grandes volúmenes de datos, ya sea por la cantidad de objetos o por el tamaño de cada objeto. Ello implica que debemos contar con estructuras adecuadas y eficientes para memoria secundaria, y en ese caso debemos optimizar también la cantidad de E/S.

Además en un escenario real de un ambiente de base de datos es necesario contar con herramientas que, además de ser capaces de trabajar con grandes volúmenes de datos, usualmente permitan insertar o eliminar objetos dinámicamente y de manera eficiente. Un objeto de un espacio métrico puede ser una imagen, una huella digital, un documento, o cualquier otro tipo de objeto. Esta es una de las razones por las que los elementos de una base de datos métrica no se pueden almacenar en bases de datos relacionales, las cuales tienen tamaño fijo de tupla; lo que además implica que las operaciones sobre datos de un espacio métrico son, en general, más costosas que las operaciones relacionales estándar. Existen índices que, en principio, resuelven estos tipos de problemas; pero aún están muy inmaduros para ser usados en la vida real por dos motivos importantes: falta de dinamismo y necesidad de trabajar en memoria principal. Estas características son sobreentendidas en los índices para bases de datos tradicionales, y la investigación apunta a poner los índices para estas nuevas bases de datos a un nivel de madurez similar.

2.3 Elección de Herramientas de desarrollo y Sistema Operativo (Justificaciones)

Con el fin de definir algunos puntos importantes al momento de la elección de las herramienta de desarrollo, es necesario tener en cuenta que la programación orientada a la Web ha generado como consecuencia la creación de varias herramientas de desarrollo, por lo que es importante identificar cuáles ofrecen un mejor rendimiento y bajo qué Sistema Operativo. Las herramientas de programación Web analizadas fueron: **PHP, ASP y JSP**, los cuales deberán operar bajo los sistemas operativos Linux y Windows utilizando criterios comunes[18].

Basados en un estudio descriptivo - deductivo; se analizan los prototipos en el que se muestra el funcionamiento de las herramientas mencionadas, con bases de datos bajo Windows/Linux. Basados en páginas Web prototipo, y en implementaciones basadas en PHP, ASP y JSP (consultadas en internet). Para

[18] Análisis comparativo de las herramientas de programación Web: PHP, ASP y JSP, bajo los sistemas operativos Linux y Windows. [En Línea]. Fecha de Consulta: 10.11.2007. Disponible en : http://dialnet.unirioja.es

tales efectos, es necesario tener en cuenta los servidores Web ISS, PWS, Apache, Tomcat e Instan ASP para la ejecución de cada proyecto en las diferentes tecnologías, teniendo en cuenta las siguientes variables: Tiempo de respuesta, complejidad de la programación, integridad de la base de datos, arquitectura de software y hardware, detección de fallas, confiabilidad y portabilidad.

Según un estudio Realizado por la Universidad del Norte*, se realizaron, de 150 pruebas, PHP mantuvo la integridad en base de datos en el 88% de las veces en Windows, y el 94% en Linux. El resultado de esto fue mayor en Windows que en Linux para ASP (47% Vs 45%), a diferencia de JSP (46% Vs 49%). Sin embargo, a pesar que PHP mantuvo la integridad en base de datos en un gran porcentaje en ambos sistemas operativos, en contraste con las otras herramientas, que no llegaron a la mitad, la diferencia no fue estadísticamente significativa (p: 0.95505).

2.4. Elección del Gestor de datos

PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle. La siguiente es una breve lista de algunas de esas características, a partir de PostgreSQL 7.1.x.

DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas. Ejemplos de su avanzada funcionalidad son consultas SQL declarativas, control de concurrencia multi-versión, soporte multi-usuario, transactions, optimización de consultas, herencia, y arrays.

Altamente_Extensible: PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.

Soporte_SQL_Comprensivo: PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.

Integridad Referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

* Documento basado en Análisis comparativo de las herramientas de programación Web: PHP, ASP y JSP, bajo los sistemas operativos Linux y Windows

Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

MVCC: MVCC, o Control de Concurrencia Multi-Versión (Multi-Version Concurrency Control), es la tecnología que PostgreSQL usa para evitar bloqueos innecesarios. Si alguna vez ha usado algún DBMS con capacidades SQL, tal como MySQL o Access, probablemente habrá notado que hay ocasiones en las una lectura tiene que esperar para acceder a información de la base de datos. La espera está provocada por usuarios que están escribiendo en la base de datos. Resumiendo, el lector está bloqueado por los escritores que están actualizando registros. Mediante el uso de MVCC, PostgreSQL evita este problema por completo. MVCC está considerado mejor que el bloqueo a nivel de fila porque un lector nunca es bloqueado por un escritor. En su lugar, PostgreSQL mantiene una ruta a todas las transacciones realizadas por los usuarios de la base de datos. PostgreSQL es capaz entonces de manejar los registros sin necesidad de que los usuarios tengan que esperar a que los registros estén disponibles.

Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Esta es similar al método del Apache 1.3.x para manejar procesos. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

Write Ahead Logging (WAL): La característica de PostgreSQL conocida como Write Ahead Logging incrementa la dependencia de la base de datosal registro de cambios antes de que estos sean escritos en la base de datos. Esto garantiza que en el hipotético caso de que la base de datos se caiga, existirá un registro de las transacciones a partir del cual se puede restaurar la base de datos. Esto puede ser enormemente beneficioso en el caso de caída, ya que cualesquiera cambios que no fueron escritos en la base de datos pueden ser recuperados usando el dato que fue previamente registrado. Una vez el sistema ha quedado restaurado, un usuario puede continuar trabajando desde el punto en que lo dejó cuando cayó la base de datos.

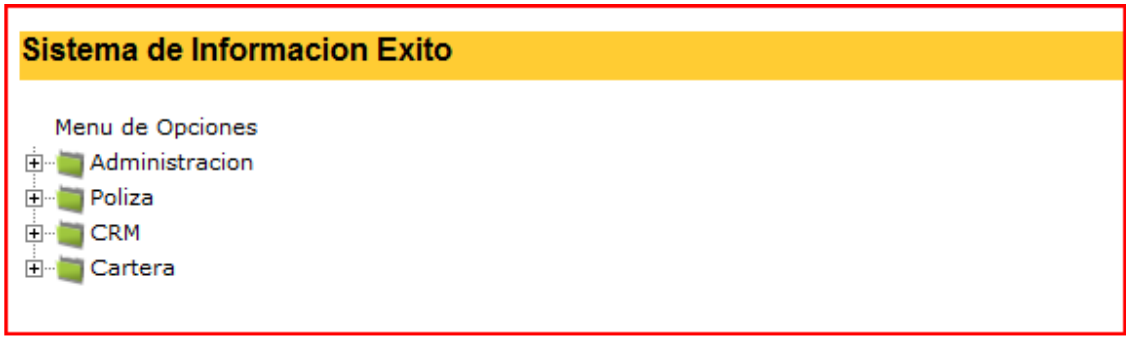
3. Diseño de interfaces interactivas – Prototipos

Para la definición de prototipos, es necesario tomar en cuenta los más representativos, los cuales, definen la estructura del sistema en términos generales.

Con el fin de plantear los prototipos del sistema, se plantean a continuación los elementos diseñados en HTML, los cuales, serán la base fundamental de cada interfaz en el sistema.

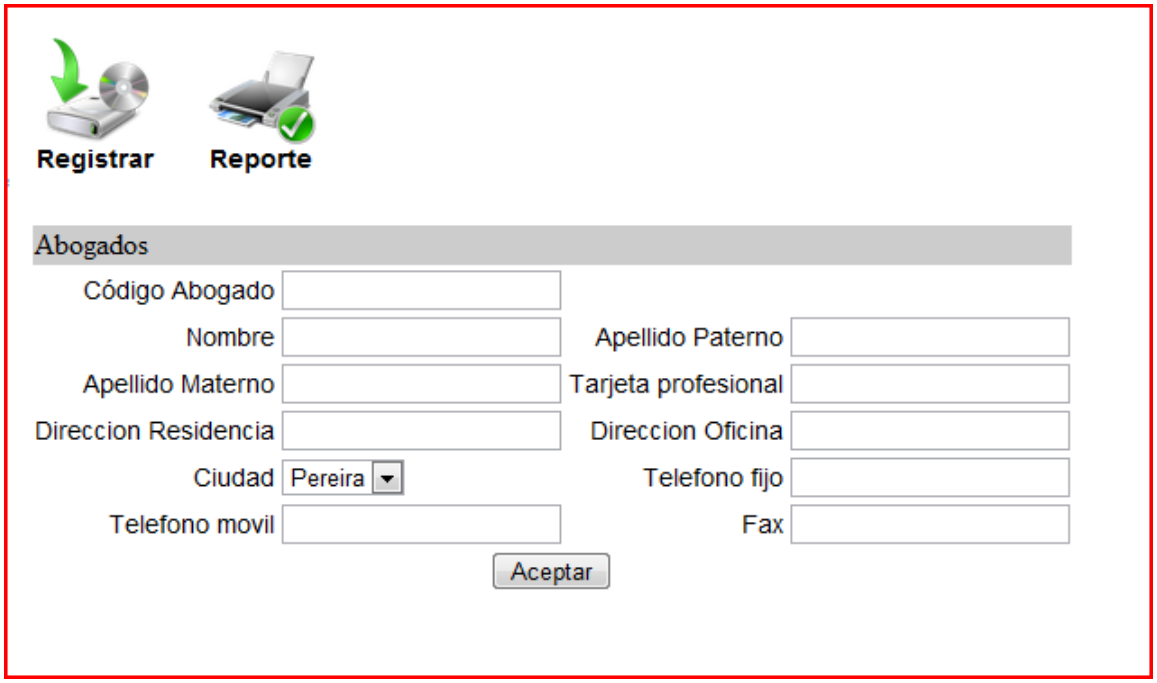
Página Principal

Estructura General de Menú

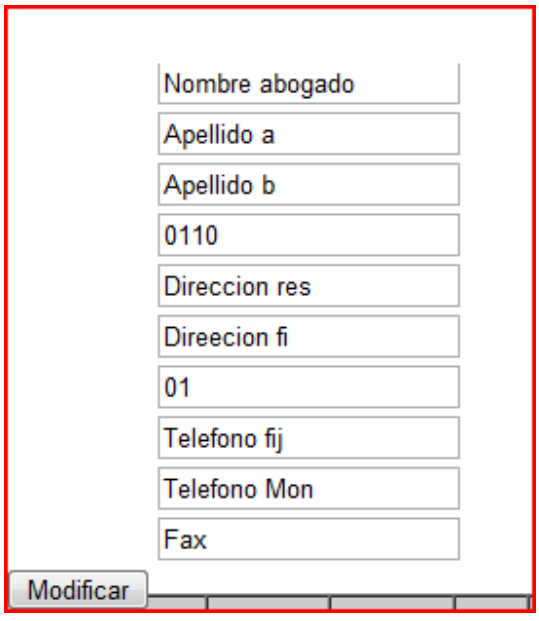


Menú Abogado

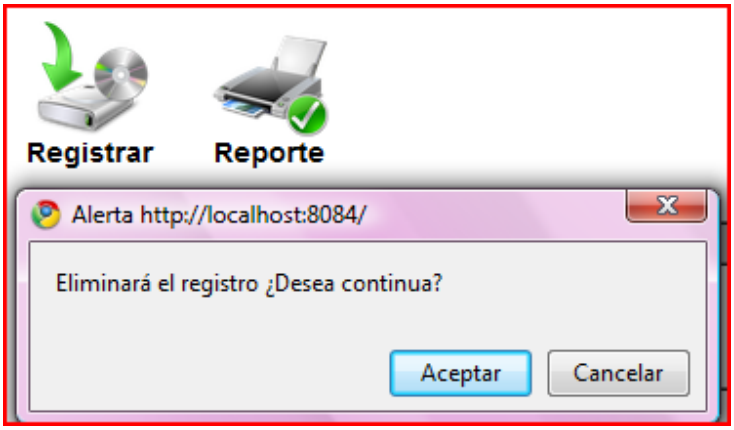
Ingresar Abogado



Editar Abogado



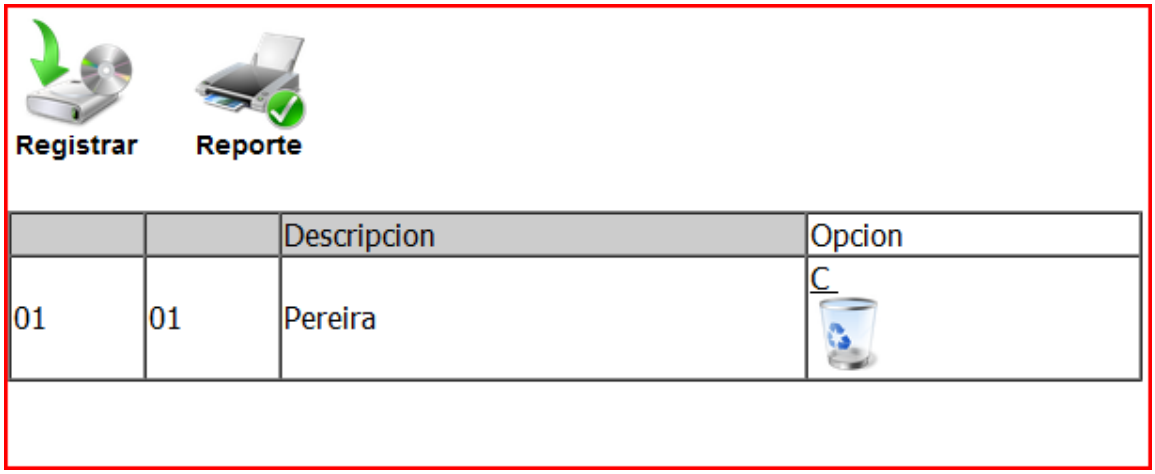
Eliminar Abogado



Menú Ciudad

Registrar Ciudad

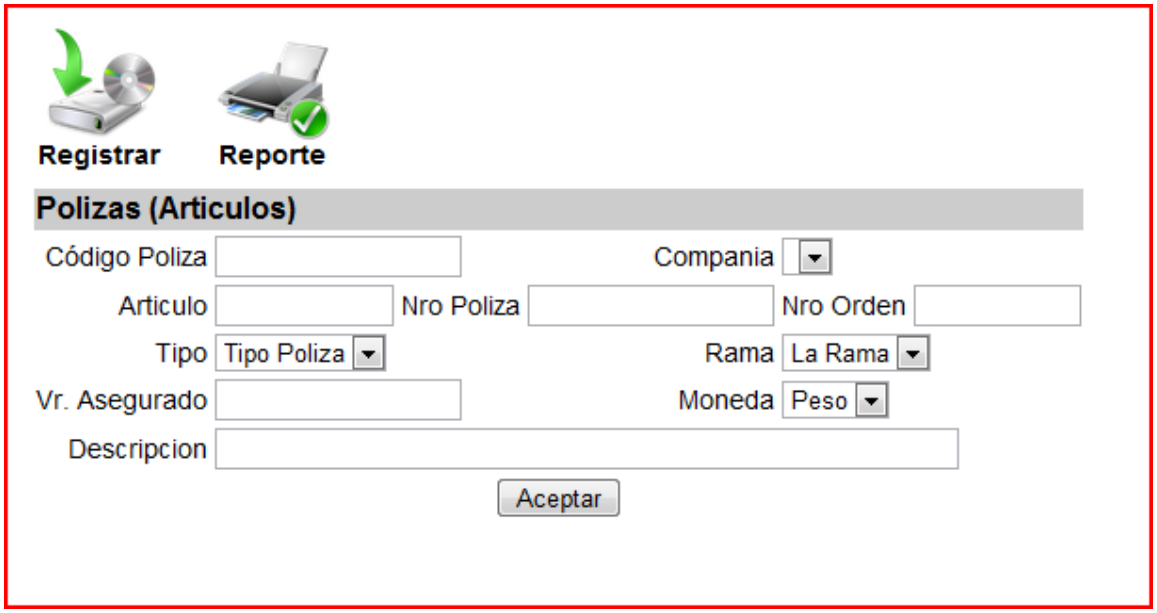
Listar Ciudad



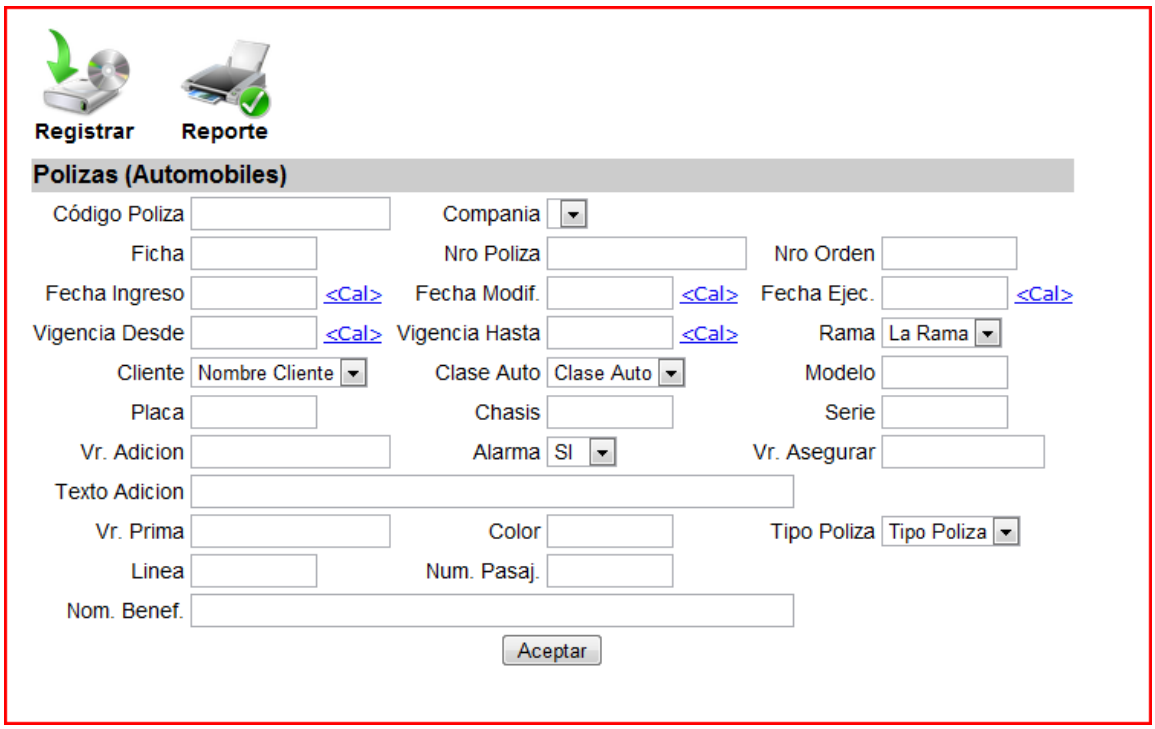
Registrar Póliza Agente

Registrar Póliza Amparo

Registrar Póliza Artículo

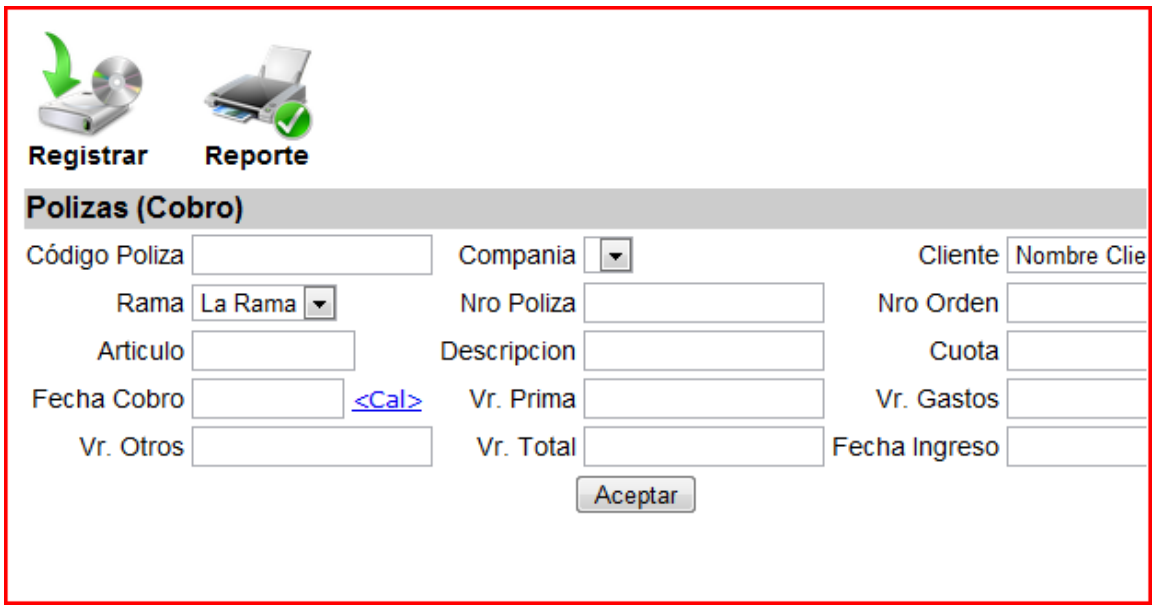


Registrar Póliza Automóvil



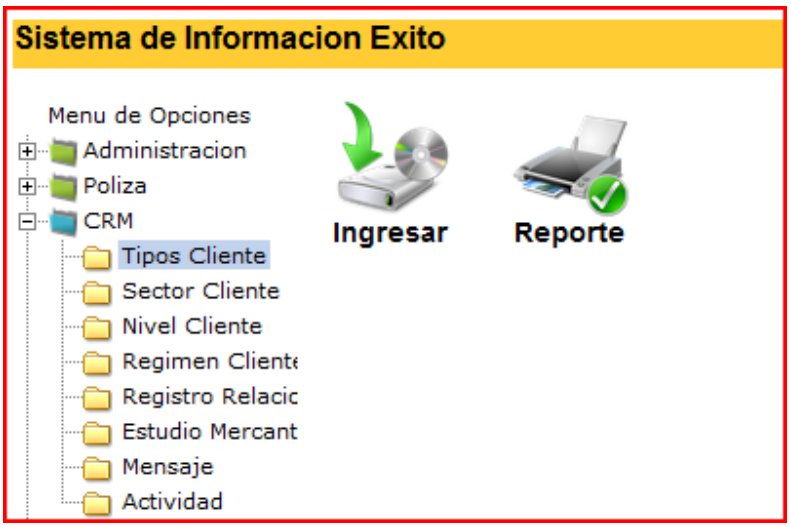
Registrar Póliza Bien

Registrar Póliza Cobro

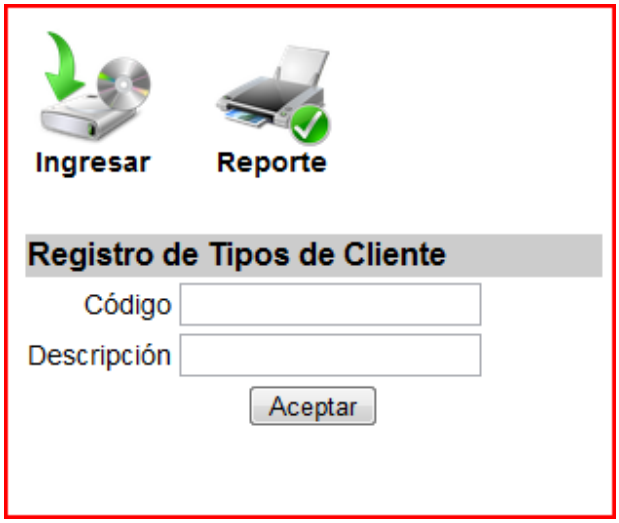


Registrar Póliza Cobro

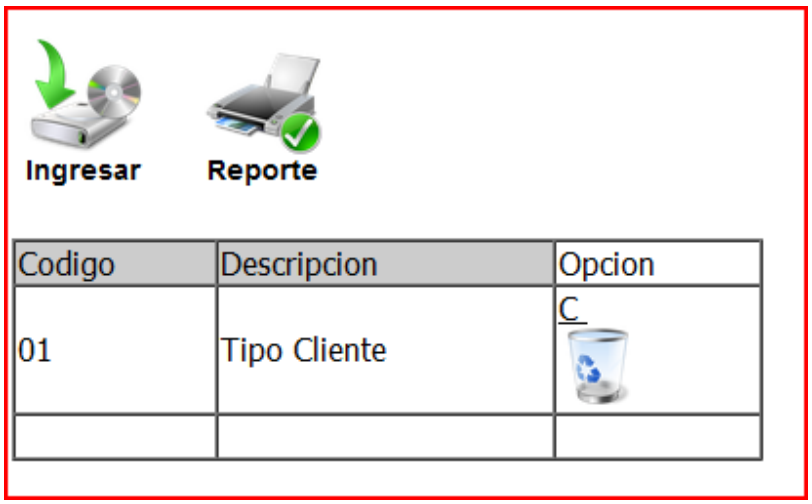
Menú Tipos cliente



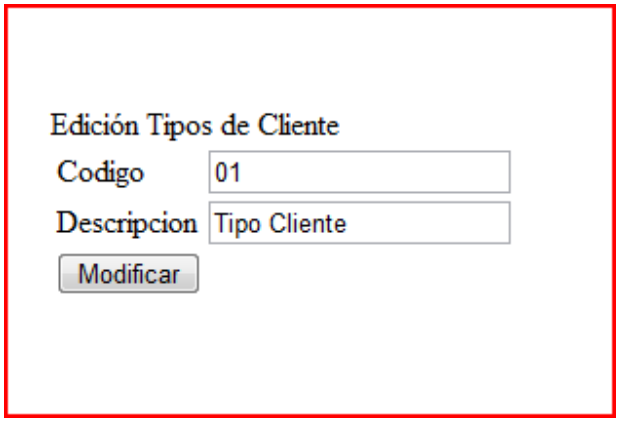
Ingresar Tipo cliente



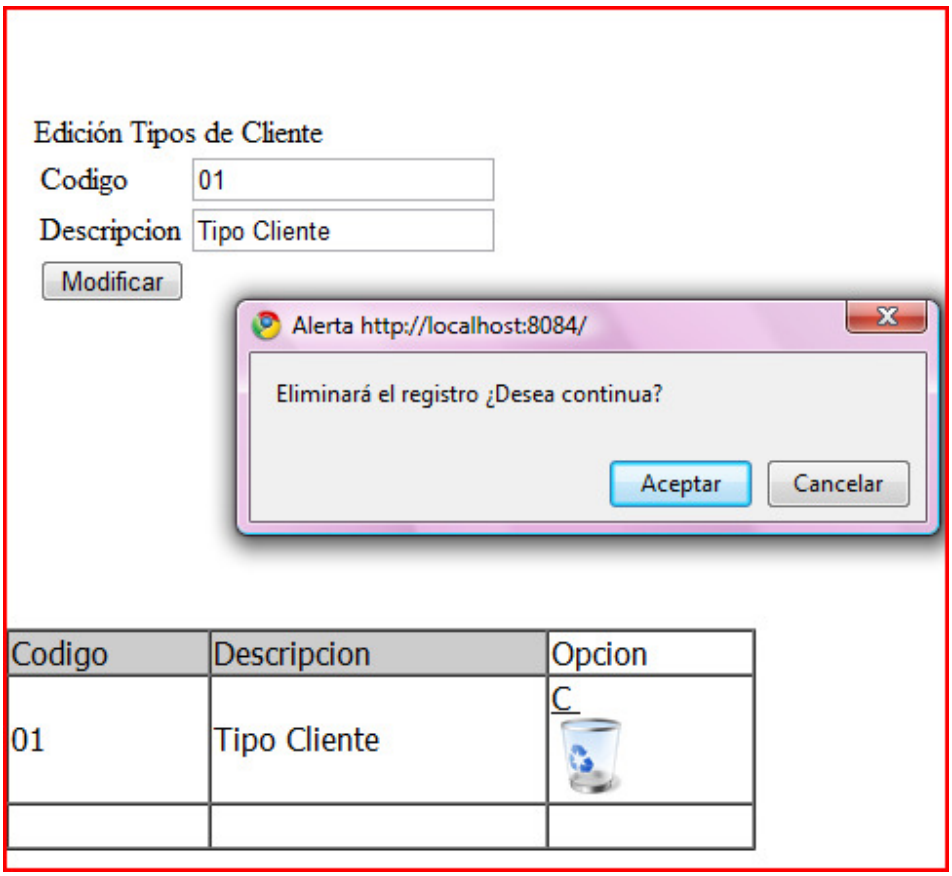
Listar Tipo Cliente



Editar Tipos cliente

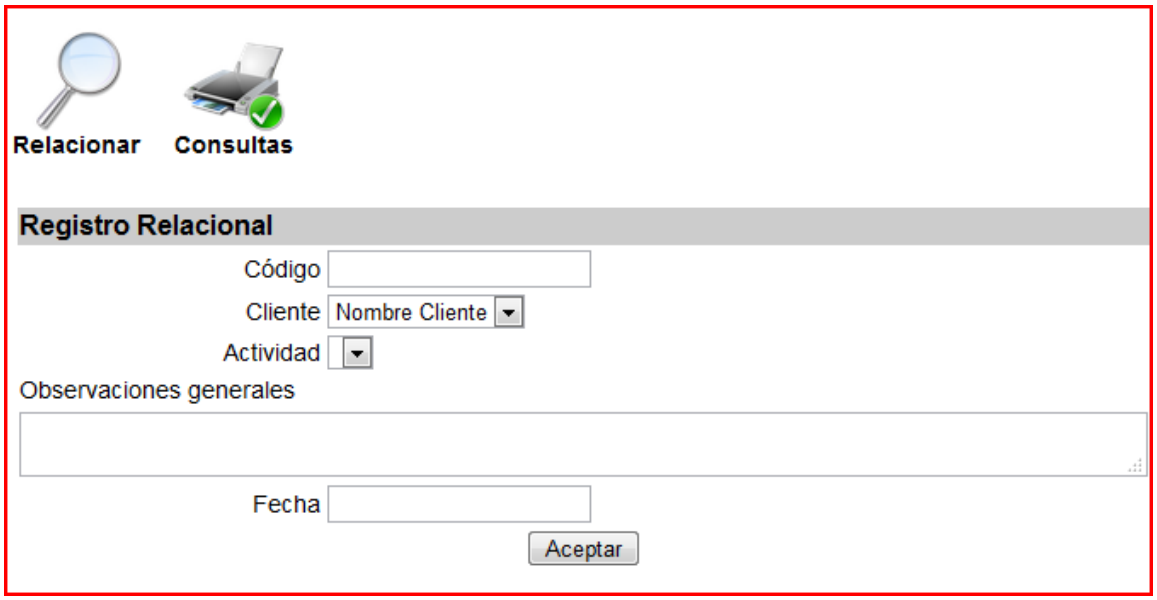


Eliminar Tipos Cliente



Menú Relación Clientes

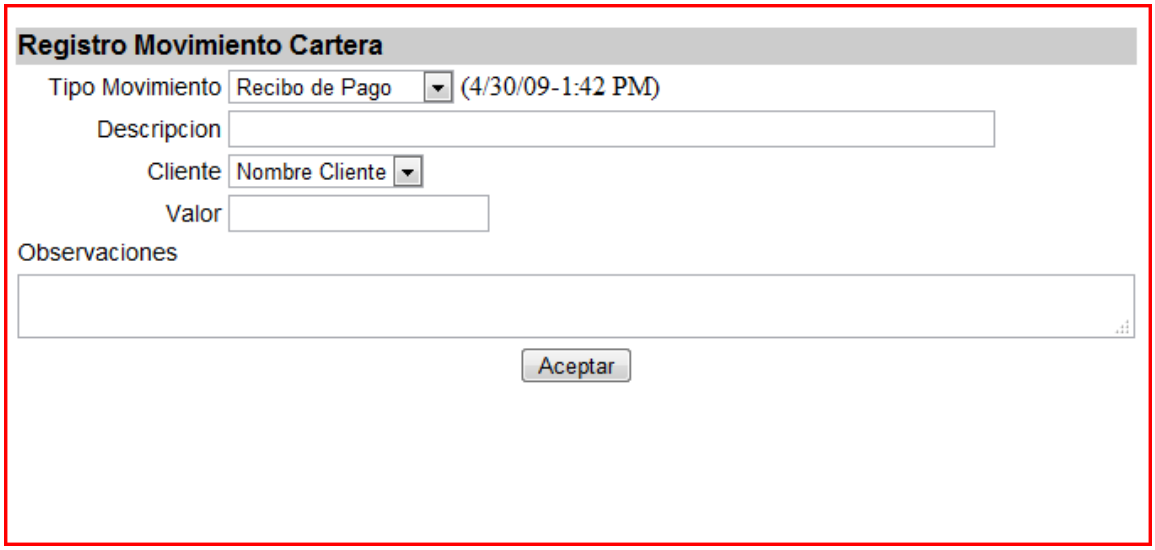
Registrar relación clientes



Registrar Estudio Mercantil

Registrar Actividad con Clientes

Registrar Movimiento de Cartera



4. Definición del esquema de replicación de datos

En el paquete postgresql-contrib-8.3 se encuentra una utilidad llamada *pgbench*. Esta utilidad permite, en primer lugar, inicializar una base de datos con una serie de tablas sencillas (observar el anexo de creación de la base de datos) y, en segundo lugar, realizar pruebas de rendimiento sobre servidores PostgreSQL mediante la ejecución de una cierta cantidad de consultas de varios tipos y con una concurrencia parametrizable.

Para lograr la definición y pruebas de replicación, es importante iniciar el servicio de PostgreSQL, actuando ya como cliente del servidor. Por eficiencia, es necesario tener en cuenta el archivo /etc/hosts. El primer paso se constituye en crear la base de datos bench_replication:

```
createdb -h pgsql1 -p 9999 -U pgpool2 bench_replication
createlang -h pgsql1 -p 9999 -U pgpool2 -d bench_replication plpgsql
```

Con log_statement y log_connections activados en /opt/pgpool2/etc/pgpool2.conf, ésto muestra las entradas en /var/log/syslog similares a las siguientes:

```
LOG: pid 4365: connection received: host=192.168.0.5 port=38024
LOG: pid 4365: statement: CREATE DATABASE bench_replication;
LOG: pid 4365: statement:  RESET ALL
LOG: pid 4365: statement:  SET SESSION AUTHORIZATION DEFAULT
```

Con log_statement = 'all' en /etc/postgresql/8.3/main/postgresql.conf, en el log de cualquiera de los PostgreSQL aparecen las siguientes líneas:

```
LOG:  connection received: host=192.168.0.3 port=33690
LOG:  connection authorized: user=pgpool2 database=postgres
LOG:  statement: CREATE DATABASE bench_replication;
LOG:  statement:  RESET ALL
LOG:  statement:  SET SESSION AUTHORIZATION DEFAULT
```

Como usuario postgres, en ambos nodos se usa psql para ver las bases de datos y verificar que se han creado:

```
$ su - postgres
$ psql -l
            List of databases
       Name         |  Owner   | Encoding
--------------------+----------+-----------
 bench_replication | pgpool2  | SQL_ASCII
 postgres          | postgres | SQL_ASCII
 secure            | postgres | SQL_ASCII
```

5. Elección del Hardware

La arquitectura del sistema vista controlador está basada en tres procesos o niveles separados. El primer nivel corresponde a la interfaz gráfica que ve el usuario en su computador, en el segundo nivel o nivel intermediario corre el servidor de aplicaciones que maneja la mayor parte de los procesamientos, y el tercer nivel almacena los datos en el servidor de base de datos. Aunque se trate de la configuración más recomendada, también es posible instalar la base de datos y el servidor de aplicaciones en el mismo servidor. La decisión final debería tomarse teniendo en cuenta la situación particular del presupuesto, como, por ejemplo, los tipos de usuario, el número de usuarios, la carga de trabajo de las máquinas, el uso previsto del sistema, etc.

Las recomendaciones que siguen están basadas en una configuración de servidores dedicados, es decir, con un servidor que alberga la base de datos y otro servidor que alberga el servidor de aplicaciones y los componentes del software desarrollado en el proyecto

Servidor De Base De Datos

La configuración del servidor está basada en un servidor Windows 2003 ó en un Linux (Debian), donde está instalado PostgreSQL, el cual funciona como servidor de datos y servidor de copias de seguridad (back-up). Es posible que se necesite

memoria adicional y espacio adicional en el disco duro si se instalan otros programas. Las especificaciones de hardware para servidores que utilizan otro sistema operativo de red deberían tener igual capacidad y rendimiento.

Unidad central de proceso
(CPU)1 Intel® Core 2 Duo, 2,6 GHz o Dual Core Xeon 2,6 GHz

Disco duro2 250 SATA 7,2K rpm 3 x 160 Gb SCSI, Ultra 320 SCSI o SATA, 15 000 rpm, con una configuración RAID 5

DVD-ROM 16 con cable SATA 48 DVD+/-RW

Memoria 1 Gb Hasta 12 Gb

Pantalla/tarjeta de video Pantalla plana (flat panel) de 17"

Dispositivo de cinta (streamer)

Servidor De Aplicaciones

El proyecto está escrito totalmente en Java™ y utiliza los servicios de servidores de aplicaciones Web compatibles con JSP. Por esta razón, no hay otros requisitos de hardware que los requisitos para el software de servidor de aplicaciones Web. La configuración del servidor descrita a continuación está basada en un servidor Linux que funciona como servidor de aplicaciones. Es posible que se necesite memoria adicional y espacio adicional en el disco duro si se instalan otros programas, como, por ejemplo una base de datos u otras aplicaciones que se ejecutan en el mismo servidor de aplicaciones. Las especificaciones de hardware para servidores que utilizan otro sistema operativo de red deberían tener igual capacidad y rendimiento.

6. Diseño de la conectividad

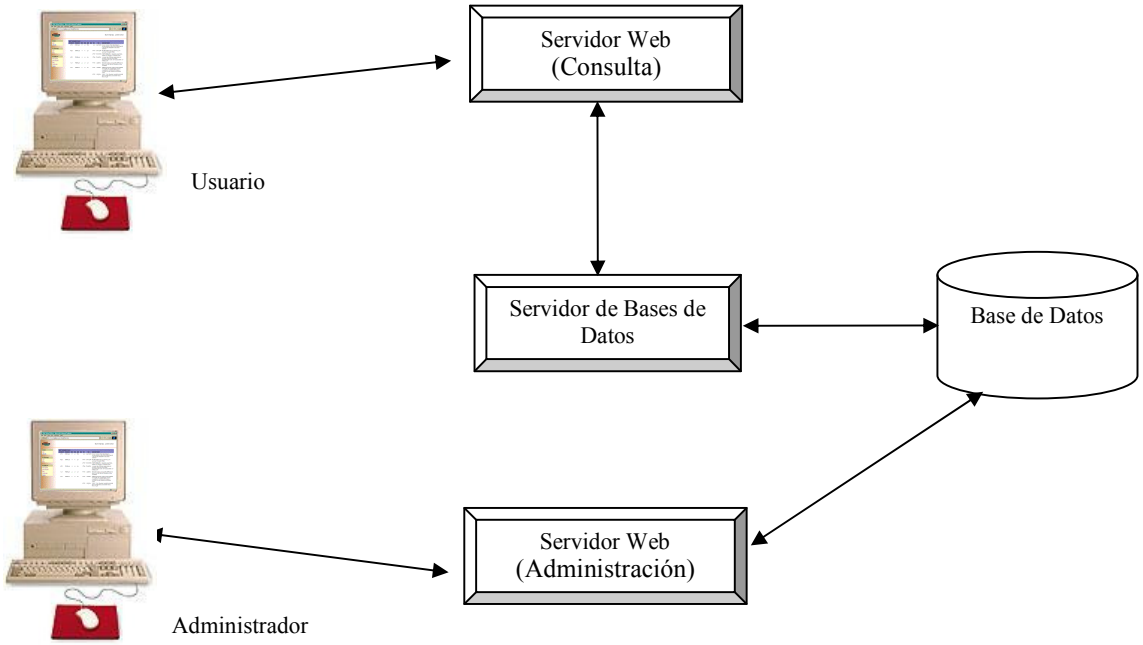
Con el advenimiento de programas basados en Navegadores para Internet, hay hoy en día un fenómeno llamado Intranet, el cual ha sido desarrollado por empresas y otras organizaciones privadas.

Una Intranet es una red privada que utiliza herramientas tipo Internet, pero disponible solamente dentro de la organización. Para grandes organizaciones, una Intranet provee a los empleados de un modo fácil de acceder a la información de la organización a través del mismo tipo de herramientas utilizadas para salir de la organización.

Como fundamento básico para el diseño de la conectividad para el proyecto, se tiene en cuenta Ethernet, es definida por su presencia, la más popular tecnología de soporte físico de LAN en uso hoy. Otros tipos de soporte físico de LAN incluyen Token Ring, Fast Ethernet, Interfaz de Datos Distribuida por Fibra (óptica) (Fiber Distributed Data Interface "FDDI"), Modo de Transferencia Asincrónico (Asynchronous Transfer Mode o "ATM") y LocalTalk.

Ethernet es popular porque logra un buen balance entre velocidad, costo y facilidad de instalación. Estos puntos fuertes, combinados con una amplia aceptación en el mercado informático y la habilidad para soportar virtualmente todos los protocolos populares de red, hacen de Ethernet una tecnología de red ideal para la mayoría de los usuarios de computadoras hoy en día.

En términos técnicos, se cuentan con usuarios generales y administradores, los cuales se ilustran con la figura abajo descrita.



7. Diseño de Objetos

7.1. Modularidad

Un sistema complejo puede dividirse en piezas más simples llamadas módulos, un sistema compuesto de módulos es llamado modular. El principal beneficio de la modularidad es que permite la aplicación del principio de separación de intereses en dos fases: al enfrentar los detalles de cada módulo por separado ignorando detalles de los otros módulos, y al enfrentar las características globales de todos los módulos y sus relaciones para integrarlos en un único sistema coherente. Si estas fases son ejecutadas en ese orden se dice que el sistema es diseñado de abajo hacia arriba (bottom up), en el orden inverso se dice que el sistema es diseñado de arriba hacia abajo (top down).

El principio de modularidad tiene tres (3) objetivos principales: capacidad de descomponer un sistema complejo, capacidad de componerlo a partir de módulos existentes y comprensión del sistema en piezas

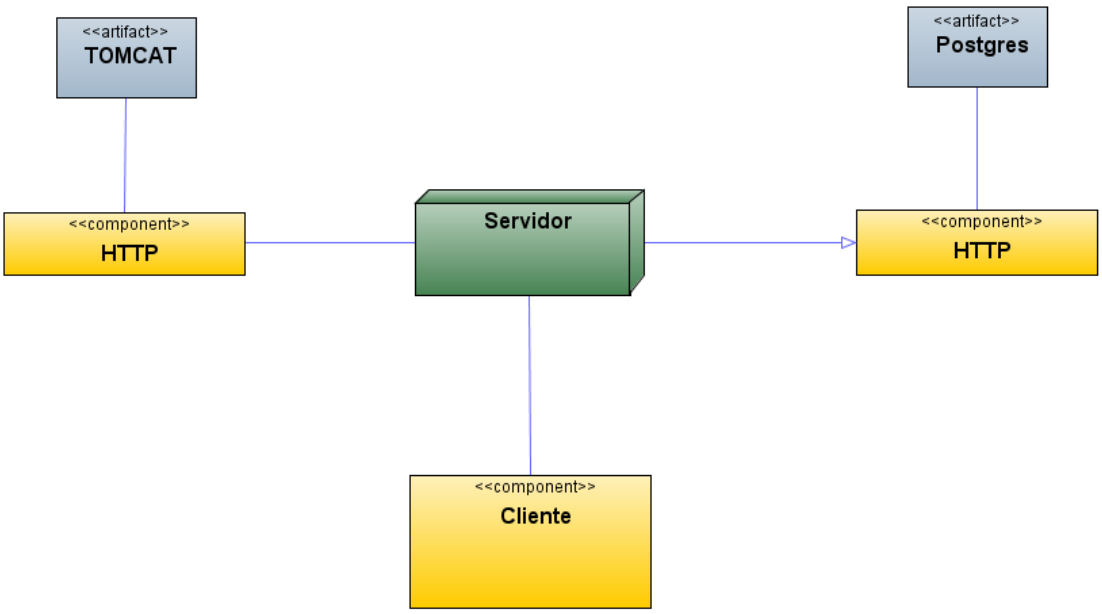
Como elemento básico trabajado en el proyecto, se tiene el Modelo Vista Controlador, que es un patrón de arquitectura de software cuya función es subdividir una aplicación en tres módulos que corresponden a la vista del usuario (la interfaz a la que accede el usuario), una lógica de control para captar los eventos que el usuario ha generado a través de la interfaz, y un modelo que gestiona los datos según le indique la lógica de control.

El modelo es la representación específica de la información con la cual el sistema opera y se compone por el Sistema de Gestión de Base de Datos y la lógica de negocio. La lógica de negocio asegura la integridad de estos y permite derivar nuevos datos.

La vista presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Por lo tanto, la vista es la encargada de presentar los datos al usuario y la interfaz necesaria para modificarlos. Un ejemplo de tecnología utilizada para el desarrollo del proyecto, la cual es JSP, ésta permite, mediante el servidor, genera HTML que interpreta el navegador del usuario mostrándole los datos y los formularios que constituyen la vista para que pueda interactuar con la aplicación.

El controlador responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. Por lo general, el controlador sería la unidad central que comunica la vista con el modelo y viceversa, asociando los eventos del usuario con los cambios que se producirán en el modelo y devolviendo los datos resultantes que genere el modelo a la vista que corresponda.

7.2. Diagrama de Despliegue – Arquitectura del sistema



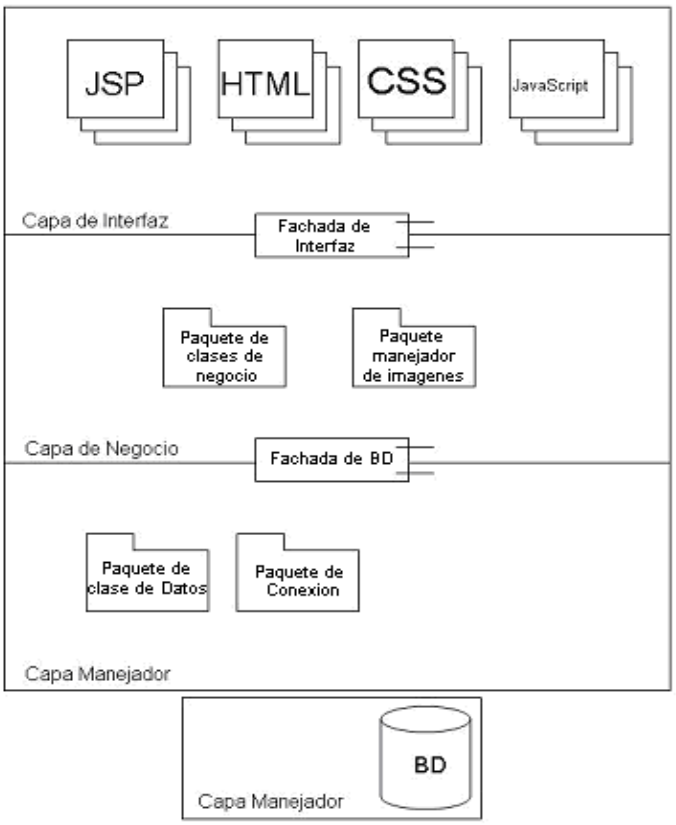
7.3. Diagrama de Componentes

Un diagrama de componentes muestra las dependencias lógicas entre componentes software, sean éstos componentes fuentes, binarios o ejecutables, ilustran las piezas del software, controladores embebidos, etc. Los diagramas de Componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema, es decir para describir la vista de implementación estática de un sistema. Los diagramas de componentes se relacionan con los diagramas de clases, ya que un componente normalmente se corresponde con una o más clases, interfaces o colaboraciones pero un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema.

Para todo sistema OO se han de construir una serie de diagramas que modelan tanto la parte estática, como Dinámica, pero llegado el momento todo esto se debe materializar en un sistema implementado. Los componentes software tienen tipo, que indica si son útiles en tiempo de compilación, enlace o ejecución. Se

consideran en este tipo de diagramas solo tipos de componentes. Instancias específicas se encuentran en el diagrama de ejecución.

Diagrama de componentes de la arquitectura del software.



Explicación

Las 3 capas que definen la arquitectura del sistema son: Interfaz, lógica y manejador.

Capa Interfaz: En esta capa se trabaja toda la primera conexión con el usuario, los archivos .JSP, Html, Css y Javascript. La comunicación entre esta capa y la capa lógica debe hacerse estrictamente a través del paquete de Interfaz.

Capa Lógica: En ésta capa se encuentran el Paquete de Clases de Negocio, Paquete Manejador de Imágenes (el cual, es el generador de las interfaces) y la Fachada de Interfaz que recibe las acciones requeridas por la capa interfaz para procesarlas y devolver una respuesta. El objetivo de esta capa es que en ella se concentre toda la lógica del sistema, de tal manera de independizar la relación del sistema con la interfaz. Es decir, está estrictamente prohibido la implementación de los Objetos del Negocio fuera de esta capa.

Capa Manejador: Esta capa se encarga del manejo, modificación y conexión de la base de datos. Contiene el paquete Clases de Datos, el paquete PKGCON y el paquete de BD (Interfaz de Base de Datos) que es la encargada de las peticiones de acceso a la base de datos desde la capa lógico. Todo acceso a la base de datos desde otras capas debe hacerse a través de este elemento.

7.4. Diseño y selección de Algoritmos

Para el desarrollo del proyecto, un algoritmo debe tener al menos las siguientes características:

Ser preciso. Esto significa que las operaciones o pasos del algoritmo deben desarrollarse en un orden estricto, ya que el desarrollo de cada paso debe obedecer a un orden lógico.

Ser definido. Ya que en el área de programación, el algoritmo se desarrolla como paso fundamental para desarrollar un programa, es necesario tener en cuenta que el computador solo desarrollará las tareas programadas y con los datos suministrados; es decir, no puede improvisar y tampoco se inventará o adivinará el dato que necesite para realizar un proceso. Por eso, el algoritmo debe estar plenamente definido; esto es, que cuantas veces se ejecute, el resultado depende estrictamente de los datos suministrados. Si se ejecuta con un mismo conjunto de datos de entrada, el resultado será siempre el mismo.

Ser finito: esta característica implica que el número de pasos de un algoritmo, por grande y complicado que sea el problema que soluciona, debe ser limitado. Todo algoritmo, sin importar el número de pasos que incluya, debe llegar a un final. Para hacer evidente esta característica, en la representación de un algoritmo siempre se incluyen los pasos inicio y fin.

Presentación formal: para que el algoritmo sea entendido por cualquier persona interesada es necesario que se exprese en alguna de las formas comúnmente aceptadas; pues, si se describe de cualquier forma puede no ser muy útil ya que solo lo entenderá quien lo diseñó. Las formas de presentación de algoritmos son: el pseudocódigo, diagrama de flujo y diagramas de Nassi/Schneiderman, entre otras.

Corrección: el algoritmo debe ser correcto, es decir debe satisfacer la necesidad o solucionar el problema para el cual fue diseñado. Para garantizar que el algoritmo logre el objetivo, es necesario ponerlo a prueba; a esto se le llama verificación o prueba de escritorio.

Eficiencia: hablar de eficiencia o complejidad de un algoritmo es evaluar los recursos de cómputo que requiere para almacenar datos y para ejecutar operaciones frente al beneficio que ofrece. En cuanto menos recursos requiere será más eficiente el algoritmo.

Selección

Para seleccionar un tipo de algoritmo, se estudiaron cuatro de las técnicas de diseño de algoritmos más conocidas: voraz, divide y vencerás, programación dinámica y backtracking. El fin principal es obtener un estilo de programación unificado al momento de iniciar el desarrollo del proyecto. A continuación de define en términos general cada uno de ellos.

Algoritmo voraz:

El algoritmo más sencillo, partiendo de un agregado solución vacío, recorrer el agregado de entrada, y añadir el elemento considerado en cada paso al agregado solución siempre que se cumplan las condiciones derivadas de la propiedad que se apuntó.

Algoritmo de programación dinámica:

Es a partir de la de la relación que establece cómo combinar las soluciones para distintas partes de los datos de entrada, establecer los casos en que puede obtenerse la solución sin utilizar dicha relación (casos base), y partiendo de dichos casos, guardar la solución para los mismos en una matriz de registros (multidimensional), y reconstruir la solución al problema utilizando los valores almacenados en la matriz de registros.

Algoritmo backtracking:

Interpretando al agregado como un conjunto, obtener todos los subconjuntos del conjunto (cuyo número es 2n), hasta encontrar uno que cumpla con las condiciones impuestas por el problema. Es decir, se trata de realizar una búsqueda exhaustiva. Llamamos árbol de expansión al árbol formado por todas las posibilidades que se estudian. En cada hoja hay una posible solución.

Algoritmo divide y vencerás:

Consiste en dividir el agregado en el número de partes que queramos, siempre que el tamaño sea superior a un tamaño umbral que admitamos. Obtener la solución de cada parte y combinar dichas soluciones para construir la solución al agregado completo. Para el caso de que el tamaño sea inferior o igual a umbral, la

solución se obtiene mediante el algoritmo anterior (o cualquier otro conocido que no divida el agregado en partes).

El algoritmo seleccionado es divide y vencerás, todo debido a su simplicidad y gran capacidad para adoptar los problemas en programación.

7.5. Medición de algoritmos

Las mediciones del mundo físico se pueden categorizar de dos maneras: directas (por ejemplo la longitud) e indirectas (por ejemplo la calidad), las métricas de software se pueden categorizar de forma similar.

Entre las medidas directas del proceso se incluyen el coste y el esfuerzo aplicados. Entre las medidas directas del producto se incluyen las líneas de código producidas, velocidad de ejecución, tamaño de memoria, y los defectos informados durante un periodo de tiempo establecido.

Sin embargo, la calidad y la funcionalidad del software o su eficiencia o mantenimiento son más difíciles de evaluar y solo pueden ser medidos indirectamente. Para el desarrollo del proyecto, el modelo intermedio será utilizado para lograr una medición, dado que realiza las estimaciones con bastante precisión. Así pues las fórmulas serán las siguientes:

E = Esfuerzo = a KLDC e * FAE (persona x mes)
T = Tiempo de duración del desarrollo = c Esfuerzo d (meses)
P= Personal = E/T (personas)

Para calcular el Esfuerzo, es necesario hallar la variable KDLC (Kilo-líneas de código), donde los PF son 261,36 (dato conocido) y las líneas por cada PF equivalen a 32 según vemos en la tabla que se ilustra a continuación:

Según estándares, se toma como base la siguiente tabla.

LENGUAJE	LDC/PF
Ensamblador	320
C	150
COBOL	105
Pascal	91

PHP	36
ASP	38
JSP	32
SQL	12

Luego de definir 32 LDC por cada PF, por el hecho de ser JSP el resultado de los KDLC será el siguiente:

KLDC= (PF * Líneas de código por cada PF)/1000 = (261,36*32)/1000= 8,363 KDLC

El tipo orgánico será el más apropiado ya que el número de líneas de código no supera los 50 KLDC, por consiguiente, los coeficientes que usaremos serán las siguientes:

PROYECTO SOFTWARE	Menú	CRUD	Control	Presentación
Orgánico	3,2	1,05	2,5	0,38
Semi-acoplado	3,0	1,12	2,5	0,35
Empotrado	2,8	1,20	2,5	0,32

Luego se obtiene la variable FAE, la cual se obtiene mediante la multiplicación de los valores evaluados en los diferentes 15 conductores de coste que se observan en la siguiente tabla:

CONDUCTORES DE COSTE	VALORACIÓN					
	Muy bajo	Bajo	Nominal	Alto	Muy alto	Extr. alto
Fiabilidad requerida del software	0,90	0,88	0,95	**1,15**	1,40	-
Tamaño de la base de datos	-	0,88	**0,95**	1,08	1,16	-

Complejidad del producto	0,45	**0,85**	0,95	1,15	1,30	1,65
Restricciones del tiempo de ejecución	-	-	0,95	**1,11**	1,30	1,66
Restricciones del almacenamiento princip	-	-	**0,95**	1,06	1,21	1,56
Volatilidad de la máquina virtual (Java)	-	0,87	**0,95**	1,15	1,30	-
Tiempo de respuesta del servidor	-	0,87	0,95	**1,07**	1,15	-
Capacidad del analista	1,2	1,06	0,95	**0,86**	0,71	-
Experiencia en la aplicación	1,04	1,05	0,95	1,00	**0,82**	-
Capacidad de los programadores	1,42	1,17	0,95	0,86	**0,45**	-
Experiencia en S.O. utilizado	1,21	1,10	**0,95**	0,90	-	-
Experiencia en el lenguaje de programaci	1,14	1,07	0,95	**0,95**	-	-
Prácticas de programación modernas	1,24	1,10	**0,95**	1,00	0,82	-
Utilización de herramientas software	1,24	1,10	0,95	**1,00**	0,83	-
Limitaciones de planificación del proyecto	1,23	**1,08**	0,95	1,04	1,10	-

FAE=1,15*1,00*0,85*1,11*1,00*1,00*1,07*0,86*0,82*0,45*1,00*0,95*1,00*1,00*1,08 = 0,53508480

Cálculo del esfuerzo del desarrollo:

E = a KLDC e * FAE = 3,2 * (1.363) ^ 1,05 * 0,53508480 = 1,91 personas /mes

Cálculo tiempo de desarrollo:

T = c Esfuerzo d = 2,5 * (15,91) ^ 0,38 = 7,15 meses

Productividad:

PR = LDC/Esfuerzo = 8363/15,91 = 525 ,64 LDC/personas mes

Personal promedio:

P = E/T = 15,91/7,15 = 2,22 personas

Analizando los resultados, será necesario un equipo de 2 personas (Las cuales serán representadas por el proponente del presente proyecto) trabajando alrededor de 7 meses, pero puesto que el desarrollo del proyecto debe realizarse en un plazo 6 meses (Según el planteamiento del cronograma inicial), se debe incrementar el esfuerzo,

7.6. Permisos y Control de Acceso a usuarios

El control de acceso a las distintas operaciones provistas por un sistema es un tema crítico para el sistema mismo, y para un proyecto orientado a la Web, estos aspectos se hacen más relevantes.

El mecanismo establecido para tal fin está basado en un esquema de usuarios y perfiles de usuarios. Los cuales ingresan al sistema desde la página principal. En este esquema, cada usuario tiene una identificación personal (login) con la cual se identifica ante el sistema, y una clave que sólo él debe conocer, con la cual acredita ser el dueño de dicha identificación.

Por otro lado, el usuario debe estar asignado a un perfil, el cual determina las opciones y operaciones que el sistema le permitirá acceder y efectuar. Naturalmente, los usuarios y perfiles deben ser administrados, y los procesos que involucra esta administración son considerados críticos, razón por la cual no pueden ser llevados a cabo por cualquiera ni de cualquier modo.

En los procesos de administración de usuarios y perfiles intervienen distintos actores: los usuarios, los jefes de los usuarios (entendiéndose por tal la cadena de mandos), los consultores y el personal de ventas de seguros.

Los usuarios son los operadores del sistema, quienes ejecutan las operaciones objeto del mismo. Los jefes directos de los usuarios son quienes asignan las tareas que deben realizar los empleados que están bajo sus órdenes y por tanto son los que definen cuáles son los usuarios que deben tener acceso a los sistemas y con qué perfil deben hacerlo. Los mandos superiores, son los que definen la estructura de los encargados de cada organismo.

Los perfiles definidos, los cuales están en capacidad de crearse en el sistema son:

Administrador: Es la persona autorizada por el Representante Legal de la Empresa, quien se encarga de administrar la aplicación, crear los usuarios necesarios, asignar perfiles de usuario y definir los producto, entre otros. El usuario administrador puede ejecutar transacciones.

Asesor: Es la persona asignada por el administrador para preparar los procesos de las diferentes transacciones,

Consultor: Es la persona asignada por el administrador con perfil de consulta únicamente. El administrador puede autorizar la consulta de uno o varios productos de la empresa.

7.7. Seguridad

La empresa debe designar, siempre que la estructura lo permita, un responsable de la seguridad de los sistemas de información y sus recursos asociados, en forma separada a las áreas de servicios informáticos.

Caso contrario recaerá en las áreas de servicios informáticos la función y responsabilidad asociada, lo que no exime a los responsables de los sistemas de las funciones de control que le compiten.

Para ambos casos corresponde las siguientes funciones y responsabilidades:

- Informar al responsable de los sistemas de información de su organismo, del programa de seguridad sobre los recursos tecnológicos relacionados y el cumplimiento de las políticas, normas, procedimientos y estándares, en vigencia.

- Informar al responsable de los sistemas de información, sobre violaciones o incidentes relacionados con la seguridad de los sistemas de información y los recursos tecnológicos relacionados.

- Asistir a los responsables de los sistemas de información en el análisis de riesgos, clasificación de grados de criticidad, permisos de accesos y permisos de usos (creación, borrado, lectura, escritura, etc.) de los mismos.

- Asistir a los técnicos de las áreas de servicios informáticos en la gestión de creación, modificación y baja de usuarios, perfiles, contraseñas y accesos a los sistemas.

Modelos de Seguridad en Java.

La seguridad se basa en tres componentes fundamentales del entorno de ejecución:

El cargador de clases, que determina cómo y cuándo pueden cargar código los programas y garantiza que los componentes del sistema no han sido reemplazados.

El verificador de archivos de clases, que garantiza que el código tiene el formato correcto, que el bytecode no viola las restricciones de seguridad de tipos de la JVM (Máquina Virtual de Java), que las pilas internas no puedan desbordarse ni por arriba ni por abajo y que las instrucciones en bytecode tengan parámetros de tipos correctos. Los bytecodes son en realidad código máquina escrito para el juego de instrucciones de la JVM, por lo que el proceso de interpretación es más rápido que el de otros lenguajes.

El gestor de seguridad, que controla el acceso a los recursos en tiempo de ejecución, como E/S de red y ficheros, creación de cargadores de clases, manipulación de hilos de ejecución, ejecución de programas externos, detener la JVM, cargar código nativo en la máquina virtual, realizar determinadas

operaciones en el entorno de ventanas o cargar ciertos tipos de clases. Un aspecto muy importante del gestor de seguridad es que una vez cargado no se puede reemplazar. Hay que señalar que el gestor de seguridad está muy relacionado con la clase AccessControler.

Conceptualmente un dominio incluye un conjunto de clases cuyas instancias tienen asignados los mismos permisos. Los dominios de protección se determinan mediante la política de seguridad activa en cada momento. Los dominios protegidos se dividen generalmente en dos categorías:

Dominios del sistema, que controlan el acceso a los recursos del sistema, tales como el sistema de archivos, acceso a la red, E/S.

Dominios de aplicación, que controlan el acceso a los recursos de una aplicación.

Modelo de permisos

Los permisos en Java son clases que representan accesos a recursos del sistema. La clase fundamental es *java.security.Permission*, que es una clase abstracta de la que se deben definir subclases para representar accesos específicos.

Generalmente, una clase de permiso pertenece al paquete en el cual será usada. Por ejemplo, el permiso que representa el acceso al sistema de ficheros local es java.io.FilePermission.

ANEXO D
PRUEBAS

CONTENIDO

1. Introduccion

Este proceso, generalmente se divide en dos en el patrón: el camino principal o secuencia normal y los caminos alternativos o excepciones. El camino principal son los pasos que da el sistema si no surge ningún imprevisto ni error, mientras que los caminos alternativos son las variaciones que pueden surgir en distintos puntos del camino principal a causa de errores, rectificaciones, etc. A cada uno de estos caminos lo llamaremos escenario de caso de uso. Todos los escenarios de caso de uso posibles serán utilizados como base para crear las pruebas.

1.1. Ámbito

Este Plan de Pruebas describe las pruebas de unidad, integración y del sistema que se aplicarán al sistema software desarrollado. El objetivo es probar todos los elementos del producto.

2. Definición de las Pruebas

La lista que proporcionamos en éste anexo identifica los elementos (casos de uso, requisitos funcionales y requisitos no funcionales) que son objetivos de las pruebas. Es decir, los elementos qué vamos a probar.

2.1. Pruebas de integridad de la base de datos y de los datos:

- Verificar el acceso al sistema (pagína principal).
- Verificar la recuperación correcta de las operaciones realizadas en la base de datos (CRUD*).
- Verificar accesos simultáneos de lectura de datos.

2.2. Pruebas de funcionalidad:

- Verificar el caso de uso: Pólizas para Agentes (Caso-01).
- Verificar el caso de uso Pólizas para Amparos (Caso-02).
- Verificar el caso de uso Pólizas para Artículos (Caso-03).
- Verificar el caso de uso Pólizas para Automóviles (Caso-04).
- Verificar el caso de uso Pólizas para Bienes(Caso-05).
- Verificar el caso de uso Pólizas para Cobros (Caso-06).
- Verificar el caso de uso Pólizas para Vida (Caso-07).

2.3. Pruebas de interfaz de usuario:

- Verificar que la navegación a través de un conjunto de pantallas es fácil.

* Normalmente definidas como

- Verificar todas las funciones.
- Verificar que todas las interfaces de usuario siguen los estándares de GUI.

2.4. Pruebas de desarrollo:

En esta sección presentamos el enfoque que vamos a utilizar para probar el sistema software. En la sección anterior hemos descrito qué elementos del sistema software vamos a probar, y en esta sección se define cómo se realizaran las pruebas.

2.4.1. Pruebas de integridad de la base de datos y de los datos.

Objetivos de la prueba	Comprobar que los procedimientos y métodos de acceso a la base de datos funcionan correctamente.
Técnicas	Invocar cada procedimiento o método de acceso a la base de datos con datos válidos e inválidos. Inspeccionar la base de datos para asegurar que los datos son los previstos, todos los eventos de la base de datos ocurren adecuadamente, o revisar los valores devueltos para asegurar que la recuperación de datos es correcta.
Criterios de finalización	Todos los procedimientos y métodos de acceso funcionan como se diseñaron y sin ningún error en los datos.
Consideraciones	Los procesos se deberían invocar manualmente. Se debería usar bases de datos de tamaño pequeño o mínimo (limitado según el número de registros) para incrementar la visibilidad de cualquier evento no aceptable. Las pruebas pueden necesitar un entorno de desarrollo DBMS para recuperar o modificar datos directamente de la base de datos.

2.4.2. Pruebas de funcionalidad.

Las pruebas de funcionalidad se deberían centrar en cualquier requisito que pueda ser trazado directamente de los casos de uso y reglas de negocio. El objetivo de estas pruebas es verificar la aceptación, procesamiento y recuperación de datos y la adecuada implementación de las reglas de negocio. Este tipo de pruebas están basadas en técnicas de caja negra, es decir, verificar la aplicación interaccionando a través de las interfaces de usuario y analizando los resultados.

Objetivos de la prueba	Asegurar la navegación correcta de la aplicación, la entrada de datos, su procesamiento y recuperación.
Técnicas	Ejecutar cada caso de uso y flujo del caso de uso con datos válidos e inválidos para verificar lo siguiente: • Cuando se utilizan datos correctos se obtienen los resultados esperados. • Cuando se utilizan datos incorrectos se obtienen los mensajes de error o advertencias adecuadas. • Cada regla de negocio se ha aplicado correctamente.
Criterios de finalización	Todas las pruebas planificadas se han ejecutado. Todos los defectos identificados se han considerado.
Consideraciones	Ninguna.

2.4.3. Pruebas de interfaz de usuario.

Las pruebas de interfaz de usuario verifican la interacción del usuario con el sistema software. El objetivo de esta prueba es asegurar que la interfaz de usuario permite al usuario acceder y navegar a través de toda la funcionalidad de la aplicación. Además, la prueba de interfaz de usuario garantiza que las interfaces de usuario cumplen los estándares.

Objetivos de la prueba	Verificar los siguientes objetivos: ● La navegación a través de la aplicación refleja adecuadamente las reglas de negocio y los requisitos incluyendo ventana a ventana, campo a campo y métodos de acceso (tabulador, movimientos del ratón y teclas de función). ● Las ventanas y sus características, como menús, tamaño, posición y estado cumplen los estándares.
Técnicas	Crear o modificar pruebas para cada ventana con el objetivo de verificar la correcta navegación y su estado.
Criterios de finalización	Cada ventana se ha verificado con éxito y es consistente con la versión de referencia o con los estándares utilizados.

Consideraciones	Ninguna.

2.4.4. Pruebas de desarrollo.

Las pruebas de desarrollo miden tiempos de respuesta, índices de transacción y otros requisitos susceptibles al tiempo. El objetivo de estas pruebas es verificar y validar que los requisitos de rendimiento se han alcanzado.
Las pruebas de desarrollo normalmente se ejecutan varias veces usando cada vez un cargo de trabajo diferente. La prueba inicial se debería realizar con una carga normal y la segunda prueba con una carga extrema.

Objetivos de la prueba	Validar el tiempo de respuesta del sistema software para las transacciones diseñadas o funciones de negocio bajo las condiciones siguientes: • Volumen de trabajo normal. • El peor volumen de trabajo.
Técnicas	Usar los procedimientos de prueba definidas para las pruebas de funcionalidad. Modificar los ficheros de datos (para incrementar el número de transacciones) o modificar los sripts para incrementar el número de iteraciones que se ejecutan en cada transición.
Criterios de finalización	Se han completado las pruebas sin ningún error y dentro de los tiempos de respuesta esperados.
Consideraciones	Ninguna.

2.5. Herramientas

Las siguientes herramientas se usarán para llevar a cabo el proceso de prueba:

Tipo de Prueba	Herramienta
Gestión del proyecto	Microsoft Project
Herramienta DBMS	Postgres
Interfaz de usuario	HTML
Funcionales	Junit (Netbeans)
Rendimiento	Aceptable

2.6. Recursos

En esta sección describimos los recursos necesarios para realizar el proceso de prueba, sus principales responsabilidades y características.

2.6.1. Recursos hardware

Recurso	Cantidad	Nombre y Tipo
PC	2	Diseño de las pruebas
PC	3	Ejecución de las pruebas

2.6.2. Recursos software

Nombre del elemento software	Tipo y otras notas
Magic Draw	Desarrollo del proyecto
Microsoft Project	Gestión del proyecto
PostgreSQL	Herramienta DBMS
Test Complete	Interfaz de usuario
JUnit Optimize it	Funcionales
Test Load	Rendimiento

2.6.3. Herramientas de soporte

Ninguna.

2.6.4. Configuración del entorno de prueba

Ninguna.

2.6.5. Recursos humanos

RECURSOS HUMANOS		
Rol	**Mínimos recursos recomendados**	**Responsabilidades específicas o comentarios**
Gestor de prueba	1	Proporcionar una gestión adecuada. Responsabilidades: • Proporcionar una dirección técnica. • Adquirir los recursos apropiados. • Informar de la gestión.

Diseñador de prueba	3	Identificar, priorizare implementar los casos de prueba. Responsabilidades: • Generar el Plan de pruebas. • Diseñar los Casos de prueba. • Evaluar el esfuerzo de prueba.
Probador (Tester)	3	Ejecutar las pruebas. Responsabilidades: 1. Ejecutar pruebas. 2. Recuperar los errores. 3. Documentar los defectos.

3. Actividades de prueba

Las actividades del proceso de prueba para este sistema software son:

Actividad	Esfuerzo (p/d)	Fecha de comienzo	Fecha de finalización
Planificación de la prueba	2	12 de Marzo	15 de Marzo
Diseño de la prueba	3	12 de Abril	20 de Abril
Implementación de la prueba	4	2 de Mayo	20 de Mayo
Ejecución de la prueba	3	1 de Junio	15 de Junio
Evaluación de la prueba	1	3 de Junio	17 de Junio

4. Resultados de las pruebas

Del proceso de prueba se obtienen los siguientes documentos de desarrollo de software:

Documento de desarrollo de software	Desarrollador	Revisión	Fecha
Plan de prueba	LSI	LSI	20 de Marzo
Casos de prueba	LSI	LSI	25 de Mayo
Informe de evaluación de pruebas	LSI	LSI	20 de Junio
Modelo de prueba	LSI	LSI	16 de Junio

5. Tareas de la etapa de pruebas

Las tareas que se realizan en cada una de las actividades son:

- **Planificación de las pruebas:**

    - Identificar los requisitos para las pruebas.
    - Valorar los riesgos.
    - Desarrollar la estrategia de pruebas.
    - Identificar los recursos necesarios para realizar las pruebas.
    - Planificar la temporalización.
    - Generar el Plan de pruebas.

- **Diseño de las pruebas:**

    - Análisis de la carga de trabajo.
    - Desarrollo de las pruebas.
    - Identificar y describir los casos de prueba.

- **Implementación de las pruebas:**

    - Establecer el entorno de prueba.
    - Desarrollar las clases de prueba, los componentes de prueba y los datos de prueba.

- **Ejecución de las pruebas:**

    - Ejecutar los casos de prueba.
    - Evaluar la ejecución del proceso de prueba.
    - Verificar los resultados.
    - Investigar los resultados no esperados.
    - Registrar los defectos.

- **Evaluación de las pruebas:**

    - Evaluar la cobertura de los casos de prueba.
    - Evaluar la cobertura del código.
    - Analizar los defectos.
    - Determinar si se han alcanzado los criterios de las pruebas.
    - Crear los informes de evaluación de las pruebas.

ANEXO F
Script Base de Datos

```
DROP DATABASE secure;

CREATE DATABASE secure
WITH OWNER = postgres
ENCODING = 'utf8';

\c secure;

CREATE TABLE tblTipoCli
(
idtipocli character varying NOT NULL,
descripcion character varying,
estado varchar(1),
CONSTRAINT keyTipoCli PRIMARY KEY (idtipocli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblTipoCli OWNER TO postgres;
INSERT INTO tblTipoCli VALUES ('01', 'Tipo Cliente');

CREATE TABLE tblSectorCli
(
idSectorCli character varying NOT NULL,
descripcion character varying,
estado varchar(1),
CONSTRAINT keySectorCli PRIMARY KEY (idSectorCli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblSectorCli OWNER TO postgres;
INSERT INTO tblSectorCli VALUES ('01', 'Ambiente, vivienda y desarrollo territorial');
INSERT INTO tblSectorCli VALUES ('02', 'Agricultura y desarrollo rural');
INSERT INTO tblSectorCli VALUES ('03', 'Autónomo e independiente');
INSERT INTO tblSectorCli VALUES ('04', 'Comercio, industria y turismo');
INSERT INTO tblSectorCli VALUES ('05', 'Comunicaciones');
INSERT INTO tblSectorCli VALUES ('06', 'Congreso');
INSERT INTO tblSectorCli VALUES ('07', 'Control');
INSERT INTO tblSectorCli VALUES ('08', 'Cultura');
INSERT INTO tblSectorCli VALUES ('09', 'Defensa');
INSERT INTO tblSectorCli VALUES ('11', 'Economía solidaria');
INSERT INTO tblSectorCli VALUES ('12', 'Educación');
INSERT INTO tblSectorCli VALUES ('13', 'Estadística');
INSERT INTO tblSectorCli VALUES ('14', 'Función pública');
INSERT INTO tblSectorCli VALUES ('15', 'Hacienda y crédito público');
INSERT INTO tblSectorCli VALUES ('16', 'Interior y justicia');
INSERT INTO tblSectorCli VALUES ('17', 'Minas y energía');
INSERT INTO tblSectorCli VALUES ('18', 'Organización electoral');
INSERT INTO tblSectorCli VALUES ('19', 'Planeación');
INSERT INTO tblSectorCli VALUES ('20', 'Poder judicial');
INSERT INTO tblSectorCli VALUES ('21', 'Presidencia');
INSERT INTO tblSectorCli VALUES ('22', 'Protección social');
INSERT INTO tblSectorCli VALUES ('23', 'Relaciones exteriores');
INSERT INTO tblSectorCli VALUES ('24', 'Transporte');
INSERT INTO tblSectorCli VALUES ('25', 'Seguridad');
```

```
CREATE TABLE tblNivelCli
(
idNivelCli character varying NOT NULL,
descripcion character varying,
estado varchar(1),
CONSTRAINT keyNivelCli PRIMARY KEY (idNivelCli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblNivelCli OWNER TO postgres;
INSERT INTO tblNivelCli VALUES ('01', 'Descentralizado');
INSERT INTO tblNivelCli VALUES ('02', 'Corporación de investigación');
INSERT INTO tblNivelCli VALUES ('03', 'Central');
INSERT INTO tblNivelCli VALUES ('04', 'Organismo y entidad regional especial');
INSERT INTO tblNivelCli VALUES ('05', 'Esp. Mixta');

CREATE TABLE tblRegimenCli
(
idRegimenCli character varying NOT NULL,
descripcion character varying,
estado varchar(1),
CONSTRAINT keyRegimenCli PRIMARY KEY (idRegimenCli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblRegimenCli OWNER TO postgres;
INSERT INTO tblRegimenCli VALUES ('01', 'Persona natural no retenedora');
INSERT INTO tblRegimenCli VALUES ('02', 'Persona natural retenedora');
INSERT INTO tblRegimenCli VALUES ('03', 'Persona jurídica contribuyente');
INSERT INTO tblRegimenCli VALUES ('04', 'Persona jurídica no contribuyente');
INSERT INTO tblRegimenCli VALUES ('05', 'Gran contribuyente');
INSERT INTO tblRegimenCli VALUES ('06', 'contribuyente');
INSERT INTO tblRegimenCli VALUES ('07', 'Gran contribuyente no responsable');
INSERT INTO tblRegimenCli VALUES ('08', 'Persona jurídica régimen especial');
INSERT INTO tblRegimenCli VALUES ('09', 'Persona jurídica empresa unipersonal');

CREATE TABLE tblEstudioMercadoCli
(
idEstudioMercadoCli character varying NOT NULL,
FechaInicio varchar,
FechaFin varchar,
FacDetCli varchar, -- Factores determinantes para la adquisición del producto por parte del cliente
FacDetPro varchar, -- Factores determinantes para la adquisición del producto según sus características
ConceptoEntornoExterno varchar,
ConceptoEntornoInterno varchar,
ResultadoAnalisis character varying,
estado varchar(1),
CONSTRAINT keyEstudioMercadoCli PRIMARY KEY (idEstudioMercadoCli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblEstudioMercadoCli OWNER TO postgres;

CREATE TABLE tblRelacionCli
(
idRelacionCli character varying NOT NULL,
idCliente varchar,
idActividadCli varchar,
FechaRegistro varchar,
Observacion varchar,
estado varchar(1),
CONSTRAINT keyRelacionCli PRIMARY KEY (idRelacionCli, idCliente)
```

```
)
WITH (OIDS=FALSE);
ALTER TABLE tblRelacionCli OWNER TO postgres;

CREATE TABLE tblMensajeCli
(
idMensajeCli character varying NOT NULL,
idCliente varchar,
Asunto varchar,
TextoMensaje varchar,
FechaMensaje varchar,
estado varchar(1),
CONSTRAINT keyMensajeCli PRIMARY KEY (idMensajeCli, idCliente)
)
WITH (OIDS=FALSE);
ALTER TABLE tblRelacionCli OWNER TO postgres;

CREATE TABLE tblActividadCli
(
idActividadCli character varying NOT NULL,
FechaInicio varchar,
FechaFin varchar,
Descripcion varchar,
estado varchar(1),
CONSTRAINT keyActividadCli PRIMARY KEY (idActividadCli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblActividadCli OWNER TO postgres;

CREATE TABLE tblCampanaCli
(
idCampanaCli character varying NOT NULL,
Director varchar,
ProductoRelacion varchar,
idTipoCampana varchar,
Objetivo varchar,
Descripcion varchar,
FechaInicio varchar,
FechaFin varchar,
Presupuesto varchar,
estado varchar(1),
CONSTRAINT keyCampanaCli PRIMARY KEY (idCampanaCli)
)
WITH (OIDS=FALSE);
ALTER TABLE tblActividadCli OWNER TO postgres;

CREATE TABLE tblActividad
(
  idActividad character varying NOT NULL,
  idModulo character varying NOT NULL,
  Descripcion character varying,
  url character varying,
  target character varying,
  estado varchar(1),
  CONSTRAINT keyActividad PRIMARY KEY (idActividad)
)
WITHOUT OIDS;
ALTER TABLE tblActividad OWNER TO postgres;

INSERT INTO tblActividad VALUES ('0101', '01', 'Abogado', 'faces/mAbogado.jsp','area' ,'a');
```

```
INSERT INTO tblActividad VALUES ('0102', '01', 'Actividad', 'faces/mActividad.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0103', '01', 'Actividad Comercial', 'faces/mActividadCom.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0104', '01', 'Afiliado', 'faces/mAfiliado.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0105', '01', 'Agente', 'faces/mAgente.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0106', '01', 'Ajustador', 'faces/mAjustador.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0107', '01', 'Amparo', 'faces/mAmparo.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0108', '01', 'Banco', 'faces/mBanco.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0109', '01', 'Ciudad', 'faces/mCiudad.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01010', '01', 'Clase Auto', 'faces/mClaseAuto.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01011', '01', 'Clausula', 'faces/mClausula.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01012', '01', 'Cliente', 'faces/mCliente.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01013', '01', 'Comañia', 'faces/mCompania.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01014', '01', 'Concepto', 'faces/mConceptoAfe.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01015', '01', 'Amparo', 'faces/mAmparo.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01016', '01', 'Deduccion', 'faces/mDeduccion.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01017', '01', 'Departamento', 'faces/mDepartamento.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01018', '01', 'Empresa', 'faces/mEmpresa.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01019', '01', 'Familiar', 'faces/mFamiliar.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01020', '01', 'Forma Pago', 'faces/mFormaPago.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01021', '01', 'Marca', 'faces/mMarca.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01022', '01', 'Mensajero', 'faces/mMensajero.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01023', '01', 'Moneda', 'faces/mMoneda.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01024', '01', 'Pais', 'faces/mPais.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01025', '01', 'Rama', 'faces/mRama.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01026', '01', 'Grupo Rama', 'faces/mRamaGrupo.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01027', '01', 'Sucursal', 'faces/mSucursal.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01028', '01', 'Taller', 'faces/mTaller.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01029', '01', 'Tecnico', 'faces/mTecnico.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01030', '01', 'Tipo Cliente', 'faces/mTipoCli.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01031', '01', 'Tipo Negocio', 'faces/mTipoNegocio.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('01032', '01', 'Tipo Poliza', 'faces/mTipoPoliza.jsp','area' ,'a');

INSERT INTO tblActividad VALUES ('0201', '02', 'Agente', 'faces/mPolizaAgente.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0202', '02', 'Amparo', 'faces/mPolizaAmparo.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0203', '02', 'Articulo', 'faces/mPolizaArticulo.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0204', '02', 'Automovil', 'faces/mPolizaAutomovil.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0205', '02', 'Bien', 'faces/mPolizaBienes.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0206', '02', 'Cobro', 'faces/mPolizaCobro.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0207', '02', 'Vida', 'faces/mPolizaVida.jsp','area' ,'a');

INSERT INTO tblActividad VALUES ('0301', '03', 'Tipos Cliente', 'faces/mTipoCli.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0302', '03', 'Sector Cliente', 'faces/mSectorCli.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0303', '03', 'Nivel Cliente', 'faces/mNivelCli.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0304', '03', 'Regimen Cliente', 'faces/mRegimenCli.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0305', '03', 'Registro Relacion', 'faces/mRelacion.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0306', '03', 'Estudio Mercantil', 'faces/mEstudioMercantil.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0307', '03', 'Mensaje', 'faces/mMensaje.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0308', '03', 'Actividad', 'faces/mActividadCli.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0309', '03', 'Campaña', 'faces/mCampana.jsp','area' ,'a');

INSERT INTO tblActividad VALUES ('0401', '04', 'Reg movimiento', 'faces/iMovimiento.jsp','area' ,'a');
INSERT INTO tblActividad VALUES ('0402', '04', 'Con movimiento', 'faces/rMovimiento.jsp','area' ,'a');


CREATE TABLE tblsis
(
  idSis character varying NOT NULL,
  des character varying NOT NULL,
  Combo varchar(1),
  ColumnaCampo varchar,
```

```
  Tabla varchar,
  CONSTRAINT keySis PRIMARY KEY (idSis)
)
WITHOUT OIDS;
ALTER TABLE tblsis OWNER TO postgres;
INSERT INTO tblsis VALUES ('idtipocli', 'Codigo', 'n' , '-', '-');
INSERT INTO tblsis VALUES ('descripcion', 'Descripcion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idpolizaagente', 'Poliza Agente', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idcompania', 'Compañia', 's', '2', 'tblCompania');
INSERT INTO tblsis VALUES ('idrama', 'Rama', 's', '3', 'tblRama');
INSERT INTO tblsis VALUES ('numero', 'Numero Poliza', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idpolizaamparo', 'Poliza Amparo', 'n', '-', '-');
INSERT INTO tblsis VALUES ('numeropoliza', 'Numero Poliza', 'n', '-', '-');
INSERT INTO tblsis VALUES ('numeroorden', 'Numero Orden', 'n', '-', '-');
INSERT INTO tblsis VALUES ('articulo', 'Articulo', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idamparo', 'Amparo', 's', '2', 'tblAmparo');
INSERT INTO tblsis VALUES ('idmoneda', 'Moneda', 's', '2', 'tblMoneda');
INSERT INTO tblsis VALUES ('valorasesoria', 'Valor Asesoria', 'n', '-', '-');
INSERT INTO tblsis VALUES ('porcentaje', 'Porcentaje', 'n', '-', '-');
INSERT INTO tblsis VALUES ('tasa', 'Tasa', 'n', '-', '-');
INSERT INTO tblsis VALUES ('valorprima', 'Valor Prima', 'n', '-', '-');
INSERT INTO tblsis VALUES ('factor', 'Factor', 'n', '-', '-');
INSERT INTO tblsis VALUES ('iddeduccion', 'Deduccion', 's', '2', 'tblDeduccion');
INSERT INTO tblsis VALUES ('porcentajeded', 'Porcentaje Deduccion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('minimodes', 'Min Descuento', 'n', '-', '-');
INSERT INTO tblsis VALUES ('observaciones', 'Observaciones', 'n', '-', '-');
INSERT INTO tblsis VALUES ('vigenciainicio', 'Inicio Vigencia', 'n', '-', '-');
INSERT INTO tblsis VALUES ('vigenciafinal', 'Fin Vigencia', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idpolizaarticulo', 'Poliza Articulo', 'n', '-', '-');
INSERT INTO tblsis VALUES ('valorasegurado', 'Valor Asegurado', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idtipopoliza', 'Tipo Poliza', 's', '2', 'tblTipoPoliza');
INSERT INTO tblsis VALUES ('idpolizaautomobil', 'Poliza Auto', 'n', '-', '-');
INSERT INTO tblsis VALUES ('fechaingreso', 'Fecha Ingreso', 'n', '-', '-');
INSERT INTO tblsis VALUES ('fechamodificacion', 'Fecha Modificacion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('fechaejecucion', 'Fecha Ejecucion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('vigenciadesde', 'Inicio Vigencia', 'n', '-', '-');
INSERT INTO tblsis VALUES ('vigenciahasta', 'Fin Vigencia', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idcliente', 'Cliente', 's', '2', 'tblCliente');
INSERT INTO tblsis VALUES ('idmarca', 'Marca', 's', '2', 'tblMarca');
INSERT INTO tblsis VALUES ('idclaseauto', 'Clase', 's', '2', 'tblClaseAuto');
INSERT INTO tblsis VALUES ('modelo', 'Modelo', 'n', '-', '-');
INSERT INTO tblsis VALUES ('placa', 'Placa', 'n', '-', '-');
INSERT INTO tblsis VALUES ('chasis', 'Chasis', 'n', '-', '-');
INSERT INTO tblsis VALUES ('serie', 'Serie', 'n', '-', '-');
INSERT INTO tblsis VALUES ('valoradicion', 'Valor Adicion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('textoadicion', 'Texto Adicion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('alarma', 'Alarma', 'n', '-', '-');
INSERT INTO tblsis VALUES ('valorasegurar', 'Valor Asegurar', 'n', '-', '-');
INSERT INTO tblsis VALUES ('color', 'Color', 'n', '-', '-');
INSERT INTO tblsis VALUES ('linea', 'Linea', 'n', '-', '-');
INSERT INTO tblsis VALUES ('nombrebeneficiario', 'Beneficiario', 'n', '-', '-');
INSERT INTO tblsis VALUES ('numeropasajeros', 'Pasajeros', 'n', '-', '-');
INSERT INTO tblsis VALUES ('ficha', 'Ficha', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idpolizabienes', 'Poliza Bien', 'n', '-', '-');
INSERT INTO tblsis VALUES ('coddep', 'Dependencia', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idciudadbien', 'Ciudad Bien', 's', '2', 'tblCiudad');
INSERT INTO tblsis VALUES ('direccion', 'Direccion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('nombreasegurado', 'Nombre', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idpolizacobro', 'Poliza Cobro', 'n', '-', '-');
INSERT INTO tblsis VALUES ('cuota', 'Cuota', 'n', '-', '-');
```

```
INSERT INTO tblsis VALUES ('fechacobro', 'Fecha Cobro', 'n', '-', '-');
INSERT INTO tblsis VALUES ('valorgastos', 'Gastos', 'n', '-', '-');
INSERT INTO tblsis VALUES ('valorotros', 'Otros', 'n', '-', '-');
INSERT INTO tblsis VALUES ('idpolizavida', 'Poliza Vida', 'n', '-', '-');
INSERT INTO tblsis VALUES ('fechaexpedicion', 'Fecha Expedicion', 'n', '-', '-');
INSERT INTO tblsis VALUES ('fechanacimiento', 'Fecha Nacimiento', 'n', '-', '-');
INSERT INTO tblsis VALUES ('edad', 'Edad', 'n', '-', '-');
INSERT INTO tblsis VALUES ('tasa', 'Tasa', 'n', '-', '-');

CREATE TABLE tblSubActividad
(
  idSubActividad character varying NOT NULL,
  idActividad character varying NOT NULL,
  Descripcion character varying,
  url character varying,
  target character varying,
  estado varchar(1),
  CONSTRAINT keySubActividad PRIMARY KEY (idSubActividad)
)
WITHOUT OIDS;
ALTER TABLE tblSubActividad OWNER TO postgres;

CREATE TABLE tblConfiguracion
(
  idUsuario character varying NOT NULL,
  idModulo character varying NOT NULL,
  estado varchar(1),
  CONSTRAINT keyConfiguracion PRIMARY KEY (idUsuario, idModulo)
)
WITHOUT OIDS;
ALTER TABLE tblConfiguracion OWNER TO postgres;

INSERT INTO tblConfiguracion VALUES ('1','01' ,'a');
INSERT INTO tblConfiguracion VALUES ('1','02' ,'a');
INSERT INTO tblConfiguracion VALUES ('1','03' ,'a');
INSERT INTO tblConfiguracion VALUES ('1','04' ,'a');

CREATE TABLE tblModulo
(
  idModulo character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyModulo PRIMARY KEY (idModulo)
)
WITHOUT OIDS;
ALTER TABLE tblModulo OWNER TO postgres;

INSERT INTO tblModulo VALUES ('01','Administracion' ,'a');
INSERT INTO tblModulo VALUES ('02','Poliza' ,'a');
INSERT INTO tblModulo VALUES ('03','CRM' ,'a');
INSERT INTO tblModulo VALUES ('04','Cartera' ,'a');

CREATE TABLE tblUsuario
(
  idUsuario character varying NOT NULL,
  Descripcion character varying,
  Pass character varying,
  Observaciones character varying,
  estado varchar(1),
  CONSTRAINT keyUsuario PRIMARY KEY (idUsuario)
```

```
)
WITHOUT OIDS;
ALTER TABLE tblUsuario OWNER TO postgres;
INSERT INTO tblUsuario VALUES ('1','Administrador','1','Gestion', 'a');

CREATE TABLE tblgUsuario
(
  idUsuario character varying NOT NULL,
  idgUsuario character varying NOT NULL,
  Estado varchar(1),
  CONSTRAINT keygUsuario PRIMARY KEY (idgUsuario, idUsuario)
)
WITHOUT OIDS;
ALTER TABLE tblgUsuario OWNER TO postgres;

CREATE TABLE tblCliente
(
  idCliente character varying NOT NULL,
  idSucursal character varying NOT NULL DEFAULT '-',
  Nombre character varying NOT NULL,
  ApellidoA character varying NOT NULL,
  ApellidoB character varying NOT NULL,
  idTipoCli character varying NOT NULL,
  DireccionRes character varying,
  DireccionOfi character varying,
  idCiudad character varying NOT NULL,
  TelefonoF character varying,
  TelefonoM character varying,
  Fax character varying,
  Correoe character varying,
  Genero character varying,
  Nacimiento varchar,
  Registro varchar,
  url character varying,
  estado varchar(1),
  CONSTRAINT keyCliente PRIMARY KEY (idCliente, idSucursal)
)
WITH (OIDS=TRUE);
ALTER TABLE tblCliente OWNER TO postgres;
INSERT INTO tblcliente(
            idcliente, idsucursal, nombre, apellidoa, apellidob, idtipocli,
            direccionres, direccionofi, idciudad, telefonof, telefonom, fax,
            correoe, genero, nacimiento, registro, url, estado)
    VALUES ('01', '01', 'Nombre Cliente', 'Apellido a', 'Apellido b', '01',
            'Direccion res', 'Direccion ofi', '01', 'Telefono fij', 'Telefono Mov', 'Fax',
            'Correo', 'Genero', '01/01/1970', '01/01/1970', 'web', 'a');

CREATE TABLE tblAbogado
(
    idAbogado character varying NOT NULL,
    Nombre character varying NOT NULL,
    ApellidoA character varying NOT NULL,
    ApellidoB character varying NOT NULL,
    Tarjeta character varying NOT NULL,
    DireccionRes character varying,
    DireccionOfi character varying,
    idCiudad character varying NOT NULL,
    TelefonoF character varying,
    TelefonoM character varying,
    Fax character varying,
```

```
   estado varchar(1),
  CONSTRAINT keyAbogado PRIMARY KEY (idAbogado)
)
WITH (OIDS=TRUE);
ALTER TABLE tblAbogado OWNER TO postgres;
INSERT INTO tblabogado(
            idabogado, nombre, apellidoa, apellidob, tarjeta, direccionres,
            direccionofi, idciudad, telefonof, telefonom, fax, estado)
    VALUES ('01', 'Nombre abogado', 'Apellido a', 'Apellido b', '0110', 'Direccion res',
            'Direecion fi', '01', 'Telefono fij', 'Telefono Mon', 'Fax', 'a');

CREATE TABLE tblActividadCom
(
  idActividadCom character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyActividadCom PRIMARY KEY (idActividadCom)
)
WITHOUT OIDS;
ALTER TABLE idActividadCom OWNER TO postgres;
INSERT INTO tblActividadCom VALUES ('01','ACtividad Economica', 'a');

CREATE TABLE tblConceptoAfe
(
  idConceptoAfe character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyConceptoAfe PRIMARY KEY (idConceptoAfe)
)
WITHOUT OIDS;
ALTER TABLE tblConceptoAfe OWNER TO postgres;
INSERT INTO tblConceptoAfe VALUES ('01','Concepto', 'a');

CREATE TABLE tblAfiliado
(
  idAfiliado character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyAfiliado PRIMARY KEY (idAfiliado)
)
WITHOUT OIDS;
ALTER TABLE tblAfiliado OWNER TO postgres;
INSERT INTO tblAfiliado VALUES ('01','Concepto', 'a');

CREATE TABLE tblAGente
(
    idAGente character varying NOT NULL,
    idTipoAgente character varying NOT NULL,
    Nombre character varying NOT NULL,
    ApellidoA character varying NOT NULL,
    ApellidoB character varying NOT NULL,
    estado varchar(1),
  CONSTRAINT keyAGente PRIMARY KEY (idAGente, idTipoAgente)
)
WITH (OIDS=TRUE);
ALTER TABLE tblAGente OWNER TO postgres;
INSERT INTO tblagente(
            idagente, idTipoAgente, nombre, apellidoa, apellidob, estado)
    VALUES ('01', '01', 'Nombre Agente', 'Apellido a', 'Apellido b', 'a');
```

```
CREATE TABLE tblAjustador
(
   idAjustador character varying NOT NULL,
   Nombre character varying NOT NULL,
   ApellidoA character varying NOT NULL,
   ApellidoB character varying NOT NULL,
   Tarjeta character varying NOT NULL,
   DireccionRes character varying,
   DireccionOfi character varying,
   idCiudad character varying NOT NULL,
   TelefonoF character varying,
   TelefonoM character varying,
   Fax character varying,
   estado varchar(1),
  CONSTRAINT keyAjustador PRIMARY KEY (idAjustador)
)
WITH (OIDS=TRUE);
ALTER TABLE tblAjustador OWNER TO postgres;
INSERT INTO tblajustador(
            idajustador, nombre, apellidoa, apellidob, tarjeta, direccionres,
            direccionofi, idciudad, telefonof, telefonom, fax, estado)
    VALUES ('01', 'Nombre', 'Apellido a', 'Apellido b', '1010', 'Direccion res',
            'Direccion fi', '01', 'Telefono f', 'Telefono m', 'fax', 'a');


CREATE TABLE tblAmparo
(
  idAmparo character varying NOT NULL,
  Nombre character varying NOT NULL,
  Abrebiatura character varying NOT NULL,
  Descripcion character varying NOT NULL,
  --sumamp boolean DEFAULT false,
  --aseamp boolean DEFAULT false,
  --tasamp boolean DEFAULT false,
  --poramp boolean DEFAULT false,
  --priamp boolean DEFAULT false,
  --codram character varying(3) DEFAULT ''::character varying,
  estado varchar(1),
  CONSTRAINT keyAmparo PRIMARY KEY (idAmparo)
)
WITH (OIDS=TRUE);
ALTER TABLE tblAmparo OWNER TO postgres;
INSERT INTO tblamparo(
            idamparo, nombre, abrebiatura, descripcion, estado)
    VALUES ('01', 'Amparo', 'AMP', 'El Amparo', 'a');

CREATE TABLE tblBanco
(
  idBanco character varying NOT NULL,
  nit character varying,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyBanco PRIMARY KEY (idBanco)
)
WITHOUT OIDS;
ALTER TABLE tblBanco OWNER TO postgres;
INSERT INTO tblBanco VALUES ('01','Banco', 'nit', 'a');

CREATE TABLE tblCiudad
(
```

```
  idCiudad character varying NOT NULL,
  idDepartamento character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyCiudad PRIMARY KEY (idCiudad)
)
WITHOUT OIDS;
ALTER TABLE tblCiudad OWNER TO postgres;
INSERT INTO tblCiudad VALUES ('01','01', 'Pereira', 'a');

CREATE TABLE tblDepartamento
(
  idDepartamento character varying NOT NULL,
  idPais character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyDepartamento PRIMARY KEY (idDepartamento)
)
WITHOUT OIDS;
ALTER TABLE tblDepartamento OWNER TO postgres;
INSERT INTO tblDepartamento VALUES ('01','01', 'Risaralda', 'a');

CREATE TABLE tblPais
(
  idPais character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyPais PRIMARY KEY (idPais)
)
WITHOUT OIDS;
ALTER TABLE tblPais OWNER TO postgres;
INSERT INTO tblPais VALUES ('01', 'Risaralda', 'a');

CREATE TABLE tblClaseAuto
(
  idClaseAuto character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyClaseAuto PRIMARY KEY (idClaseAuto)
)
WITHOUT OIDS;
ALTER TABLE tblClaseAuto OWNER TO postgres;
INSERT INTO tblClaseAuto VALUES ('01', 'Clase Auto', 'a');

CREATE TABLE tblClausula
(
  idClausula character varying NOT NULL,
  Nombre character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyClausula PRIMARY KEY (idClausula)
)
WITHOUT OIDS;
ALTER TABLE tblClausula OWNER TO postgres;
INSERT INTO tblClausula VALUES ('01', 'Nombre', 'Clausula', 'a');

CREATE TABLE tblDeduccion
(
  idDeduccion character varying NOT NULL,
  Descripcion character varying,
```

```
  estado varchar(1),
  CONSTRAINT keyDeduccion PRIMARY KEY (idDeduccion)
)
WITHOUT OIDS;
ALTER TABLE tblDeduccion OWNER TO postgres;
INSERT INTO tblDeduccion VALUES ('01', 'Deduccion', 'a');

CREATE TABLE tblTipoDocumento
(
  idTipoDocumento character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyTipoDocumento PRIMARY KEY (idTipoDocumento)
)
WITHOUT OIDS;
ALTER TABLE tblTipoDocumento OWNER TO postgres;
INSERT INTO tblTipoDocumento VALUES ('01', 'Tipo Documento', 'a');

CREATE TABLE tblFamiliar
(
  idFamiliar character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyFamiliar PRIMARY KEY (idFamiliar)
)
WITHOUT OIDS;
ALTER TABLE tblFamiliar OWNER TO postgres;
INSERT INTO tblFamiliar VALUES ('01', 'Familiar', 'a');

CREATE TABLE tblFormaPago
(
  idFormaPago character varying NOT NULL,
  Descripcion character varying,
  Numerocuenta character varying,
  --codint smallint DEFAULT 0,
  --indcta boolean DEFAULT false,
  --indcia boolean DEFAULT false,
  --indctacia boolean DEFAULT false,
  --indbco boolean DEFAULT false,
  estado varchar(1),
  CONSTRAINT keyFormaPago PRIMARY KEY (idFormaPago)
)
WITH (OIDS=TRUE);
ALTER TABLE tblFormaPago OWNER TO postgres;
INSERT INTO tblFormaPago VALUES ('01', 'Forma Pago', '1002', 'a');

CREATE TABLE tblMarca
(
  idMarca character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyMarca PRIMARY KEY (idMarca)
)
WITHOUT OIDS;
ALTER TABLE tblMarca OWNER TO postgres;
INSERT INTO tblMarca VALUES ('01', 'Mazda', 'a');

CREATE TABLE tblMensajero
(
    idMensajero character varying NOT NULL,
```

```
    Nombre character varying NOT NULL,
    ApellidoA character varying NOT NULL,
    ApellidoB character varying NOT NULL,
    DireccionRes character varying,
    TelefonoF character varying,
    TelefonoM character varying,
    estado varchar(1),
  CONSTRAINT keyMensajero PRIMARY KEY (idMensajero)
)
WITH (OIDS=TRUE);
ALTER TABLE tblMensajero OWNER TO postgres;
INSERT INTO tblMensajero VALUES ('01', 'Nombre', 'Apellido a', 'Apellido b', 'Direcciones res', 'Telefono f',
'Telefono m', 'a');

CREATE TABLE tblMoneda
(
  idMoneda character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyMoneda PRIMARY KEY (idMoneda)
)
WITHOUT OIDS;
ALTER TABLE tblMoneda OWNER TO postgres;
INSERT INTO tblMoneda VALUES ('01', 'Peso', 'a');

CREATE TABLE tblTipoNegocio
(
  idTipoNegocio character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyTipoNegocio PRIMARY KEY (idTipoNegocio)
)
WITHOUT OIDS;
ALTER TABLE tblTipoNegocio OWNER TO postgres;
INSERT INTO tblTipoNegocio VALUES ('01', 'Tipo Negocio', 'a');

CREATE TABLE tblRama
(
  idRama character varying NOT NULL,
  idRamaGrupo character varying NOT NULL,
  Descripcion character varying,
  Comision int4,
  estado varchar(1),
  CONSTRAINT keyRama PRIMARY KEY (idRama)
)
WITHOUT OIDS;
ALTER TABLE tblRama OWNER TO postgres;
INSERT INTO tblRama VALUES ('01', '01', 'La Rama', 1000, 'a');

CREATE TABLE tblRamaGrupo
(
  idRamaGrupo character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyRamaGrupo PRIMARY KEY (idRamaGrupo)
)
WITHOUT OIDS;
ALTER TABLE tblRamaGrupo OWNER TO postgres;
INSERT INTO tblRamaGrupo VALUES ('01', 'La Rama', 'a');
```

```
CREATE TABLE tblSucursal
(
  idSucursal character varying NOT NULL,
  idCiudad character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keySucursal PRIMARY KEY (idSucursal, idCiudad)
)
WITHOUT OIDS;
ALTER TABLE tblSucursal OWNER TO postgres;
INSERT INTO tblSucursal VALUES ('01', '01', 'La sucursal', 'a');

CREATE TABLE tblTaller
(
  idTaller character varying NOT NULL,
  idCiudad character varying NOT NULL,
  nombre character varying NOT NULL,
  direccion character varying NOT NULL,
  telefonoF character varying NOT NULL,
  telefonC character varying NOT NULL,
  fax character varying NOT NULL,
  correoe character varying NOT NULL,
  estado varchar(1),
  CONSTRAINT keyTaller PRIMARY KEY (idTaller)
)
WITH (OIDS=TRUE);
ALTER TABLE tblTaller OWNER TO postgres;
INSERT INTO tblTaller VALUES ('01', '01', 'Nombre', 'Direcciones', 'Telefono f', 'Telefono m', 'fax', 'correoe',
'a');

CREATE TABLE tblTecnico
(
  idTecnico character varying NOT NULL,
  idSucursal character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyTecnico PRIMARY KEY (idTecnico, idSucursal)
)
WITHOUT OIDS;
ALTER TABLE tblTecnico OWNER TO postgres;
INSERT INTO tblTecnico VALUES ('01', '01', 'Tecnico', 'a');

CREATE TABLE tblTipoPoliza
(
  idTipoPoliza character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyTipoPoliza PRIMARY KEY (idTipoPoliza)
)
WITHOUT OIDS;
ALTER TABLE tblTipoPoliza OWNER TO postgres;
INSERT INTO tblTipoPoliza VALUES ('01', 'Tipo Poliza', 'a');

CREATE TABLE tblEmpresa
(
  idEmpresa character varying NOT NULL,
  Descripcion character varying,
  estado varchar(1),
  CONSTRAINT keyEmpresa PRIMARY KEY (idEmpresa)
)
```

```
WITHOUT OIDS;
ALTER TABLE tblEmpresa OWNER TO postgres;
INSERT INTO tblEmpresa VALUES ('01', 'Empresa', 'a');

CREATE TABLE tblCompania
(
  idCompania character varying NOT NULL,
  Razon character varying,
  Direccion varchar,
  Telefono varchar,
  idCiudad varchar,
  correo varchar,
  web varchar,
  estado varchar(1),
  CONSTRAINT keyCompania PRIMARY KEY (idCompania)
)
WITHOUT OIDS;
ALTER TABLE tblCompania OWNER TO postgres;
INSERT INTO tblCompania VALUES ('01', 'Razon Compañia', 'Direcciones', 'Telefono f', '01', 'correoe', 'web',
'a');

CREATE TABLE tblPolizaAgente
(
  idPolizaAgente character varying NOT NULL,
  idCompania character varying NOT NULL,
  idRama character varying NOT NULL,
  Numero character varying NOT NULL,
  estado varchar(1),
  CONSTRAINT keyPolizaAgente PRIMARY KEY (idPolizaAgente, idCompania, idRama, Numero)
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaAgente OWNER TO postgres;

CREATE TABLE tblPolizaAmparo
(
  idPolizaAmparo character varying NOT NULL,
  idCompania character varying NOT NULL,
  idRama character varying NOT NULL,
  NumeroPoliza character varying NOT NULL,
  NumeroOrden character varying NOT NULL,
  articulo varchar DEFAULT 0,
  idAmparo character varying NOT NULL,
  idMoneda character varying NOT NULL,
  ValorAsesoria double precision DEFAULT 0,
  Porcentaje numeric(5,2) DEFAULT 0,
  Tasa numeric(10,4) DEFAULT 0,
  ValorPrima double precision DEFAULT 0,
  Factor integer DEFAULT 0,
  idDeduccion character varying NOT NULL,
  PorcentajeDed numeric(5,2) DEFAULT 0,
  MinimoDes double precision DEFAULT 0,
  Observaciones character varying,
  VigenciaInicio varchar,
  VigenciaFinal varchar,
  estado varchar(1),
  CONSTRAINT keytblPolizaAmparo PRIMARY KEY (idPolizaAmparo, idCompania, idRama, NumeroPoliza,
NumeroOrden, articulo)
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaAmparo OWNER TO postgres;
```

```
CREATE TABLE tblPolizaArticulo
(
  idPolizaArticulo character varying NOT NULL,
  idCompania character varying NOT NULL,
  idRama character varying NOT NULL,
  NumeroPoliza character varying NOT NULL,
  NumeroOrden character varying NOT NULL,
  articulo int4 NOT NULL DEFAULT 0,
  Descripcion character varying NOT NULL,
  --coaseg double precision DEFAULT 0,
  idMoneda character varying NOT NULL,
  ValorAsegurado double precision DEFAULT 0,
  idTipoPoliza character varying NOT NULL,
  estado varchar(1),
  CONSTRAINT keyPolizaArticulo PRIMARY KEY (idPolizaArticulo, idCompania, idRama, NumeroPoliza,
NumeroOrden, articulo)
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaArticulo OWNER TO postgres;

CREATE TABLE tblPolizaAutomobil
(
  idPolizaAutomobil character varying NOT NULL,
  idCompania character varying NOT NULL,
  idRama character varying NOT NULL,
  NumeroPoliza character varying NOT NULL,
  NumeroOrden character varying NOT NULL,
  FechaIngreso varchar,
  FechaModificacion varchar,
  FechaEjecucion varchar,
  VigenciaDesde varchar,
  VigenciaHasta varchar,
  idCliente character varying NOT NULL,
  idMarca character varying NOT NULL,
  idClaseAuto character varying NOT NULL,
  --faseco character varying,
  Modelo character varying,
  Placa character varying,
  Chasis character varying,
  Serie character varying,
  --coduso smallint DEFAULT 0,
  ValorAdicion int DEFAULT 0,
  TextoAdicion character varying,
  Alarma varchar DEFAULT false,
  --dctrec numeric(5,2) DEFAULT 0,
  ValorAsegurar int DEFAULT 0,
  ValorPrima int DEFAULT 0,
  idTipoPoliza character varying NOT NULL,
  Color character varying,
  Linea character varying,
  NombreBeneficiario character varying,
  NumeroPasajeros int DEFAULT 0,
  --codafi smallint DEFAULT 1,
  Ficha character varying,
  --codtar smallint DEFAULT 0,
  --codexc smallint DEFAULT 0,
  estado varchar(1),
  CONSTRAINT tpolaut_pkey PRIMARY KEY (idPolizaAutomobil, idCompania, idRama, NumeroPoliza,
NumeroOrden)
```

```
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaAutomobil OWNER TO postgres;

CREATE TABLE tblPolizaBienes
(
  idPolizaBienes character varying NOT NULL,
  idCompania character varying NOT NULL,
  idRama character varying NOT NULL,
  NumeroPoliza character varying NOT NULL,
  NumeroOrden character varying NOT NULL,
  idCliente character varying NOT NULL,
  Descripcion character varying NOT NULL,
  coddep int DEFAULT 0,
  idCiudadBien character varying NOT NULL,
  Direccion character varying NOT NULL,
  ValorPrima double precision DEFAULT 0,
  --codafi smallint DEFAULT 0,
  --codtar smallint DEFAULT 0,
  --numtar character varying(20) DEFAULT ''::character varying,
  NombreAsegurado character varying NOT NULL,
  Ficha character varying,
  FechaIngreso varchar,
  FechaEjecucion varchar,
  FechaModificacion varchar,
  estado varchar(1),
  CONSTRAINT tpolbienes_pkey PRIMARY KEY (idPolizaBienes, idCompania, idRama, NumeroPoliza,
NumeroOrden)
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaBienes OWNER TO postgres;

CREATE TABLE tblPolizaCobro
(
  idPolizaCobro character varying NOT NULL,
  idCompania character varying NOT NULL,
  idRama character varying NOT NULL,
  NumeroPoliza character varying NOT NULL,
  NumeroOrden character varying NOT NULL,
  articulo int4 NOT NULL DEFAULT 0,
  Descripcion character varying NOT NULL,
  cuota int4 NOT NULL DEFAULT 0,
  FechaCobro varchar,
  ValorPrima int4 DEFAULT 0,
  ValorGastos int4 DEFAULT 0,
  ValorOtros int4 DEFAULT 0,
  ValorTotal int4 DEFAULT 0,
  FechaIngreso varchar,
  estado varchar(1),
  CONSTRAINT keyPolizaCobro PRIMARY KEY  (idPolizaCobro, idCompania, idRama, NumeroPoliza,
NumeroOrden, articulo)
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaCobro OWNER TO postgres;

CREATE TABLE tblPolizaVida
(
   idPolizaVida character varying NOT NULL,
   idCompania character varying NOT NULL,
   idRama character varying NOT NULL,
```

```
  NumeroPoliza character varying NOT NULL,
  NumeroOrden character varying NOT NULL,
  idCliente character varying NOT NULL,
  FechaIngreso varchar,
  FechaModificacion varchar,
  FechaExpedicion varchar,
  VigenciaInicio varchar,
  VigenciaHasta varchar,
  FechaNacimiento varchar,
  Edad int4 DEFAULT 0,
  idTipoPoliza character varying NOT NULL,
  Tasa int4 DEFAULT 0,
  ValorAsegurado int4 DEFAULT 0,
  ValorPrima int4 DEFAULT 0,
  --codafi smallint DEFAULT 0,
  Ficha character varying(10) DEFAULT ''::character varying,
  --codtar smallint DEFAULT 0,
 --numtar character varying(20) DEFAULT ''::character varying,
 --subgrupo boolean DEFAULT false,
 --codexc smallint DEFAULT 0,
 estado varchar(1),
 CONSTRAINT keyPolizaVida PRIMARY KEY (idPolizaVida, idCompania, idRama, NumeroPoliza,
NumeroOrden)
)
WITH (OIDS=TRUE);
ALTER TABLE tblPolizaVida OWNER TO postgres;

CREATE TABLE tblMovimiento
(
        idMovimiento varchar,
        idTipoMovimiento varchar,
   idCliente varchar,
        Descripcion varchar,
        signo varchar,
        Fecha varchar,
        Hora varchar,
        Valor int,
   Observaciones varchar,
   Estado varchar(1),
        CONSTRAINT IDtblMovimiento PRIMARY KEY (idMovimiento, idTipoMovimiento, idCliente,
signo) USING INDEX TABLESPACE pg_default
)
WITHOUT OIDS;
ALTER TABLE tblMovimiento OWNER TO postgres;
```

ANEXO F
MANUAL TÉCNICO

CONTENIDO

Instalación de la aplicación

Para la instalación de la aplicación es necesario configurar un servidor WEB detal forma que permita la interacción del los clientes con el restaurante en línea desde cualquier punto de acceso.

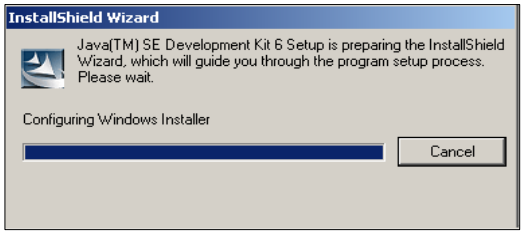
El servidor es un elemento esencial distribución de recursos para el proyecto y para tales efectos se usa el software TOMCAT, el cual permite la interacción de la base de datos con los clientes, los cuales cuentan con un BROWSER (navegador) que procese las transacciones.

Los siguientes pasos son requeridos para la instalación y configuración de la aplicación:

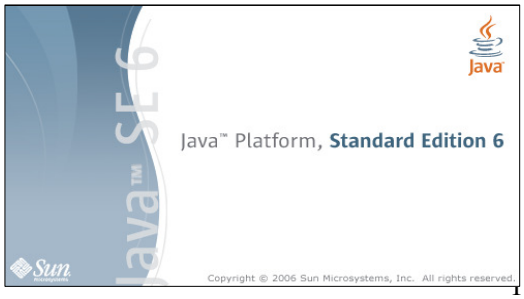
1. Instalar JAVA

a. Buscamos la opción para instalar java llamada jdk-6-beta-windows-i586, ejecutamos esta opción:

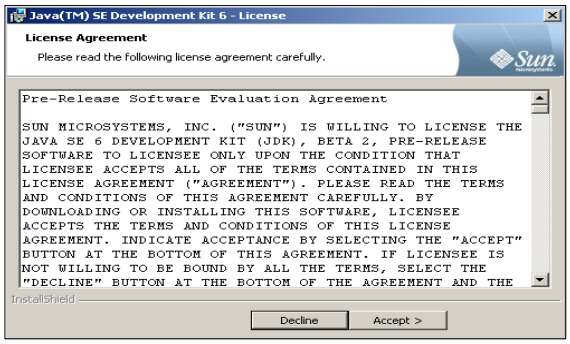
b. Al momento de darle doble clic empieza a ejecutar el archivo.



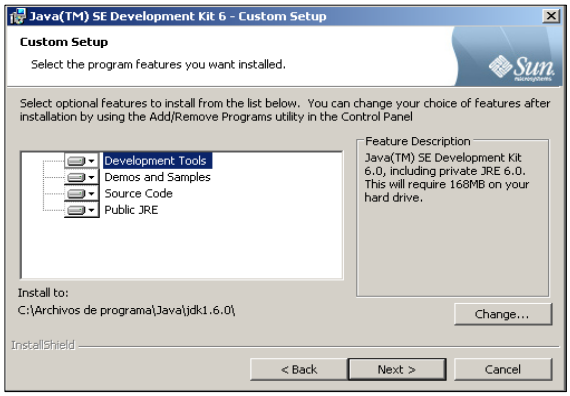
c.



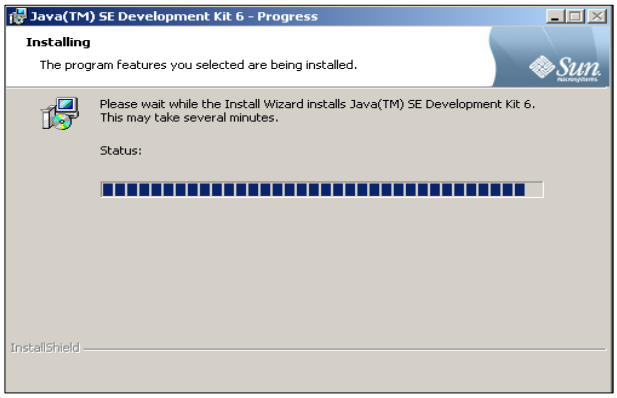
d. En este cuadro aceptamos los términos del contrato



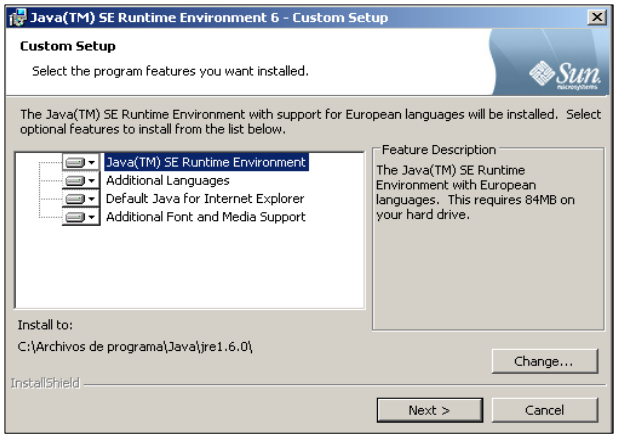
. El siguiente paso es darle siguiente (Next)



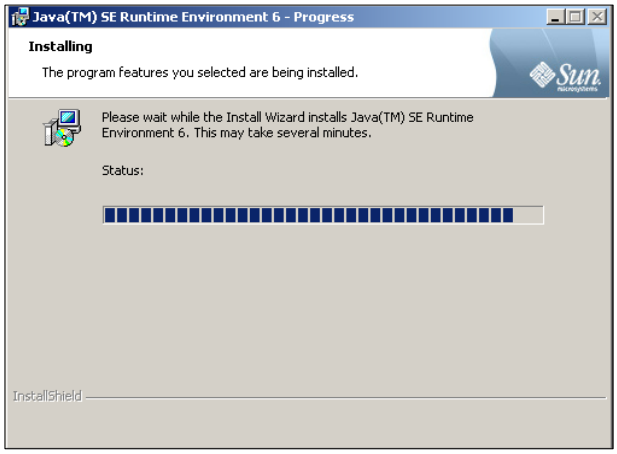
f.



g. Seguimos el mismo proceso dándole siguiente (Next)

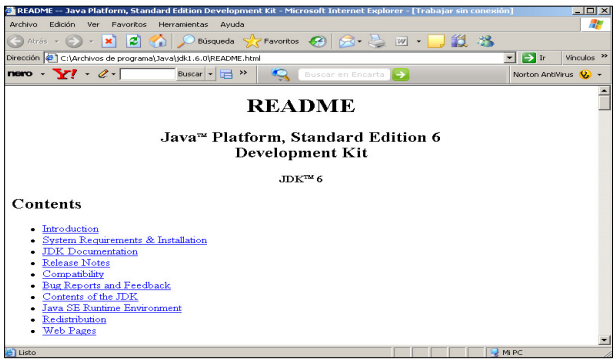


h.



i. La instalación ha finalizado

j. En este cuadro nos muestra la nota de lectura cuando ya se ha finalizado la instalación de JAVA

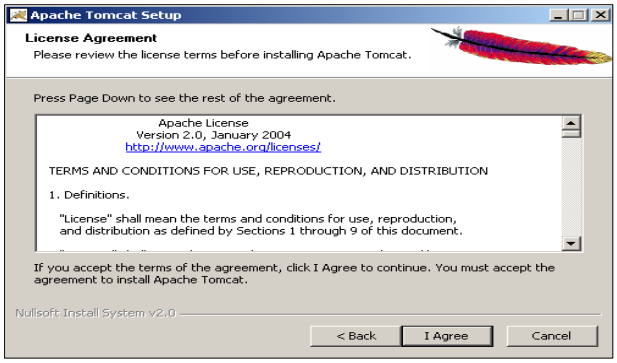


2. Instalar TOMCAT

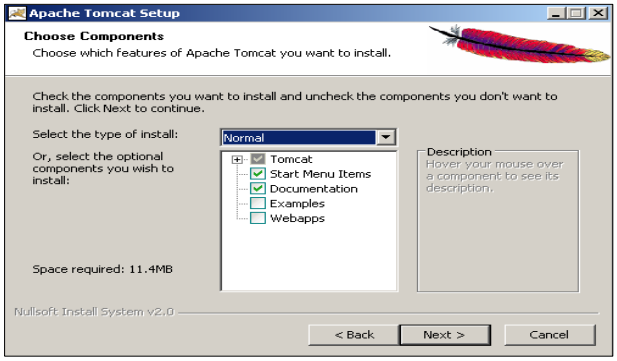
a. Esta es la opción para empezar a ejecutar el archivo de Tomcat 5.5

b.

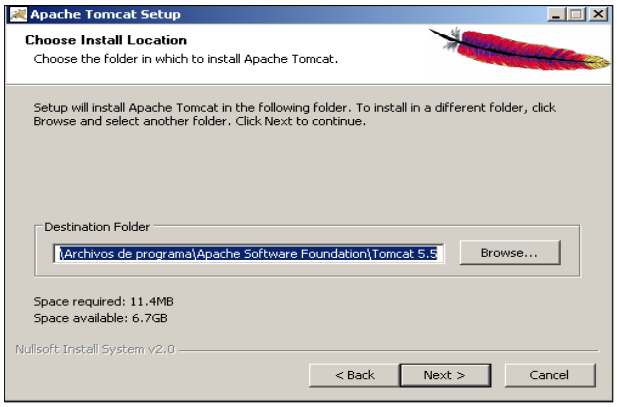
c.


Apache Tomcat Setup
License Agreement
Please review the license terms before installing Apache Tomcat.
Press Page Down to see the rest of the agreement.
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/
TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
1. Definitions.
"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.
If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Apache Tomcat.
Nullsoft Install System v2.0
< Back
I Agree
Cancel

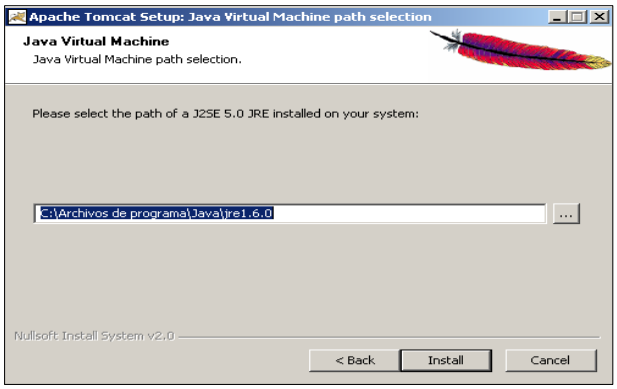
d.


Apache Tomcat Setup
Choose Components
Choose which features of Apache Tomcat you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select the type of install:
Normal
Or, select the optional components you wish to install:
Tomcat
Start Menu Items
Documentation
Examples
Webapps
Description
Hover your mouse over a component to see its description.
Space required: 11.4MB
Nullsoft Install System v2.0
< Back
Next >
Cancel

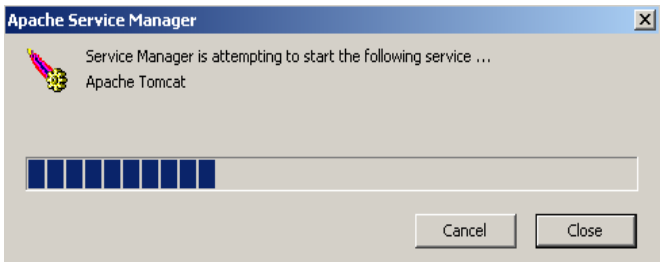
e.


Apache Tomcat Setup
Choose Install Location
Choose the folder in which to install Apache Tomcat.
Setup will install Apache Tomcat in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
\Archivos de programa\Apache Software Foundation\Tomcat 5.5
Browse...
Space required: 11.4MB
Space available: 6.7GB
Nullsoft Install System v2.0
< Back
Next >
Cancel

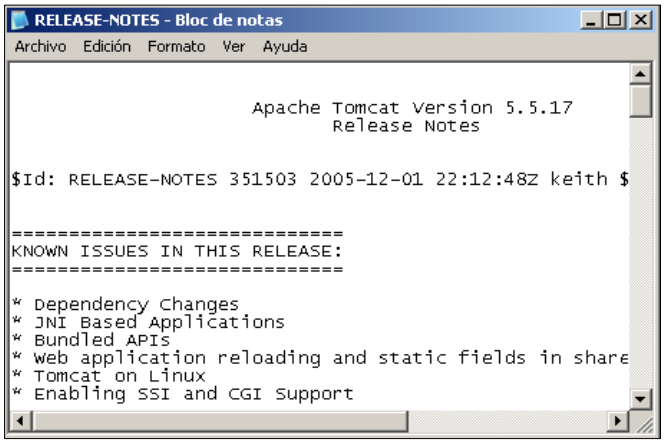
f.


Apache Tomcat Setup: Java Virtual Machine path selection
Java Virtual Machine
Java Virtual Machine path selection.
Please select the path of a J2SE 5.0 JRE installed on your system:
C:\Archivos de programa\Java\jre1.6.0
...
Nullsoft Install System v2.0
< Back
Install
Cancel

j.


Apache Service Manager
Service Manager is attempting to start the following service ...
Apache Tomcat
Cancel
Close

k.


RELEASE-NOTES - Bloc de notas
Archivo Edición Formato Ver Ayuda
Apache Tomcat Version 5.5.17
Release Notes
$Id: RELEASE-NOTES 351503 2005-12-01 22:12:48Z keith $
==========
KNOWN ISSUES IN THIS RELEASE:
==========
* Dependency Changes
* JNI Based Applications
* Bundled APIs
* Web application reloading and static fields in share
* Tomcat on Linux
* Enabling SSI and CGI Support

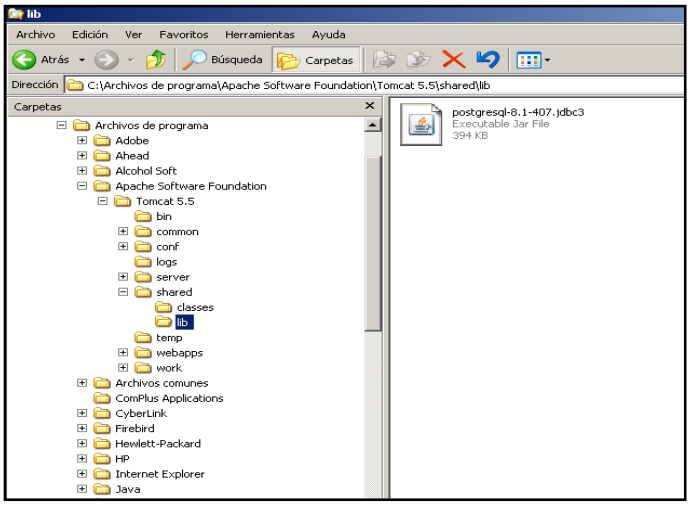
3. Copiado del software

Tomar del CD el archivo Secure que se encuentra adjunto y copiarlo en la carpeta C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\WEBapps\ROOT, y luego de descomprimirla allí mismo se deberá crear una capeta llamada Secure.

4. Instalación de DRIVERS

Para la puesta en marcha de la aplicación es necesario el uso de DRIVERS que permitan la conexión con las bases de datos. Para tales efectos es necesario copiar el archivo postgresql-8.1-407. jdbc3.jar en la carpeta C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\shared\lib, tal y como lo indica el siguiente gráfico:

a.

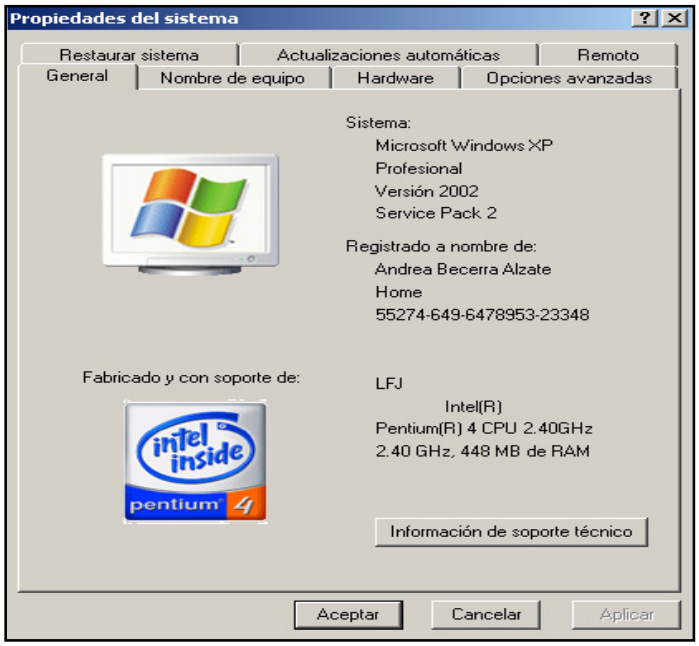


Ejecución de SCRITP SQL

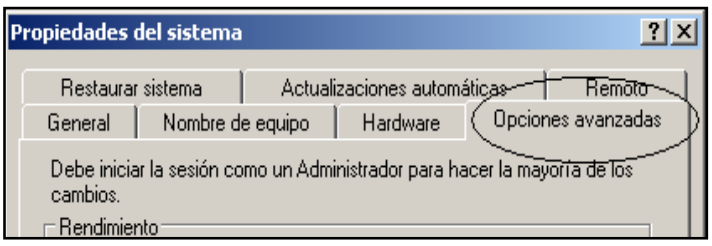
Para la instalación de las bases de datos es necesario ejecutar el SCRITP SQL que permita la creación de las bases de datos del proyecto.

Primero que todo es necesario ingresar una variable de entorno para que se pueda ejecutar el comando *psql*, y se logra mediante los siguientes pasos

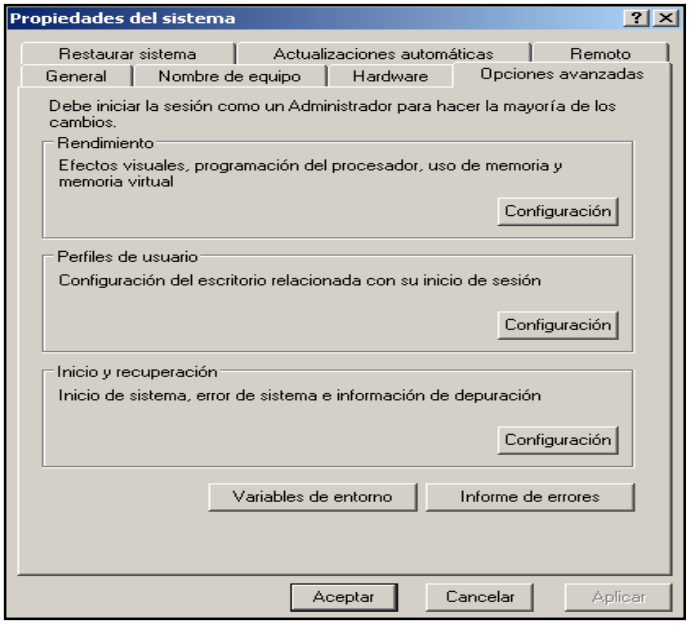
a. Abrir la ventana de propiedades de MI PC



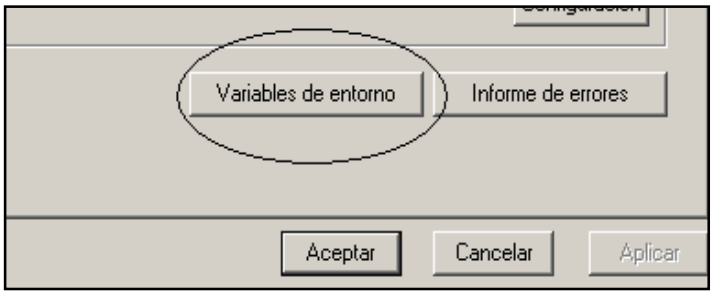
Luego de lograr abrir la ventana anterior es necesario hacer clic en la pestaña de opciones avanzadas, tal y como aparece en el siguiente gráfico:



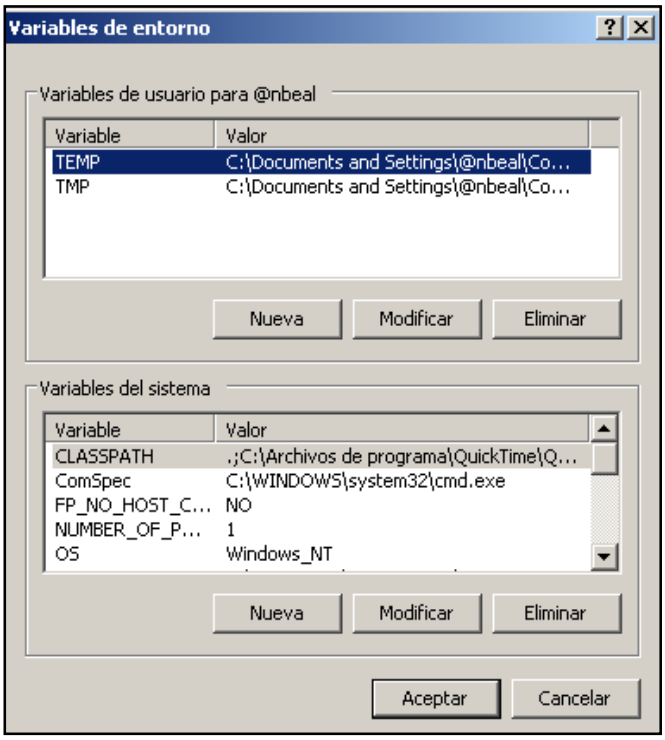
Lo anterior mostrará la siguiente ventana



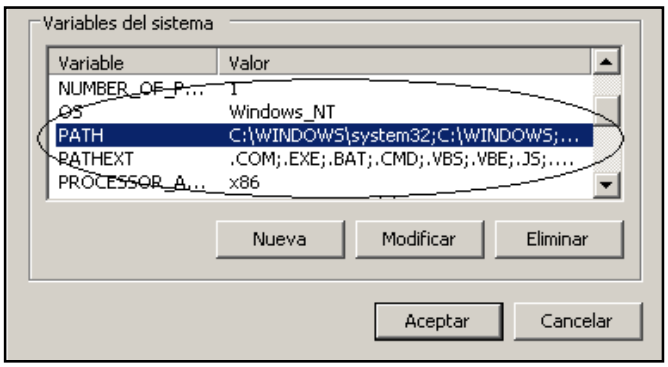
b. Para luego hacer clic en el botón variables de entorno, basarse en el siguiente gráfico:



Lo anterior nos mostrará la siguiente interfaz:

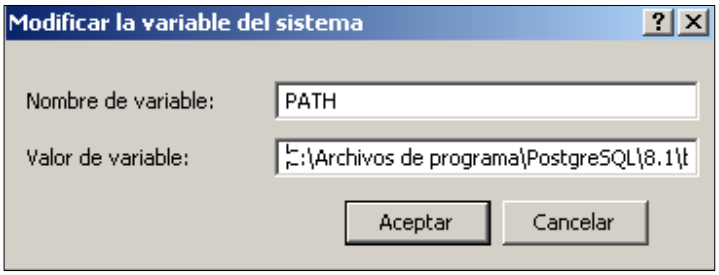


c. El paso a seguir es buscar en el área de Variables del sistema la variable Path, todo con el fin de incluir el PATH del POSTGRES, observe el siguiente gráfico:



Y como lo indica el gráfico será necesario hacer clic en el botón modificar para hacer el nuevo ingreso, lo anterior se logra mediante la siguiente ventana:

d. Para lograr dicho cometido es necesario agregar la cadena " *;C:\Archivos de programa\PostgreSQL\8.1\bin;* " tal y como aparece en el resalto sobre el siguiente gráfico:

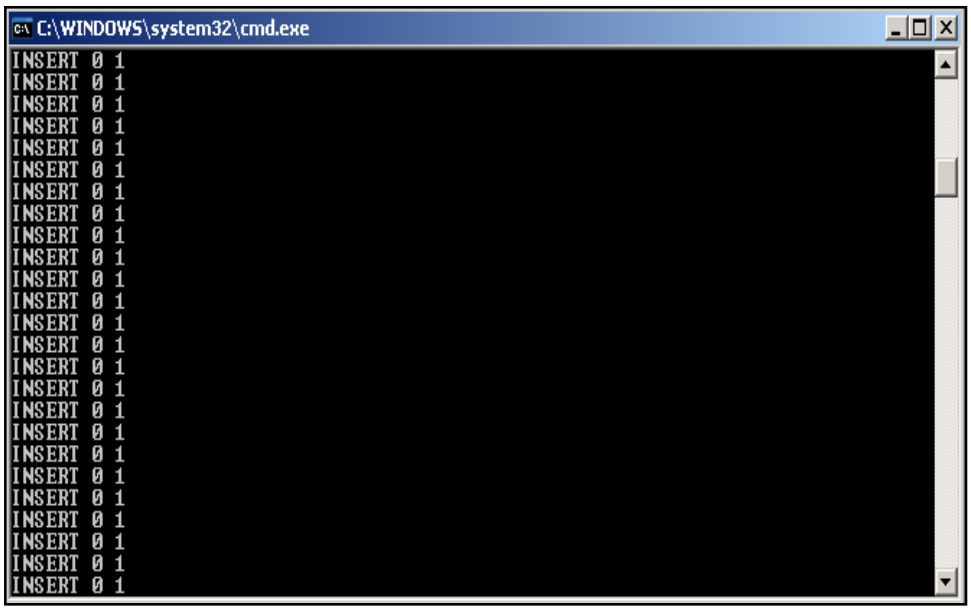


Para terminar el primer paso es necesario aceptar en todas las ventanas.

Segundo que todo que todo es necesario ingresar en el símbolo de sistema para la ejecución del_programa_PSQL. En el momento que se ingrese al símbolo de sistema es necesario entrar en la carpeta C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\WEBapps\ROOT\Secure\sql por medio de los comandos correspondientes. Luego se deberá ingresar el siguiente comando:

Psql –U postgres < i.sql

Donde la U deberá estar dispuesta en mayúsculas, y al terminar la operación aparecerá la siguiente ventana.



Compilación clase base

El software que se está instalando requiere de la configuración de la clase que permite la conexión con la base de datos, ésta deberá ser compilada y copia en la carpeta C:\Archivos de programa\Apache.

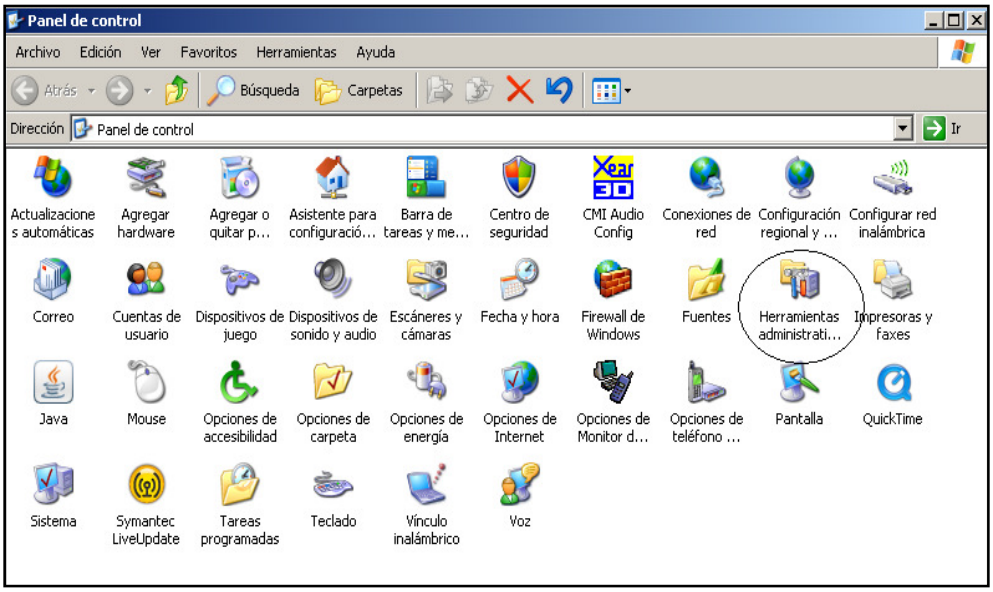
Software Foundation\Tomcat 5.5\WEBapps\ROOT\WEB-INF\classes\Secure, es importante anotar que es necesario crearla, ya que en el sistema no se encuentra.

Luego, y por medio del símbolo de sistema se compilará utilizando un BAT; dicho archivo se encuentra en C:\Archivos de programa\Apache Software Foundation\Tomcat 5.5\webapps\ROOT\Secure\class, el cual tiene como nombre: **c.bat**, el cual contiene las siguientes instrucciones:

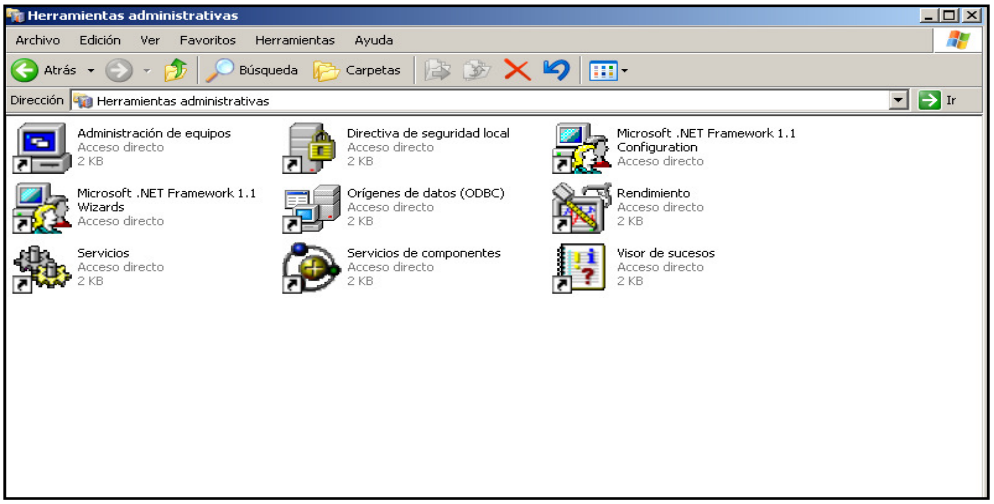
```
echo off
javac co.java
copy co.class ..\..\WEB-INF\classes\Secure
```

Configuración de los DRIVES para los REPORTES

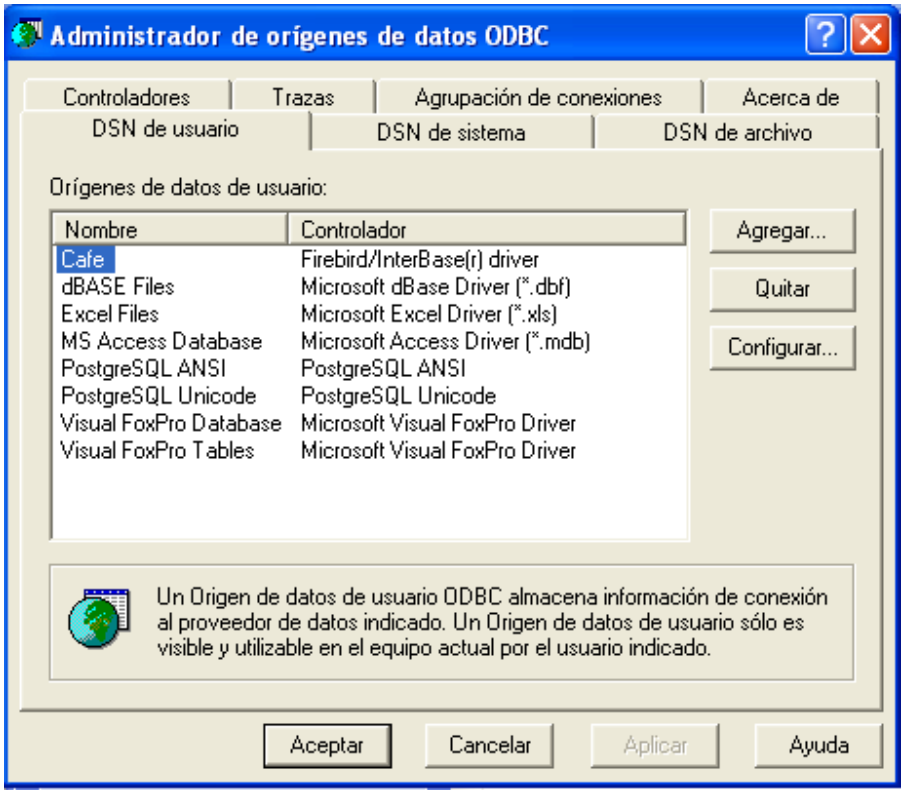
Los reportes hacen conexión con la base de datos mediante ODBC, y para tales efectos es necesario ingresar al panel de control y buscar el icono de Herramientas administrativas, abajo se ilustra un ejemplo de cómo identificarlo:

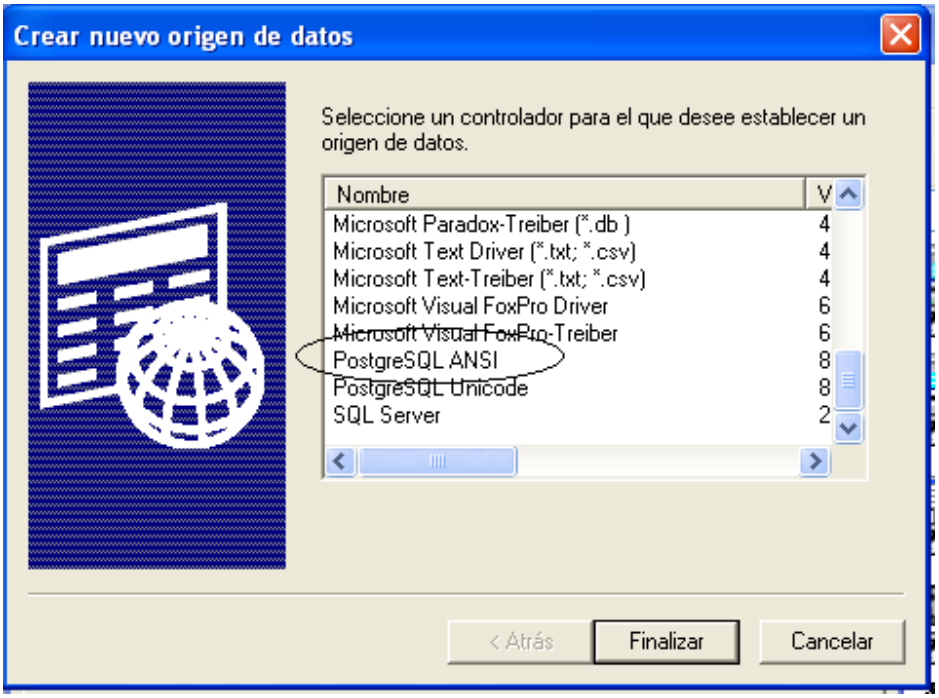


Al momento de hacer clic en dicho icono se desplegará la siguiente ventana:



Y luego se deberá hacer clic en el icono Orígenes de datos (ODBC), para luego ingresar un nuevo elemento mediante la siguiente ventana:





Luego de debe hacer clic en la pestaña DNS de sistema para que aparezca la siguiente ventana:

El paso a seguir buscar el nombre: PostgreSQL ANSI y presionar el botón finalizar, Observar todos los campos y que información contiene, debido a que se deberá ingresar la misma información para lograr un buen efecto en la configuración.

ANEXO F
PAGOS EN IÍNEA

Como objetivo específico, el proyecto plantea la necesidad de plantear el sistema para pagos electrónicos, a continuación se analizan los diferentes sistemas aplicables al proyecto. Es importante denotar que Los métodos del negocio electrónico permiten a las compañías poder conectar sus sistemas informáticos internos y externos en forma más eficiente y flexible, poder trabajar en forma más estrecha con proveedores y socios, y poder satisfacer de mejor forma las necesidades y expectativas de sus clientes[19].

En la práctica, por lo general el *e-business*[20] incluye el *e-comerce*[21]. El objetivo del *e-comerce* es agregar canales de ingreso por medio de la Web Mundial o Internet para construir y mejorar las relaciones con clientes y socios y mejorar la eficiencia. A menudo, el *e- comerce* incluye el uso de sistemas de gestión de conocimiento.

El *e-business* es más que simplemente el *e-comerce*, puesto que implica procesos comerciales que abarcan toda la cadena de valor, como son, la compra electrónica y gestión de la cadena de suministro, procesamiento electrónico de la órdenes, manejo del servicio al cliente, y cooperación con socios comerciales. Existen normas técnicas especiales para el *e-business* que facilitan el intercambio de datos entre las compañías. Las soluciones de software para el *e-business* permiten la integración de los procesos comerciales internos y entre empresas distintas. El *e-business* se puede realizar por medio de la Web, Internet, intranets, *extranets* o alguna combinación de éstos.

Es necesario hacer un análisis de las posibles formas de pagos de productos vía internet, las cuales se definen a continuación.

MONEYBOOKERS

Moneybookers da la oportunidad a cualquier negocio o consumidor con una dirección de email de enviar y recibir pagos online con seguridad y coste reducido en tiempo real.

Entre las características generales:

- Enviar dinero vía email desde su tarjeta de crédito o cuenta bancaria
- Hacer compras online
- Recibir pagos vía email

[19] Negocios en Internet. Referencias conceptuales. [En Línea]. Fecha de Consulta: 09-01-2008. Disponible en : http://www.lektor.cl/definicion.php
[20] Entiéndase por negocios en Internet
[21] Entiéndase por la gestión en términos generales de la comercialización en internet.

Para Colombia, es necesario tener una cuenta en Bancolombia, dónde será consignado a una cuenta determinada el valor de la transacción, pero pagando una comisión a Moneybookers del 3% por cada operación.

Moneybookers ofrece un producto que idealmente satisface a los negocios pequeños, comerciantes online, individuales y otros que no están actualmente siendo servidos de forma efectiva por los métodos de pagos tradicionales. Como una verdadera solución global de pagos.

Moneybookers *Limited* es una compañía de transmisión de dinero regulada bajo la Ley del RU. Pertenece a *Gatcombe Park Ventures Limited*, Londres.

Paypal

PayPal permite a las empresas o consumidores que disponen de correo electrónico enviar y recibir pagos en Internet de forma segura, cómoda y rentable.
La red de PayPal se basa en la infraestructura financiera existente de cuentas bancarias y tarjetas de crédito para crear una solución global de pago en tiempo real.

PayPal tiene la posibilidad de comercializar en nuestro país, solo hace falta tener una tarjeta de crédito de cualquier Banco Colombiano para obtener la cuenta Paypal. Con eso podremos hacer Pagos a usuarios que acepten Paypal, con la ventaja de pagar con la Tarjeta de Crédito.

También se puede utilizar la tarjeta Virtual de Bancolombia e-prepago, con el fin de dar apoyo al proceso.

Paypal proporciona un producto ideal para pequeñas empresas, vendedores por Internet, particulares y otros usuarios no satisfechos con los mecanismos de pago tradicionales.

PayPal (*Europe*) Ltd es una sociedad anónima limitada, registrada en el Reino Unido. PayPal Inc. (la empresa matriz de PayPal (*Europe*) Ltd) pasó a ser propiedad de eBay en octubre de 2002, y está ubicada en California, EE.UU. PayPal ganó en los SIIA *Codie Awards* 2002 el premio a la Mejor solución de comercio electrónico y está reconocido por PC *Magazine* como uno de Los 100 mejores sitios Web.

En términos generales, PayPal es una empresa perteneciente al sector del comercio electrónico por Internet que permite la transferencia de dinero entre usuarios que tengan correo electrónico, una alternativa al tradicional método en papel como los cheques o giros postales. PayPal también procesa peticiones de

pago en comercio electrónico y otros servicios Webs, por los que cobra un porcentaje

Tabla 2. Valores PayPal.

Descripción	Cuenta Personal
Abrir cuenta	Gratis
Enviar dinero	Gratis
Retirar fondos	Gratis para €100,00 EUR o más, €1,00 EUR para €99,99 EUR o menos para cuentas bancarias de España **Tarifas para otros bancos**
Añadir fondos	Gratis
Acepte los siguientes pagos ingresados: • Saldo de PayPal • Transferencia instantánea de PayPal • PayPal eCheck	Gratis
Acepte pagos ingresados con tarjeta de débito o crédito	3,4% más €0,35 EUR. Sin límite de transacciones para pagos enviados por Skype, para el resto hay un límite de 2 transacciones al año 3,4% más €0,35 EUR para pagos con tarjeta recibidos utilizando PayPal en Skype
Transacciones de varias divisas	El tipo de cambio incluye una tarifa del 2,5%*

NETELLER

NETELLER ofrece a sus miembros el método más seguro y sencillo para administrar y transferir fondos. Con miles de comerciantes adheridos, un servicio de atención al cliente a su disposición las 24 horas del día y los 7 días de la semana y con flexibles opciones de depósito y retirada de fondos.

Al crear una cuenta de billetera electrónica NETELLER podrá enviar dinero a cualquiera de los clientes y miembros. Una cuenta de billetera electrónica NETELLER ayuda a evitar la complejidad de cheques y giros, y además protege la confidencialidad de sus datos personales y financieros.

Si se requiere retirar fondos instantáneamente en moneda local, la tarjeta NETELLER Card es el modo más sencillo y más rápido para retirar fondos de su cuenta de billetera electrónica NETELLER.

Dicho sistema esta soportado por Bancolombia, Banco de Bogotá y AvVillas; cada banco opera como ente que recibe las consignaciones y no tiene ningún costo, y el agente de transferencias NETELLER cobra por cada transacción el 2% como comisión en las ventas.

PLATAFORMA SELECCIONADA PARA IMPLANTACIÓN

PLACETOPAY

Para la implementación de Placetopay es necesario disponer del paquete de las clases para cada versión equivalente. Para la implementación es necesario seguir los siguientes pasos:

1. Crear Llave PGP: Crear las llaves PGP, que son necesarias para encriptar la información entre la Plataforma de pagos y el sitio web. Para crear la llave siga el siguiente vinculo:

Describe los estándares seguidos para la generación de llaves en la plataforma.

a) Llave de la plataforma
Deberá generarse trimestralmente un par de llaves para la comunicación con la plataforma y así asegurar la integridad de las mismas. Estas llaves servirán para encriptar los mensajes internos, como para pasar tramas entre cualquier comercio y la plataforma. Se responsabilidad del operador de la plataforma enviar con 15 días de antelación la nueva llave pública a todos los comercios para su instalación.

b). Llave del comercio
Deberá importarse la llave de cada comercio para poder enviar las respuestas encriptadas. Se recomienda a cada comercio cambiarlas por lo menos una vez por año. Será el comercio quien genere su propia llave y envíe la llave pública a la plataforma.

2. Importar y Firmar llave: Una vez creada la llave, es necesario importar la llave de PlacetoPay que se enviara con el paquete de los componentes de Placetopay, la llave PlacetoPay es un archivo extensión .asc y se importara y firmara.

3. Exportar llave: La llave creada para su sitio, una vez firmada, necesita ser exportada como un archivo .asc para ser enviarla por correo.

4. Configurar el componente PlacetoPay: Para el uso adecuado del componente de pago, es necesario asignarle una serie de variables y rutas.

Dentro del paquete Placetopay, existe un archivo llamado **p2p_client,** en este archivo existen unas variables como se muestran a continuación:

Descripción de variables:

☐ **GNUPG_PROGRAM_PATH:** Es la ubicación o directorio en donde se encuentra el comando gnuPG. Para la ilustración corresponde a Windows.

☐ **GNUPG_HOME_DIRECTORY:** Esta variable describe la ubicación en donde se encuentra el anillo de llaves correspondiente a su sitio previamente firmado.

☐ **P2P_CustomerSiteID:** El identificador de sitio, es un código único generado por **EGM** para el uso de PlacetoPay en los sitios. Para solicitar este código, es necesario enviar los datos del comercio a **EGM**, la llave exportada en el paso 3 como archivo adjunto y las direcciones URL donde se espera enviar el formulario y recibir respuesta a la Plataforma de PlacetoPay.

☐ **P2P_KeyID:** Es el identificador de la llave de su sitio, en los 3 primeros pasos, se le indica cual es este identificador de la llave de su sitio.

☐ **P2P_Passphrase:** Es la contraseña que le asigno al momento de crear su llave del sitio.

☐ **P2P_RecipientKeyID:** Es el identificador de la llave de PlacetoPay, sino sabe como obtenerlo puede solicitarlo a las oficinas de **EGM.** Una vez configurados estos datos, se puede proseguir con las pruebas, recuerden que para realizar la implementación del componente PlacetoPay es necesario que utilice las URL de donde se envía y se recibirá la respuesta de PlacetoPay. En el archivo **p2p_client** se encuentran 2 funciones:

a) getPaymentButton(): Esta es la función que enlaza su sitio con el botón de pago de la plataforma de PlacetoPay. Recibe por petición POST 6 parametros, donde 4 son requeridos y 2 son opcionales. Los 4 parametros requeridos son: Reference o la referencia que se va a pagar, TotalAmount o el valor total que se piensa pagar, TaxAmount o el valor correspondiente al IVA, DevolutionBaseAmount o la base de devolucion del IVA y los 2 opcionales son: ShopperName o Nombre del comprador y ShopperEmail o Email del Comprador.
Una vez llamado este método, lo llevara hacia la Plataforma de pago de Placetopay, donde quedara asistido por un asistente para proceder con el pago. Si llamado el método, no genera ninguna respuesta, es necesario

verificar que esta mal configurado o no esta funcionando. Dentro de esta función, hay una línea de código “header('Location: ' . $paymentRequest);”, es necesario comentarla y después de llamar al método **getPaymentRedirect()** del objeto $p2p, se llama la función: getErrorMessage() del objeto $p2p para saber que error puede estar pasando.

b) processTransaction(): Una vez terminado el proceso de pago desde la plataforma de PlacetoPay, se reenviara hacia la URL que se definió en donde se iba a recibir la respuesta del pago. Es aquí en donde se utiliza este método para
saber los datos de la transacción y así asentar la información con los registros de la base de datos de su sitio. El metodo recibe 2 parametros por petición POST que
vienen desde la plataforma de PlacetoPay, el parámetro **CustomerSiteID** que es el identificador del sitio y debe corresponder al asignado en su sitio y el parámetro **PaymentResponse** que es una trama que es procesada por este método para devolver información importante del pago. El cuerpo de este método es modificable para que usted haga la modificación en el registro de su BD y realice su cambio.

5. Realizar la sonda: La sonda, es un proceso que comprueba constantemente o según sus políticas lo definan los registros que se encuentren en estado pendiente con la plataforma de pagos de PlacetoPay para ver si se aprobó ese registro o se declino y posteriormente registrarlo en su DB. Este archivo de sonda se llama
resolvePayment(), y ahí se encuentra un código fuente de referencia para hacer la Sonda. Se configura el identificador del sitio P2P_CustomerSiteID, como se hizo en el paso pasado.

Procedimiento

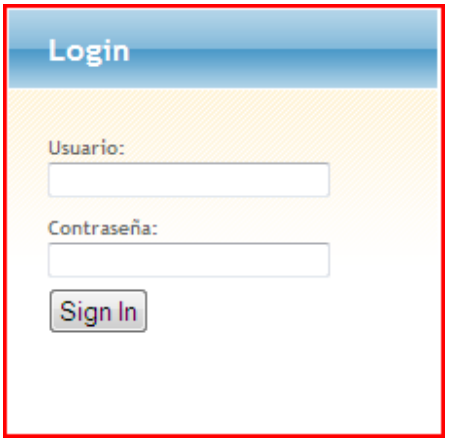
En la actualidad el proceso depende de la disponibilidad presupuestal de la empresa Gilberto Arenas para iniciar la contratación específica para dar continuidad al proceso anteriormente definido.

ANEXO G
MANUAL DE USUARIO

El sistema de información de de Gilberto Arenas está basado en gestión web, por lo que para utilizarlo es necesario contar con un navegador para internet. Luego de ingresar al dicha aplicación usted visualizar una pantalla, tal y como aparece en la figura abajo descrita.

Para ingersar al sistema es necesario tener un usuario y una clave, datos que serán suministrados por el encargado del sistema, si aun no cuenta con una, por favor solicítela en éstas instancias.

Para el ingreso al sistema, se mostrará una pantalla igual a la que aparecerá abajo descrita en la figura.



Luego de suministrar los datos correctos, por favor presione el clic Sign in.

Luego de estar registrado correctamente, aparecerá una imagen idéntica a la ilustrada a continuación

Lo cual indica que ya ingresó al sistema, como característica general, se mostrará un menú generado por el sistema con base al perfil generado para cada usuario, tal y como se observa a continuación

La ilustración anterior muestra un menú al lado izquierdo, el cual podrá usar para ingresar a los submenús.

Menu Administración

El menú administración, se usa para administrar la información base del sistema, tambien puede existir un usuario con un número determinado de persimos para algunas tablas.

Menú Póliza

Una de las opciones fundamentales del sistema correspoden al registro de pólizas, dónde se usa para ingresar y gestionar las diferentes pólizas de la empresa, éste servicio permite administrar los productos de la empresa, a continuación se ilustra mediante una figura

Menú Clientes

Existe un manejo integral para el manejo de clientes, el cual se acerca a un manejo CRM, para tales efectos, se maneja tipos de clientes, nivel cliente, régimen de cliente, estudios mercantiles, entre otros.

Menú Cartera

La cartera de la empresa es básica, la cual se basa en el manejo de movimientos, el menú para la gestión de estos elementos se ilustra en la figura abajo descrita.

Menú Recortes

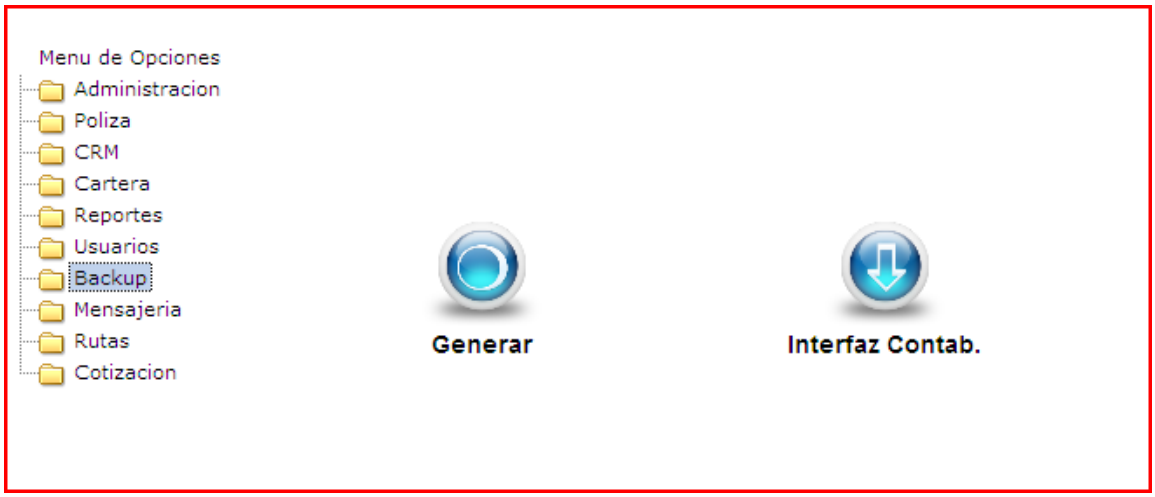
Los reportes de ésta opción, son los generados desde las tablas, y se usan principalmente para consultas en pantalla.

Menú Usuarios

El sistema permite la definición de usuarios y sus perfiles mediante ésta opción.

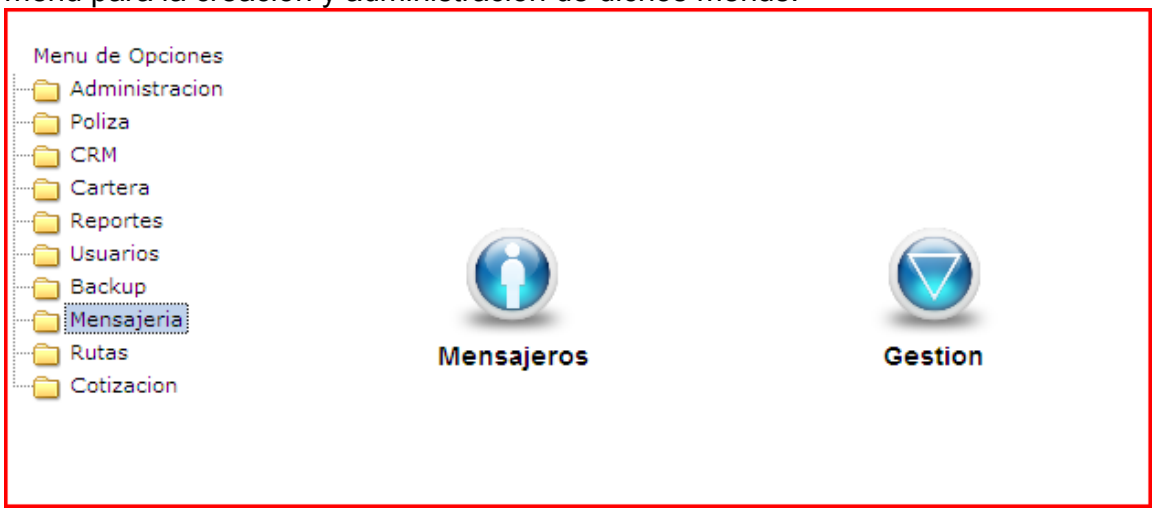
Menú Backup

Una de los requisitos fue la creación de una opción para respaldo, el menú abajo indicado mediante la figura, muestra la forma de ingresar a dichas opciones.



Menú Mensajería

Los mensajeros requieren un control integral de los encargos, es entonces, un menú para la creación y administración de dichos menus.



Menú Rutas

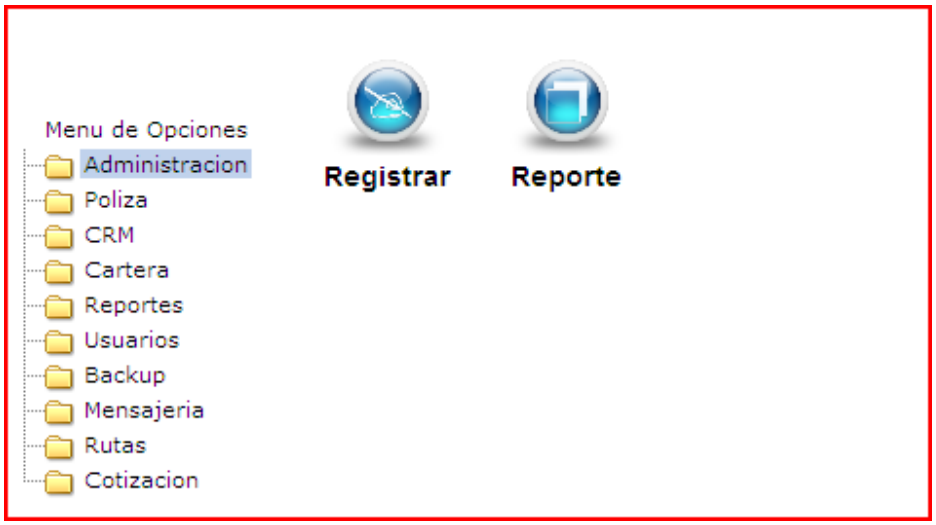
La ruta de negocios es la herramienta base para la creación y definición de los negocios y sus procesos, permitiendo el ingreso, procesos y manejos de asesores.

Menú Cotización

Uno de los objetivos principales del sistema es permitir hacer cotizaciones el línea de los productos.

Ingreso de la información

El ingreso de la información se lleva acabo por medio del siguiente menú

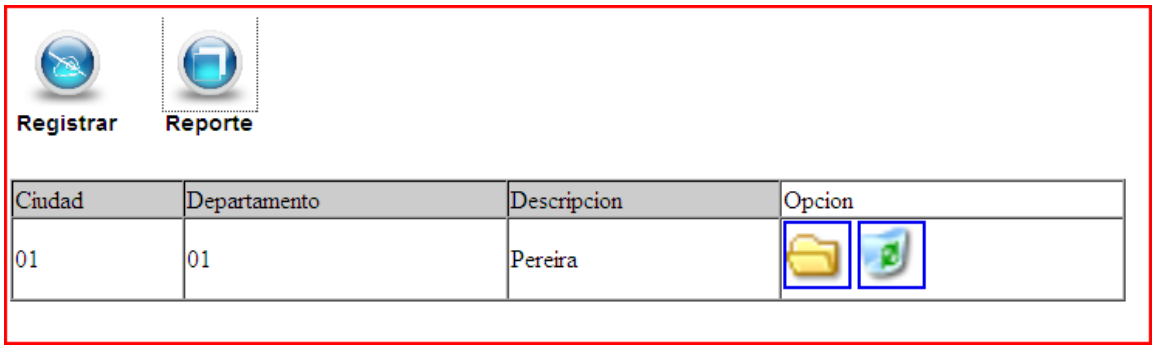


Luego de hacer click en la opción registrar aparecerá la siguiente ventana:

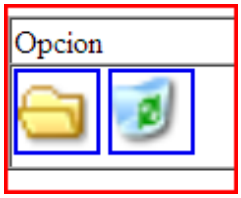
En el caso específico, se tiene un ejemplo para el ingreso de abogados, el cual, perfectamente nos indica la forma de hacerlo; el siguiente paso es ingresar la información correctamente, y luego presionar el botón aceptar.

Listado de la información

El listado de la información se lleva a cabo por medio de la opción pertinente, la cual está representada por la siguiente figura

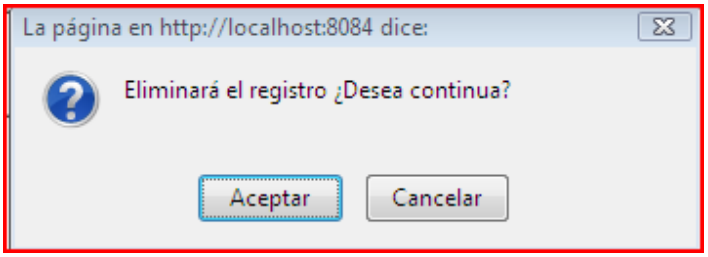


Ésta cuenta con dos opciones, editar y eliminar



Las cuales permiten editar cualquier registro, claro esta que si tiene permisos para hacerlo según el perfil.

Al momento de eliminar presentará una pantalla similar a la siguiente:

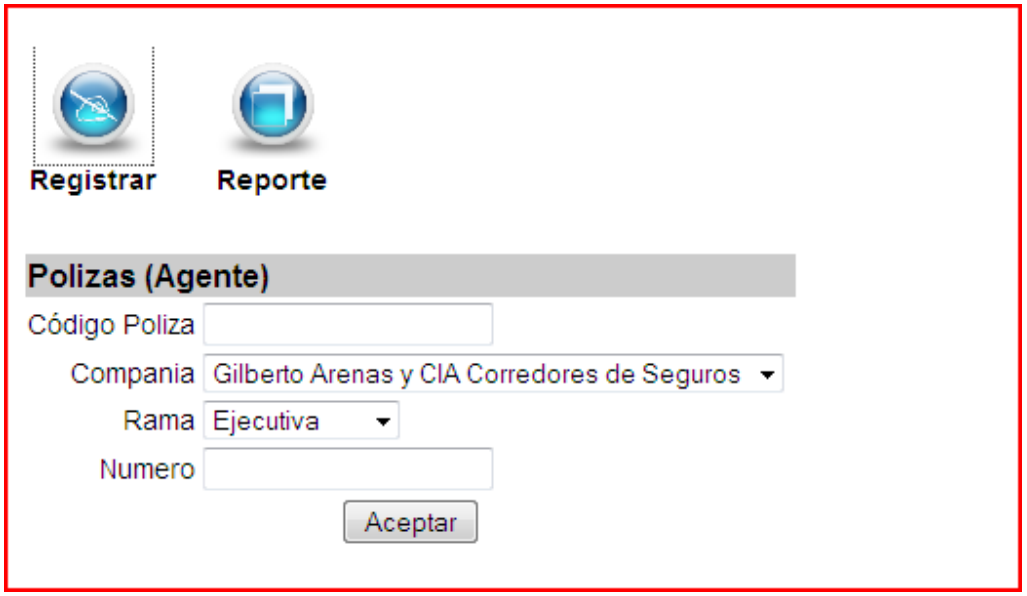


La cual pretende verificar que desea eliminar el registro, es importante anotar que, luego de presionar aceptar el registro no podrá recuperarse.

Pólizas

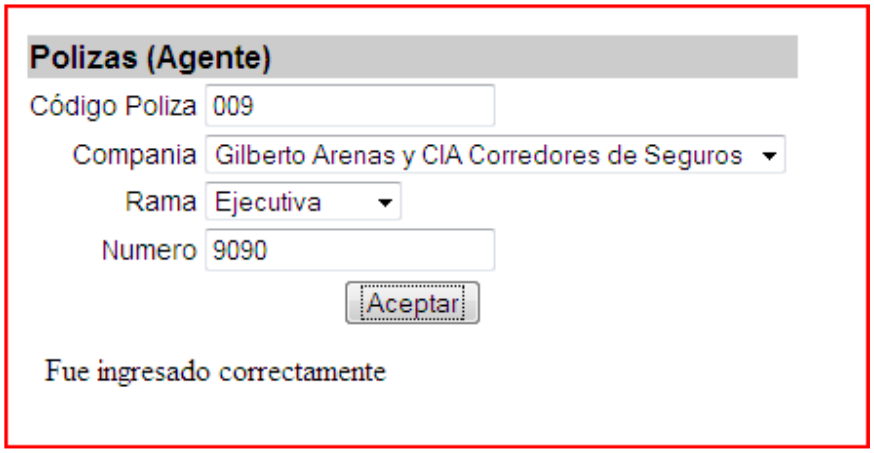
El menú de ingreso de pólizas se mostrará de la siguiente forma:

La cual al momento de presionar el botón Registrar aparecerá la siguiente ventana.

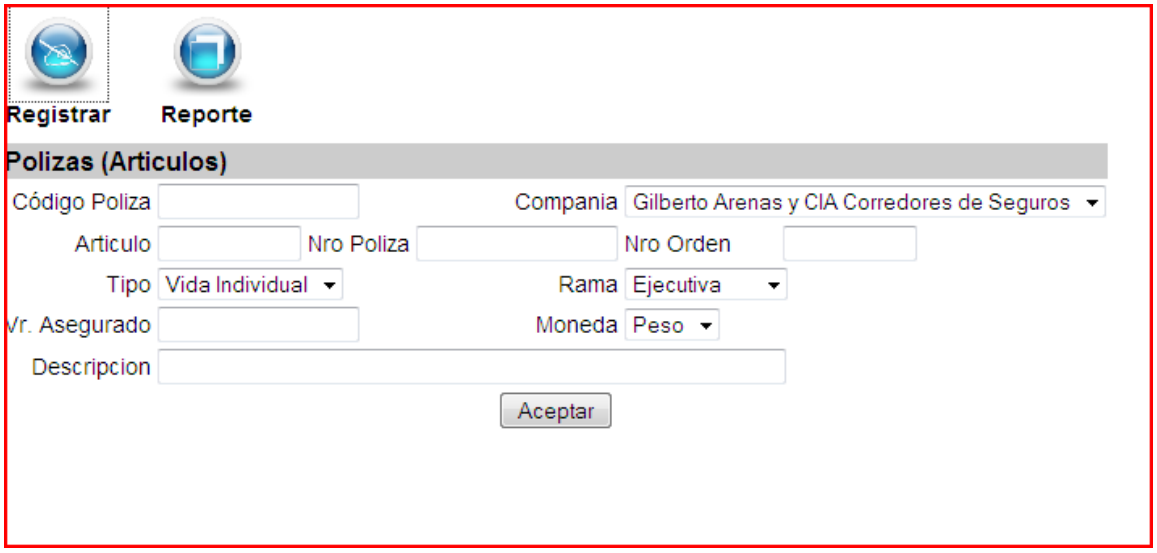


Por lo general el ingreso de las pólizas solicitará los datos correspondientes, con el ánimo de ilustrar el ejemplo, se muestra el ingreso de pòlizas para Agentes.

Luego de ingresar los datos aparecerá una ventana similar a la ilustrada en la figura abajo descrita.



Para ilustrar mejor el proceso de muestra otra ventana para el ingreso de polizas de vida.

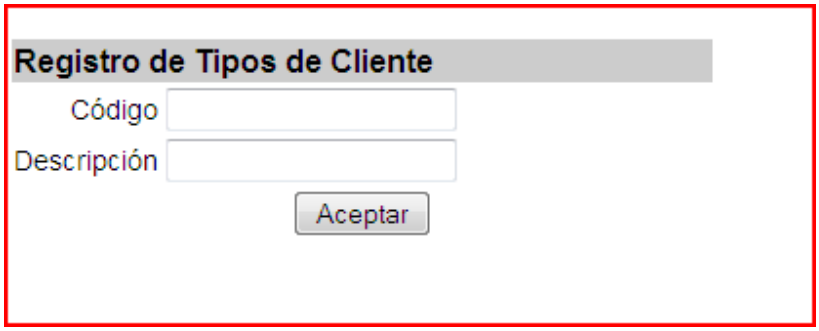


Manejo de clientes

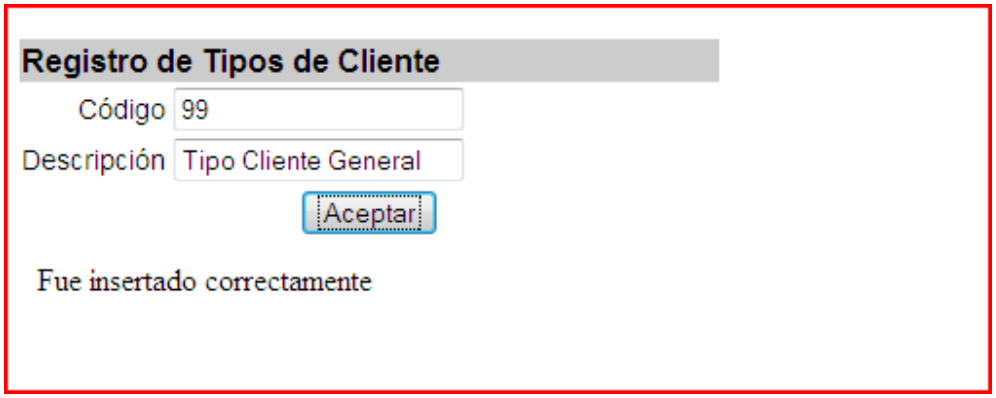
El manejo de clientes se gestiona por las opciones tipos de clientes, sectores, campañas y todo lo relacionado con los clientes.

Para ingresar un tipo de clientes, deberá seleccionar la opción correspondiente, para tales efectos, es necesesario hacer referencia al siguiente menú

Luego aparecerá una ventana que permite el ingreso de dichos elementos, con el fin de ilustrar mejor, se plantea la siguiente ventana:

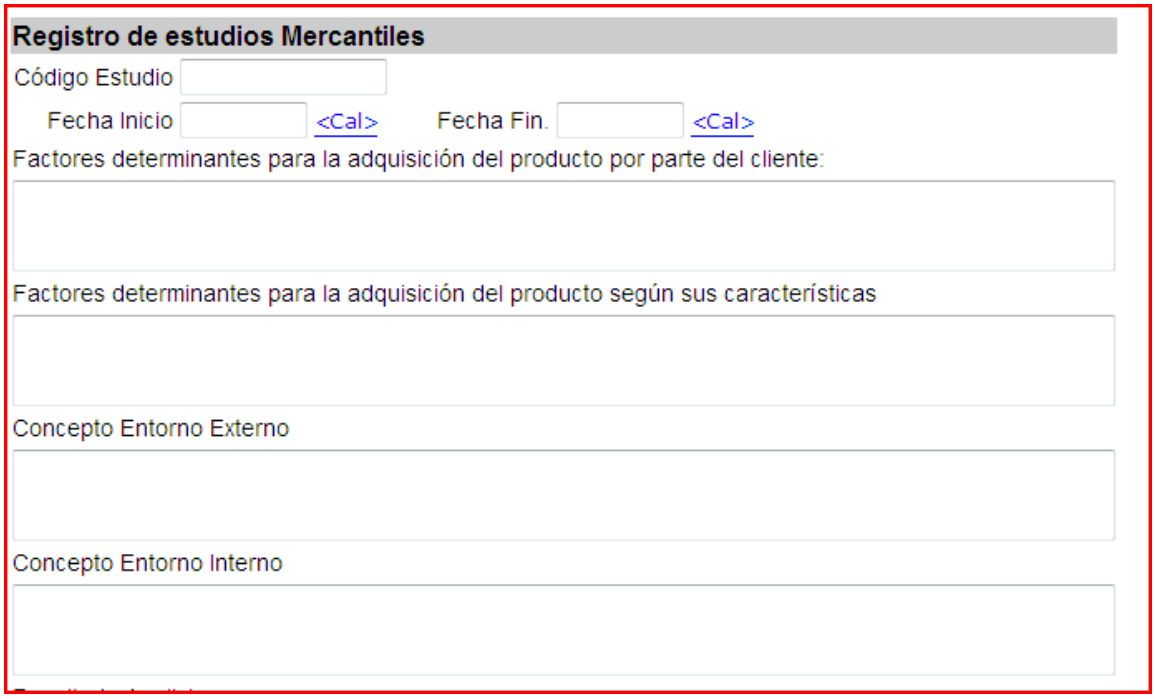


La cual, tiene un botón de Aceptar, que luego de ingresada la información en forma correcta, ingresará a la base de datos el tipo de cliente correcto. Para registrar a efectividad en el proceso, el sistema mostrará la siguiente ventana:



Estudio mercantil.

El se ingresa por medio de la siguiente ventana:



La cual, permite el ingreso de los estudios mercantiles