UNIVERSIDAD DE ORIENTE
NÚCLEO ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UNA APLICACIÓN PARA LA AUTOMATIZACIÓN DEL CONTROL ADMINISTRATIVO DE AFILIADOS Y MONITOREO DE SERVICIOS PARA UNA EMPRESA DE ENCOMIENDAS PRIVADA

Realizado por:
Br. Ramos Aleman, Angel Luis
Br. Roa Quintero, Miguel Alejandro

Trabajo de Grado presentado como requisito parcial para optar al título de
INGENIERO EN COMPUTACIÓN

Puerto La Cruz, Julio de 2010

UNIVERSIDAD DE ORIENTE
NÚCLEO ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UNA APLICACIÓN PARA LA AUTOMATIZACIÓN DEL CONTROL ADMINISTRATIVO DE AFILIADOS Y MONITOREO DE SERVICIOS PARA UNA EMPRESA DE ENCOMIENDAS PRIVADA

Revisado y Aprobado por:

_____

Ing. Gabriela Veracierta
Tutor Académico

Puerto La Cruz, Julio de 2010

UNIVERSIDAD DE ORIENTE

NÚCLEO ANZOÁTEGUI

ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS

DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS

DESARROLLO DE UNA APLICACIÓN PARA LA AUTOMATIZACIÓN DEL CONTROL ADMINISTRATIVO DE AFILIADOS Y MONITOREO DE SERVICIOS PARA UNA EMPRESA DE ENCOMIENDAS PRIVADA

Jurado Calificador:

Ing. Gabriela Veracierta

Tutor Académico

Ing. Pedro Dorta

Jurado Principal

Ing. Rhonald Rodríguez

Jurado Principal

Puerto La Cruz, Julio de 2010

RESOLUCIÓN

De acuerdo al Artículo Nº 41
Del Reglamento de Trabajo de Grado de la Universidad De Oriente:

"LOS TRABAJOS DE GRADO SON DE LA EXCLUSIVA PROPIEDAD DE LA UNIVERSIDAD DE ORIENTE, Y SÓLO PODRÁN SER UTILIZADOS PARA OTROS FINES CON EL CONSENTIMIENTO DEL CONSEJO DE NÚCLEO RESPECTIVO, QUIÉN DEBERÁ PARTICIPARLO PREVIAMENTE AL CONSEJO UNIVERSITARIO, PARA SU AUTORIZACIÓN".

DEDICATORIA

Quiero dedicar este trabajo en primer lugar a mi hija Ambar Valeria, por ser la luz de mis ojos y el regalo más grande que me ha dado Dios, trayendo alegría a mi vida y a mi hogar con cada sonrisa que emana de su rostro.

A mi esposa por siempre estar a mi lado en todo momento cuando lo necesite dándome alientos para seguir adelante y darme la dicha de ser padre.

A mi madre por darme la vida, darme consejos en los momentos en los que necesitaba de ella y siempre prestarme su apoyo aunque muchas veces no estuviera de acuerdo.

A mi padre por ser mi ejemplo a seguir y esforzarse en todo momento por su familia para que nunca faltara nada, sacrificando sus propios beneficios para darnos todo lo que nunca pudo tener y ser mi mejor amigo.

A mi hermano Ricardo que aunque muchas veces discutimos es una persona muy especial en mi vida y ha sido proveedor de muchos momentos de alegría.

Angel Ramos.

DEDICATORIA

A mi madre, Vilma, por haberme enseñado que sin constancia, humildad y perseverancia no se puede llegar lejos, gracias por querer lo mejor para mí, por cuidarme como siempre lo haces, y apoyarme en todo a pesar de la distancia.

A mi padre, Carlos, quien me enseñó la importancia del trabajo y del esfuerzo, ya que los éxitos nunca caen del cielo, sino que se ganan, por inculcarme principios sólidos como la ética, dedicación, honradez y responsabilidad, por ser mi ejemplo a seguir, y por brindarme toda la ayuda que necesite.

A mi familia, en especial a mi tía Rosa y abuela Lucia quienes estuvieron siempre pendientes de mis decisiones y aconsejándome el camino a seguir, así a como a mis hermanos, por siempre seguir de cerca mis pasos.

A Ysa, por ser esa persona tan especial con quien siempre he podido contar, por ofrecerme en todo momento su cariño, amistad, comprensión, por darme alientos para seguir adelante y no rendirme a pesar de las adversidades.

A Luis, Chugui, mis panas, con quienes compartí cada vivencia durante estos años.

A Angel por ser un excelente compañero y amigo con quien compartí la experiencia de elaborar este proyecto.

Miguel Roa

AGRADECIMIENTOS

A mi familia, por estar siempre al pendiente de mí y guiarme para seguir adelante; a mis padres por impulsarme a ser quien soy hoy en día y prestarme siempre todo su apoyo en mis decisiones; a mi esposa e hija por llegar a mi vida en el momento más indicado; a mis hermanos, tíos y primos.

A mi compañero de tesis y amigo (Miguel Roa) por compartir conmigo uno de los momentos más importantes de nuestras vidas y carrera aportando excelentes ideas para hacer de este trabajo lo mejor.

A mis amigos José M. Arias, Eliannys Lugo, Marcel Suarez, Jairo González, Gabriela Noriega, María J. Castañeda, por acompañarme a lo largo de mi carrera y ser excelentes personas.

A mis profesores por haberme impartido sus conocimientos para hacer de mí un gran profesional.

A nuestra tutora la profesora Gabriela Veracierta por prestarnos su colaboración en todo momento y soportarnos por un buen tiempo.

A la Universidad de Oriente “La casa más alta de estudio” por haberme dado la oportunidad de estudiar y haber sido mi hogar durante estos años.

Angel Ramos.

AGRADECIMIENTOS

A la Universidad de Oriente nuestra casa de estudio, por proporcionarnos la infraestructura en la cual tuvimos la oportunidad de estudiar y compartir con nuestros amigos, y por haber sido un hogar durante estos maravillosos años.

A nuestros profesores a quienes lo largo de todos estos años han dejado una gran cantidad de enseñanzas que nos encargamos de aprovechar, y que además han sido claves en nuestra formación académica y formar parte de esta meta, sin ustedes esto no hubiese sido posible.

A la profesora Gabriela por haber sido nuestra tutora académica, quien nos acompaño en la realización de este proyecto, por haber hecho más amena esta experiencia, por brindarnos su invaluable ayuda y total disposición, además, nos ofreció sus valiosos conocimientos y las herramientas necesarias, y por último, pero no de menos importancia, por tenernos paciencia y comprendernos.

A la profesora Zulirais por siempre tener un consejo y ser una mano amiga, y transmitirme siempre esas buenas vibras, además, siempre nos recordaba que hay que ver el lado bueno de las cosas a pesar de las dificultades, y por transmitirnos sus valiosos conocimientos.

A todas esas personas que no nombro pero que tengo muy presente, ya que siempre estuvieron allí apoyándome y brindándome todo su apoyo y ayuda incondicional, así mismo quiero agradecer, a todos aquellos que de una u otra forma me ayudaron a la realización de este proyecto para que

pudiera alcanzar esa meta tan anhelada, como lo es un título de ingeniero en computación, a todos ustedes mis más sinceros agradecimientos.

Miguel Roa

RESUMEN

Las empresas de encomiendas se dedican a prestar los servicios de encomiendas y transporte de paquetería. En un mercado tan competitivo como el del correo privado, donde coexisten tantas firmas dedicadas a la misma actividad, tienen que saber ganarse el gusto y preferencia de los clientes gracias a la filosofía de calidad, eficiencia y rapidez, que envuelve cada uno de los procesos operativos de la marca, la garantía de entrega y el equipo humano. Atendiendo a esta problemática se propuso el desarrollo de un software para la automatización del control administrativo de los afiliados y el monitoreo de paquetes. En este informe se presenta el diseño de un Sistema de Control Administrativo de Afiliados y Monitoreo de Paquetes (*SCAAMP*) que apoyará los procesos descritos anteriormente. Este es un proyecto basado en el Proceso Unificado de desarrollo de software, bajo el Lenguaje Unificado de Modelado (*UML*), para ser implantado bajo cualquier plataforma, programado con el lenguaje de programación Java y cuyos datos son almacenados en una base de datos PostgreSQL.

ÍNDICE

CAPÍTULO I
EL PROBLEMA

INTRODUCCIÓN

El ser humano en la búsqueda de satisfacer sus necesidades ha hecho uso de los recursos que le ofrece el medio, el cual ha ido paulatinamente transformándose como respuesta a exigencias cada vez mayores de los bienes materiales. Esta transformación ha dado paso a la tecnología, una revolución que tiene lugar en la actualidad, que comenzó en silencio y ha evolucionado hasta comprender mayor parte del mundo, convirtiéndose de esta manera en el corazón de las actividades cotidianas. Hoy en día el uso de aplicaciones informáticas para la administración, procesamiento y distribución de la información en una organización, se hace cada vez más indispensable.

Estas aplicaciones permiten lograr ahorros significativos en tiempo y mano de obra, debido a que automatizan tareas operativas de la organización y ofrecen un gran apoyo en el proceso de toma de decisiones que permiten, entre otras cosas, lograr ventajas competitivas en el momento de la implantación y uso de la aplicación.

Una aplicación informática es un sistema abierto, es decir interactúa con su ambiente mediante el intercambio de información y se adapta a las necesidades del ambiente que lo contenga; permite el uso de computadoras que automatizan los procesos rutinarios, su entrada está constituida por datos y su salida por información.

1.1 PLANTEAMIENTO DEL PROBLEMA

En los diferentes sectores de la economía, las empresas juegan un papel fundamental para la productividad y el desarrollo económico y social. Según su tipo esta participación puede darse en determinado nivel o alcance, es decir, que de acuerdo con sus características, las empresas pueden tener un determinado crecimiento y desempeño o verse condicionadas por factores que limitan su permanencia y consolidación en los mercados locales, regionales, nacionales e internacionales.

Debido a lo difícil que es en ocasiones establecerse y mantenerse en el mercado, algunas empresas se ven en la necesidad de enviar o traer mercancía de otros lugares, abarcando de esta manera un mercado más amplio, y obtener mercancía que la competencia no posee, logrando así competir con otras empresas, consolidándose en el mercado y satisfaciendo las necesidades de los distintos sectores de la población.

Es por ello que algunas empresas e inclusive personas se ven en la necesidad de contactar empresas dedicadas al traslado de encomiendas, ya que de esta manera pueden recibir y enviar, productos, mercancía o materia prima, de un lugar a otro, donde no se cuente con ellos o no puedan ser adquiridos con facilidad.

Uno de los aspectos más importantes y que son tomados en cuenta por el cliente a la hora de seleccionar la empresa de encomiendas con la que desea trabajar o realizar sus envíos es, la calidad del servicio, la cual se ve reflejada en el control y seguimiento de los paquetes o encomiendas, así como también la rapidez con la que efectúan los envíos y entregas de los mismos.

Por ende, este tipo de empresas deben contar con un software que pueda brindar dicha confiablidad tanto al cliente como a ella misma, ya que para la firma también es de suma importancia tener no sólo el control y seguimiento de los paquetes que son enviados y recibidos, sino también todo lo concerniente a la forma de pago y cancelación de los mismos; así como el registro de los socios o transportistas a los cuales se le hace responsable del traslado y entrega de dichas encomiendas, lo cual facilitaría la gestión administrativa de la misma.

De acuerdo con lo antes expuesto, la finalidad de este proyecto fue el desarrollo de una aplicación, que facilite a la empresa el manejo y control de la información anteriormente descrita, para que de esta manera pueda tener un mejor desempeño a nivel competitivo. Además de esto, el uso de la aplicación producirá ventajas inherentes tales como la reducción de costos, la optimización de los recursos.

El software a diseñar en este proyecto estuvo constituido por una serie de módulos, los cuales proporcionarán a la empresa un manejo eficiente sobre todo lo concerniente a la administración y control, entre los cuales tendría, registro de: envíos, afiliados, rutas, unidades de transporte, clientes; consulta y rastreo de encomiendas, administración de remesas (relación de envíos transportados por unidad) por socio, entre otros.

Para el desarrollo del proyecto se utilizó la metodología del Proceso Unificado Racional (*Rational Unified Process* en inglés), junto con el Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*). Esta metodología proporciono una guía para elaboración de distintos diagramas y artefactos que facilitará la ejecución del proceso de elaboración de un Software, ya que describen desde

diferentes perspectivas orientadas a los diferentes involucrados en el proyecto.

Las herramientas a utilizadas para la implementación del proyecto fueron el lenguaje de programación Java y el sistema gestor de base de datos PostgreSQL.

1.2 OBJETIVOS

1.2.1 Objetivo General

Desarrollar una aplicación para la automatización del control administrativo de afiliados y monitoreo de servicios para una empresa de encomiendas privada.

1.2.2 Objetivos Específicos

- Identificar los requisitos necesarios para la elaboración del proyecto.
- Definir los elementos necesarios que conformaran la arquitectura del sistema.
- Establecer la arquitectura para la elaboración del software.
- Diseñar la base de datos e interfaz que permitan el manejo de la información de manera clara y sencilla.

- Codificar los distintos módulos requeridos para el desarrollo del proyecto.

- Realizar las pruebas de integración del software desarrollado.

CAPÍTULO II
MARCO TEÓRICO

INTRODUCCIÓN

En este capítulo, se definen los antecedentes que son la base para la realización del proyecto, además se describen los fundamentos teóricos necesarios para el entendimiento y comprensión del mismo, con la finalidad de tener una introducción a las herramientas de diseño y permitir al lector poseer una idea de la naturaleza del contenido del resto del documento.

2.1 ANTECEDENTES DE LA INVESTIGACIÓN

Para efectuar el presente trabajo de investigación se utilizará la metodología del proceso unificado de desarrollo de software (*RUP*) y el lenguaje unificado de modelado (*UML*), como herramientas para diseñar una aplicación que permita automatizar las actividades relacionadas con control administrativo de socios y rastreo de paquetes para una empresa de encomiendas. En el departamento de Computación y Sistemas de la Universidad de Oriente, Núcleo Anzoátegui, se han realizado proyectos que facilitan el manejo administrativo o control de información de instituciones y empresas, los cuales han utilizado estas técnicas y metodologías, lo cual permitió contribuir en el desarrollo de esta investigación. A continuación se nombrarán algunos de estos proyectos:

- Pérez, L (2003). Realizó el **"Desarrollo de un Software para Automatizar las Actividades de Control Interno de una Empresa de Transporte de Alimentos"**, como requisito parcial para optar al

título de Ingeniero en Computación en la Universidad de Oriente – Núcleo Anzoátegui. Para el desarrollo del proyecto se utilizó la metodología de Proceso Unificado Racional (*RUP*) conjunto con el Lenguaje Unificado de Modelado (*UML*). El sistema se diseñó para organizar y automatizar de manera eficiente las actividades de control interno de la empresa de transporte único. El sistema permite mostrar consultas y reportes referentes a la empresa, específicamente con respecto al control de unidades, viajes realizados por cada unidad, la generación de pagos, etc.

- Arcila, A. y Zacarías, E. (2006). Realizaron el **"Desarrollo de un Software como Soporte para la Automatización de las Actividades Administrativas que se llevan a cabo en una Institución Educativa"**, como requisito parcial para optar al título de Ingeniero en Computación en la Universidad de Oriente – Núcleo Anzoátegui. Este trabajo permitió la automatización de los procesos administrativos de la guardería preescolar "Trencito mi Rosita Mística", llevándose a cabo las siguientes gestiones: Inscripción de Alumnos, Facturación, Contratación de Personal, Pago de Nóminas, Relación de Ingresos vs Egresos, entre otras. El desarrollo del proyecto se guió por la metodología de Proceso Unificado Racional (*RUP*).

- Tabare, M (2006). Realizó el **"Desarrollo de un Software que Permite la Adquisición y Procesamiento de Datos Asociados a la Producción de Barras de Acero en una Empresa Siderúrgica"**, como requisito parcial para optar al título de Ingeniero en Computación en la Universidad de Oriente – Núcleo Anzoátegui. El objetivo principal era automatizar la captura de datos asociados a la

producción de barras de acero y unificar los diversos sistemas empleados en la obtención de datos de proceso y producción, usando el proceso unificado de desarrollo de software.

- Salazar F. (2007). Realizó el **"Desarrollo De Un Software Para La Automatización Del Control De Entrada Y Salida De Bienes, Equipos Y Materiales En Una Empresa Cervecera"**, como requisito parcial para optar al título de Ingeniero en Computación en la Universidad de Oriente – Núcleo Anzoátegui. Este proyecto permitió la automatización de los procesos de control, para la creación de pases de entrada, salida y búsqueda de pases ya creados, se utilizó la metodología del Proceso Unificado de Desarrollo de Software. La arquitectura se construyó utilizando el lenguaje de programación Visual Basic 6.0.

- Pacheco R. (2008). Realizó el **"Desarrollo de una aplicación para automatizar los módulos de compra y venta de un sistema administrativo para una empresa bajo la licencia de software libre"**, como requisito parcial para optar al título de Ingeniero en Computación en la Universidad de Oriente – Núcleo Anzoátegui. Este trabajo de investigación se llevó a cabo para diseñar y desarrollar un módulo para la compra y venta de insumos y/o productos que se integre a un sistema administrativo, utilizando la filosofía de software libre.

- Lugo E. y Pino M. (2009). Realizaron el **"Desarrollo de un Software Gerencial para los Procesos Administrativos del Área de Postgrado del Núcleo de Anzoátegui de la Universidad de Oriente"**, como requisito parcial para optar al título de Ingeniero en

Computación en la Universidad de Oriente – Núcleo Anzoátegui. Este proyecto se realizó con la finalidad de mejorar la integración y control de los procesos Administrativos del Área de Postgrado del Núcleo de Anzoátegui de la Universidad de Oriente, desarrollando una aplicación gerencial bajo ambiente Web.

2.2 FUNDAMENTOS TEÓRICOS

2.2.1 Transporte (Marín, 2009)

Se puede entender y emplear como la acción de llevar bienes o personas, de un lugar a otro, también se define como el conjunto de los diversos modos de traslado de bienes y personas.

2.2.1.1 Clasificación del Transporte

- **Según su ámbito de operaciones**
  - **Nacional**: Es aquel autorizado para efectuar transporte entre distintos puntos del territorio nacional.
  - **Internacional**: Es el autorizado para efectuar transporte con el exterior desde el territorio nacional, es decir intercambio de naciones.
  - **Mixto**: Es el autorizado para efectuar transporte, tanto en distintos puntos del territorio nacional, como el exterior.

- **Según la nacionalidad de su matrícula**
  - **Nacional:** Si su matrícula ha sido expedida en el país.
  - **Extranjera**: Si su matrícula ha sido expedida por otro país.

- **Según el medio natural donde opera**
  - **Marítimo**: Es la llamada navegación por mar, ríos (fluvial, lagos).
  - **Aéreo**: Es la navegación por aire aviones helicóptero, globos etc.
  - **Terrestre**: Son vehículos proyectados para circular en tierra.

- **Según su función de carga**
  - En el transporte de mercancías y de bienes y servicios.
  - Transporte de pasajero.
  - Encomienda o transporte de pequeños bultos.
  - Vehículos de guerra.

- **Según su modo**
  - **Ordinario**: Un sólo modo de transporte (marítimo, aéreo terrestre y ferroviario).
  - **Transporte Multimodal, intermodal o combinado**: Son dos o más modos de transporte en una operación de tránsito aduanero.

- **Según los operadores de transporte**
  - **El Porteador:** Es el que se encarga de cualquier modo que sea, para efectuar o hacer efectuar el transporte
  - **El Transportista:** Es el que efectivamente realiza el transporte
  - **El Consolidador:** Es un operador distinto del porteador que transporta por intermedio de éste, mercancías o carga en forma agrupada, bajo su propio nombre y responsabilidad destinado a uno o varios destinatarios finales
  - **Mensajería Internacional:** “Couriere” Comprende el transporte expreso de correspondencia documentos y encomiendas consignado a una empresa operadora de mensajería internacional, para ser entregados

a terceras personas, bajo la modalidad puerta a puerta por vía aérea, terrestre o marítima

2.2.1.2 Transporte Comercial (Howard y Freeman, 2009)

El transporte comercial moderno está al servicio del interés público e incluye todos los medios e infraestructuras implicadas en el movimiento de las personas o bienes, así como los servicios de recepción, entrega y manipulación de tales bienes.

2.2.1.3 Guía de Encomienda (Arranz, 2001)

Documento que expide una empresa de transporte de encomienda (terrestre), como constancia de haber recibido del embarcador, determinadas mercancías para transportarlas hasta su destino. Tiene carácter de contrato de transporte de carga.

2.2.2 Automatización (Sánchez, 2009)

2.2.2.1 Definición

La automatización es un sistema donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.

Un sistema automatizado consta de dos partes principales:

- La **Parte Operativa** es la parte que actúa directamente sobre la máquina. Son los elementos que hacen que la máquina se mueva y realice la operación deseada. Los elementos que forman la parte operativa son los accionadores de las máquinas como motores, cilindros, compresores y los captadores como fotodiodos, finales de carrera.
- La **Parte de Mando** suele ser un autómata programable (tecnología programada), aunque hasta hace bien poco se utilizaban relés electromagnéticos, tarjetas electrónicas o módulos lógicos neumáticos (tecnología cableada). En un sistema de fabricación automatizado el autómata programable esta en el centro del sistema. Este debe ser capaz de comunicarse con todos los constituyentes de sistema automatizado.

2.2.2.2 Objetivos de la Automatización

- Mejorar la productividad de la empresa, reduciendo los costes de la producción y mejorando la calidad de la misma.
- Mejorar las condiciones de trabajo del personal, suprimiendo los trabajos penosos e incrementando la seguridad.
- Realizar las operaciones imposibles de controlar intelectual o manualmente.
- Mejorar la disponibilidad de los productos, pudiendo proveer las cantidades necesarias en el momento preciso.
- Simplificar el mantenimiento de forma que el operario no requiera grandes conocimientos para la manipulación del proceso productivo.
- Integrar la gestión y producción.

2.2.3 Programación Orientada a Objetos (Joyanes, 2003)

2.2.3.1 Definición

La programación orientada a objetos o POO (*OOP, de sus siglas en inglés*) es un método de implementación en el que los programas se organizan como colecciones cooperativas de objetos, cada uno de los cuales representan una instancia de alguna clase, y cuyas clases son todos miembros de una jerarquía de clases unidas mediante relaciones de herencias.

Existen tres importantes partes en la definición de la programación orientada a objetos:

1. Utiliza objetos, no algoritmos, como bloques de construcción lógicos (jerarquía de objetos).
2. Cada objeto es una instancia de una clase.
3. Las clases se relacionan unas con otras por medio de relaciones de herencia.

Un programa puede parecer orientado a objetos, pero si cualquiera de estos elementos no existe, no es un programa orientado a objetos. Específicamente, la programación sin herencia es distinta de la programación orientada a objetos; se denomina programación con tipos abstractos de datos o programación basada en objetos.

2.2.3.2 Principios de los Lenguajes Orientados a Objetos

Para que un lenguaje se llame a sí mismo orientado a objetos debe soportar tres conceptos: objetos, clases y herencia. Sin embrago, ha llegado a pensarse más comúnmente que los lenguajes orientados a objetos son lenguajes construidos sobre el trípode encapsulación, herencia y polimorfismo. La razón de este cambio de filosofía es que con el paso de los años, los desarrolladores de software han llegado a darse cuenta que la encapsulación y el polimorfismo son partes tan integrantes de la construcción de sistemas orientados a objetos como la clase y la herencia.

- **Encapsulación:** Es una característica muy potente, y junto con la ocultación de la información, representan el concepto avanzado de objeto, que adquiere su mayor relevancia cuando encapsula e integra datos, más las operaciones que manipulan los datos en dicha entidad.
- **Herencia:** Es una propiedad que le permite a los objetos ser construidos a partir de otros objetos. Dicho de otro modo, la capacidad de un objeto para utilizar las estructuras de datos y los métodos previstos en antepasados o ascendientes. El objetivo final es la reutilizabilidad o reutilización, es decir, reutilizar código anteriormente ya desarrollado. La herencia se apoya en el significado de ese concepto en la vida diaria, así las clases básicas o fundamentales se dividen en subclases, es decir, supone una clase base y una jerarquía de clases que contiene las clases derivadas de la clase base.
- **Polimorfismo:** Es la propiedad por la cual un mismo mensaje puede actuar de diferente modo cuando actúa sobre objetos diferentes ligados por la propiedad de la herencia. La gran ventaja ofrecida por el polimorfismo es permitir que los nuevos tipos de datos sean

manipulados de forma similar que los tipos de datos predefinidos, logrando así ampliar el lenguaje de programación de una forma más ortogonal, en un sentido más general supone que un mismo mensaje puede producir acciones (resultados) totalmente diferentes cuando se recibe por objetos diferentes.

2.2.3.3 Conceptos Fundamentales de la OOP

La programación orientada a objetos es una nueva forma de programar que trata de encontrar una solución a estos problemas. Introduce nuevos conceptos, que superan y amplían conceptos antiguos ya conocidos. Entre ellos destacan los siguientes:

- **Clase**: definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas, (de C A D). Es la facilidad mediante la cual la clase D ha definido en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D.
- **Objeto**: entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.
- **Método**: algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un "mensaje". Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del

objeto, o la generación de un "evento" con un nuevo mensaje para otro objeto del sistema.

- **Evento**: un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente. También se puede definir como evento, a la reacción que puede desencadenar un objeto, es decir la acción que genera.
- **Mensaje**: una comunicación dirigida a un objeto, que le ordena que ejecute uno de sus métodos con ciertos parámetros asociados al evento que lo generó.
- **Propiedad o atributo**: contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.
- **Estado interno**: es una variable que se declara privada, que puede ser únicamente accedida y alterada por un método del objeto, y que se utiliza para indicar distintas situaciones posibles para el objeto (o clase de objetos). No es visible al programador que maneja una instancia de la clase.
- **Componentes de un objeto**: atributos, identidad, relaciones y métodos.
- **Representación de un objeto**: un objeto se representa por medio de una tabla o entidad que esté compuesta por sus atributos y funciones correspondientes.

2.2.4 Lenguaje de Programación Java (Joyanes, 2002)

2.2.4.1 Definición

Es un lenguaje de programación, que utiliza desarrollo orientado a objetos. Su sintaxis está basada en C++, pero elimina la mayoría de los aspectos oscuros de este lenguaje: se elimina el preprocesador de C, las funciones, y los punteros.

En Java no existen las funciones, salvo como métodos de acceso a una clase. Las variables siempre están incluidas dentro de clases, para favorecer la encapsulación del código. Sus librerías de objetos predefinidas proporcionan herramientas para el desarrollo de las aplicaciones. Los objetos obtienen un espacio de almacenamiento durante su creación, que es recuperado automáticamente cuando se destruyen.

Las aplicaciones Java pueden tener varias líneas de ejecución (hilos) concurrente. La multitarea por hilos está implementada en el propio lenguaje

En la actualidad, Java se utiliza para desarrollar aplicaciones empresariales a gran escala, para mejorar la funcionalidad de servidores web, para proporcionar aplicaciones para los dispositivos domésticos y para muchos otros propósitos.

2.2.4.2 Características básicas Java (Froufe, 1999)

Las características principales que nos ofrece Java respecto a cualquier otro lenguaje de programación, son:

- **Es Simple:** Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos. C++ es un lenguaje que adolece de falta de seguridad, pero C y C++ son lenguajes más difundidos, por ello Java se diseñó para ser parecido a C++ y así facilitar un rápido y fácil aprendizaje.
- **Es Orientado A Objetos**: Java trabaja con sus datos como objetos y con interfaces a esos objetos. Soporta las tres características propias del paradigma de la orientación a objetos: encapsulación, herencia y polimorfismo. Las plantillas de objetos son llamadas, como en C++, *clases* y sus copias, *instancias*.
- **Es Distribuido**: Java se ha construido con extensas capacidades de interconexión TCP/IP. Existen librerías de rutinas para acceder e interactuar con protocolos como *http* y *ftp.* Esto permite a los programadores acceder a la información a través de la red con tanta facilidad como a los ficheros locales.
- **Es Robusto**: Java realiza verificaciones en busca de problemas tanto en tiempo de compilación como en tiempo de ejecución. Lo cual ayuda a detectar errores, lo antes posible, en el ciclo de desarrollo.
- **Es de Arquitectura Neutral**: Para establecer Java como parte integral de la red, el compilador Java compila su código a un fichero objeto de formato independiente de la arquitectura de la máquina en que se ejecutará. Cualquier máquina que tenga el sistema de ejecución (*run-time*) puede ejecutar ese código objeto.
- **Es Seguro**: En Java, características como los punteros o el *casting* implícito que hacen los compiladores de C y C++ se eliminan para prevenir el acceso ilegal a la memoria.
- **Es Portable**: Java implementa estándares de portabilidad para facilitar el desarrollo. Además, Java construye sus interfaces de usuario a

través de un sistema abstracto de ventanas de forma que las ventanas puedan ser implantadas en entornos.

- **Es Interpretado**: El intérprete Java (*sistema run-time*) puede ejecutar directamente el código objeto. Enlazar (*linkar*) un programa, normalmente, consume menos recursos que compilarlo.
- **Es Multihilo**: Al ser multihilvanado, Java permite muchas actividades simultáneas en un programa. Los hilos, son básicamente pequeños procesos o piezas independientes de un gran proceso.
- **Es Dinámico**: Java se beneficia todo lo posible de la tecnología orientada a objetos. Java no intenta conectar todos los módulos que comprenden una aplicación hasta el tiempo de ejecución.

2.2.4.3 Entorno de Desarrollo Java (Lemay, 2003)

Para desarrollar código Java se requiere algún paquete de programación Java. La compañía Sun Microsystems, creadora de Java distribuye gratuitamente el Java Development Kit (*JDK*), la cual se trata de un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar programas en Java.

Otra alternativa son los IDE (*Integrated Development Environment*), por sus siglas en ingles, son entornos de desarrollo integrado, que ofrecen un ambiente gráfico en los que se tiene acceso a las herramientas del JDK por lo que es posible escribir, compilar y ejecutar código Java.

Ejemplo de algunos IDE’s son:

- NetBeans Open-Source

- Eclipse Open-Source
- Forte de Sun
- JBuilder de Borland
- JCreator de Xinox
- JDeveloper de Oracle

2.2.5 Entorno de Desarrollo Integrado NetBeans (Boudreu, Tulach y Wielenga, 2004)

2.2.5.1 Definición

NetBeans es un IDE para el lenguaje de programación Java principalmente, esto significa que precisa que tenga instalado Java en su equipo de trabajo, por lo que hay que tener instalado en el equipo de desarrollo el JDK que incluye la maquina Virtual. Las últimas versiones de NetBeans IDE cuentan también con soporte para otros lenguajes como C++, PHP y Ruby.

2.2.5.2 NetBeans vs otros IDE's

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las API's de NetBeans y un archivo especial (*manifest file*) que lo identifica como módulo. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

2.2.5.3 Características del NetBeans

Algunas de las características que hacen interesante la elección de NetBeans Platform como plataforma para la aplicación son las siguientes:

- Los proyectos desarrollados no dejan de ser multiplataforma, y poseen largadores (*launchers*) para cada plataforma.
- Sistema de ventanas práctico para desarrollar las interfaces de usuario
- Sistema de ficheros virtual en el cual se van montan los diferentes módulos con el cual se van adaptando automáticamente los menús, barra de herramientas, menús contextuales, etc. de la aplicación
- Su licencia permite construir tanto aplicaciones open source como comerciales
- Compatibilidad con Java Web Start
- No es obligatorio que una aplicación deba tener interfaz de usuario gráfica (*GUI*), ya que la plataforma permite dejar de lado la misma y seguir disfrutando del resto de los beneficio, por ejemplo la actualización de módulos desde un repositorio remoto.
- Soporte completo para desarrollar desde NetBeans IDE, por lo que no se necesitará otra herramienta adicional para el desarrollo.

2.2.6 Sistema de Base de Datos **(Elmasri y Navathe, 1997)**

2.2.6.1 Definición

Una Base de Datos es un conjunto de datos relacionados entre sí. Este conjunto de datos debe ser lógicamente coherente, con cierto significado inherente. Una colección aleatoria de datos no puede considerarse

propiamente una base de datos. Toda base de datos se diseña y se construye con datos para un propósito específico. Está dirigida a un grupo de usuarios y tiene ciertas aplicaciones preconcebidas que interesan a dichos usuarios.

Una base de datos tiene una fuente de la cual se derivan los datos, cierto grado de interacción con los acontecimientos del mundo real y un público que está activamente interesado en el contenido de la base de datos. Estas pueden ser de cualquier tamaño y tener diversos grados de complejidad. La generación y el mantenimiento de las bases de datos pueden ser manuales o mecánicos.

Las bases de datos computarizadas se pueden crear y mantener con un grupo de programas de aplicación escritos específicamente para esa tarea, o bien mediante un Sistema de Gestión de Base de Datos (*SGBD*) de propósito general, el cual facilita el proceso de definir, construir y manipular bases de datos para diversas aplicaciones. Para definir una base de datos hay que especificar los tipos de datos, las estructuras y las restricciones de los datos que se almacenarán en ella. Construir una base de datos es el proceso de guardar los datos de los mismos en algún medio de almacenamiento controlado por el SGBD. En la manipulación de una base de datos intervienen funciones como consultar la base de datos para obtener datos específicos, actualizar la base de datos para reflejar cambios en el minimundo y generar informes a partir de los datos.

2.2.6.2 Componentes Principales de una Base de Datos

- **Datos:** Los datos representan la base de datos propiamente dicha.

- **Hardware:** Se refiere a los dispositivos de almacenamiento en donde reside la base de datos, así como a los dispositivos periféricos (unidad de control, canales de comunicación, etc.) necesarios para su uso.
- **Software:** Está constituido por un conjunto de programas que se conoce como Sistema Manejador de Base de Datos (*DBMS: Data Base Management System*). Este Sistema maneja todas las solicitudes formuladas por los usuarios a la base de datos.
- **Usuarios:** Existen tres clases de usuarios relacionados con una base de datos.
- **El programador de aplicaciones:** Es el que crea programas de aplicación que utilizan la base de datos**.**
- **El usuario final:** Es el que tiene acceso a la base de datos por medio de un lenguaje de consulta o de programas de aplicación.
- **El administrador de la Base de Datos (DBA: Data Base Administrator):** Es el que se encarga del control general del Sistema de base de datos.

2.2.6.3 Ventajas en el Uso de Base de Datos

- Globalización de la Información.
- Eliminación de Información Redundante, Duplicada o inconsistente.
- Almacenamiento Persistente de Objetos y Estructuras de Datos de Programas.
- Suministro de Múltiples Interfaces con los Usuarios.
- Respaldo y Recuperación.
- Cumplimiento de las Restricciones de Integridad.
- Restricción de los Accesos No Autorizados.

2.2.6.4 Clasificación

De acuerdo a su implementación y uso, las bases de datos se pueden clasificar en Bases de Datos Relacionales (las cuales se adaptan al modelo relacional) y Bases de Datos en tiempo real a continuación sus definiciones.

- **Bases de Datos Relacional**

Es aquella que se diseña y construye a partir del modelo de datos relacional, de acuerdo a sus especificaciones, características y restricciones.

El modelo relacional de los datos fue introducido por Codd (1970), se basa en una estructura de datos simple y uniforme, el cual tiene fundamentos teóricos sólidos.

Representa la Base de Datos como una colección de relaciones. En términos informales, cada relación semeja una tabla o, hasta cierto punto, un archivo simple. Si se visualiza una relación como una tabla de valores, cada fila de la tabla representa una colección de datos relacionados entre sí. Estos valores se pueden interpretar como hechos que describen una entidad o un vínculo entre entidades del mundo real. El nombre de la tabla y los nombres de las columnas ayudan a interpretar el significado de los valores que están en cada fila de la tabla. Todos los nombres en las columnas tienen el mismo tipo de datos. En la terminología del modelo relacional, una fila se denomina tupla, una cabecera de columna es un atributo y la tabla es una relación. El tipo de datos que describen los tipos de valores que pueden aparecer en cada columna se le llama dominio.

En este modelo, los datos son almacenados en tablas de datos bidimensionales, lo cual hace que la representación sea comprendida y visualizada con facilidad. El SGBD puede trabajar con dos o más tablas a la vez, relacionando la información a través de vínculos establecidos por una columna o campo en común. Los sistemas relacionales ofrecen distintos tipos de procesos de datos, como simplicidad y generalidad, facilidad de uso para el usuario final, periodos cortos de aprendizaje y las consultas de información se especifican de forma sencilla. Las tablas son un medio de representar la información de una forma más compacta y es posible acceder a la información contenida en dos o más tablas.

- **Base de Datos a Tiempo Real (RTDB)**

El modelo de una Base de Datos a Tiempo Real (RTDB) es un arreglo tridimensional al cual se denomina tabla, según sus características propias, como los atributos, valores y procedimientos.

Un sistema de Base de Datos en intervalos de tiempo real (RTDB), es un sistema que proporciona operaciones de la Base de Datos en intervalos de tiempo. La RTDB tiene dos características primarias que los distinguen de sistemas tradicionales de Base de Datos:

Manejan objetos de datos en tiempo real además de los objetos de datos fijos tradicionales. Los objetos de datos en tiempo real, llegan a ser poco confiables mientras que el mundo externo cambia, y por lo tanto necesitan ser actualizados frecuentemente para mantener la información actual.

Las transacciones en RTDB tienen intervalos de tiempo (plazo para almacenar los datos).

2.2.6.5 Formas Normales

Las relaciones que describe Codd al definir su modelo relacional, son transformadas en tablas al momento de trabajar con una base de datos. Las Formas Normales buscan optimizar estas estructuras eliminando básicamente la redundancia utilizando como medio principal las dependencias funcionales. Las Formas Normales son un pequeño número de reglas que de cumplirse hacen que las estructuras posean la menor cantidad de redundancia posible.

- **Primera Forma Normal (1NF):** No hay campos múltiples (todo los campos son atómicos). Todas las filas deben tener el mismo número de columnas.
- **Segunda Forma Normal (2NF):** Todo campo que no sea clave debe depender por completo de toda la clave.
- **Tercera Forma Normal (3NF):** No hay dependencias transitivas. Un campo debe depender de la clave y no de otro campo.
- **Forma Normal de Boyce-Codd (BCNF):** Todos los determinantes de la tabla son clave candidata.
- **Cuarta Forma Normal (4NF):** Una fila no debe contener dos o más campos multi-valorados (aquellos que pueden contener más de un valor simultáneamente) sobre una entidad.
- **Quinta Forma Normal (5NF):** Una tabla puede almacenar atributos dependientes a la clave sólo por unión.
- **Reglas de Codd.**

2.2.6.6 Modelo Entidad – Relación

Un diagrama o modelo entidad-relación (a veces denominado por su siglas, *E-REntity relationship*) es una herramienta para el modelado de datos de un sistema de información. Estos modelos expresan entidades relevantes para un sistema de información, sus inter-relaciones y propiedades.

El Modelo Entidad-Relación es un concepto de modelado para bases de datos, propuesto por Peter Chen en 1976, mediante el cual se pretende visualizar los objetos que pertenecen a la Base de Datos como entidades (esto es similar al modelo de Programación Orientada a Objetos) las cuales tienen unos atributos y se vinculan mediante relaciones.

Es una representación conceptual de la información. Mediante una serie de procedimientos se puede pasar del modelo E-R a otros, como por ejemplo el modelo relacional.

El modelado entidad-relación es una técnica para el modelado de datos utilizando diagramas entidad relación. No es la única técnica pero sí la más utilizada. Brevemente consiste en los siguientes pasos:

- Se parte de una descripción textual del problema o sistema de información a automatizar (los requisitos).
- Se hace una lista de los sustantivos y verbos que aparecen.
- Los sustantivos son posibles entidades o atributos.
- Los verbos son posibles relaciones.
- Analizando las frases se determina la cardinalidad de las relaciones y otros detalles.

- Se elabora el diagrama (o diagramas) entidad-relación.
- Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

2.2.7 Sistema Manejador de Base de Datos (DBMS)

2.2.7.1 Definición

El DBMS es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las bases de datos. Se compone de un lenguaje de definición de datos (*DDL: Data Definition Language*), de un lenguaje de manipulación de datos (*DML: Data Manipulation Language*) y de un lenguaje de consulta (*SQL: Structured Query Language*).

- **El Lenguaje de Definición de Datos (DDL):** Es utilizado para describir todas las estructuras de información y los programas que se usan para construir, actualizar e introducir la información que contiene una Bbase de datos.
- **El Lenguaje de Manipulación de Datos (DML):** Es utilizado para escribir programas que crean, actualizan y extraen información de las bases de datos.
- **El Lenguaje de Consulta (SQL):** Es empleado por el usuario para extraer información de las Bases de Datos. El Lenguaje de consulta permite al usuario hacer requisiciones de datos sin tener que escribir un programa, usando instrucciones como el SELECT, el PROJECT y el JOIN.

La secuencia conceptual de operaciones que ocurren para acceder cierta información que contiene una base de datos es la siguiente:

- El usuario solicita información contenida en la Base de Datos.
- El DBMS intercepta este requerimiento y lo interpreta.
- El DBMS realiza las operaciones necesarias para acceder y/o actualizar la información solicitada.
- El administrador de la Base de Datos (*DBA*).
- El DBA es la persona encargada de definir y controlar las Bases de Datos corporativas, además proporciona asesoría a los usuarios y ejecutivos que la requieran.

2.2.8 Sistema Gestor de Bases de Datos PostgreSQL (Mora, 2005)

2.2.8.1 Definición

PostgreSQL es un gestor de bases de datos orientadas a objetos 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 Sistemas Gestores de Bases de Datos (SGBD) comerciales.

2.2.8.2 Características del PostgreSQL

- La API de acceso al SGBD se encuentra disponible en C, C++, Java, PERL, PHP, Python y TCL, entre otros.

- Cuenta con un rico conjunto de tipos de datos, permitiendo además su extensión mediante tipos y operadores definidos y programados por el usuario.
- Su administración se basa en usuarios y privilegios.
- Sus opciones de conectividad abarcan TCP/IP, sockets Unix y sockets NT, además de soportar completamente ODBC.
- Los mensajes de error pueden estar en español y hacer ordenaciones correctas con palabras acentuadas o con la letra ‘ñ’.
- Es altamente confiable en cuanto a estabilidad se refiere.
- Puede extenderse con librerías externas para soportar encriptación, búsquedas por similitud fonética (soundex), etc.
- Control de concurrencia multi-versión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multi-usuario.
- Soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados, subconsultas y casi todos los tipos y operadores soportados en SQL92 y SQL99.
- Implementación de algunas extensiones de orientación a objetos. En PostgreSQL es posible definir un nuevo tipo de tabla a partir de otra previamente definida.

2.2.8.3 Entorno Gráfico PgAdmin III

Es el máximo exponente de cliente gráfico de PostgreSQL. En pgAdmin3 podemos ver y trabajar con casi todos los objetos de la base de datos, examinar sus propiedades y realizar tareas administrativas. También incorpora funcionalidades para realizar consultas, examinar su ejecución (como el comando explain) y trabajar con los datos.

2.2.9 Ingeniería De Software (Pressman, 1993)

2.2.9.1 Definición

Surge de la ingeniería de sistemas y de hardware. Abarca un conjunto de tres elementos claves: métodos, herramientas y procedimientos que facilitan al gestor controlar el proceso de desarrollo del software y suministrar a los que practiquen dicha ingeniería las bases para construir software de alta calidad de una forma productiva. A continuación se examinan brevemente cada uno de estos elementos:

- Los **métodos** de la ingeniería de Software indican cómo construir técnicamente el software. Los métodos abarcan un amplio espectro de tareas que incluyen: planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructura de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimiento.
- Las **herramientas** de la ingeniería del software suministran un soporte automático o semiautomático para los métodos. Hoy existen herramientas para soportar cada uno de los métodos mencionados anteriormente. Cuando se integran las herramientas de forma que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo de software, llamado ingeniería de software asistida por computadora (*del inglés, CASE)*.
- Los **procedimientos** de la ingeniería de software unen los métodos y las herramientas facilitando un desarrollo racional y oportuno del software de computadora como son: los procedimientos, la secuencia

en la que se aplican los métodos, las entregas (documentos, informes, formas, etc.) que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios, y las directrices que ayudan a los gestores de software a evaluar el progreso.

2.2.9.2 Paradigma del Ciclo de Vida Clásico para la Ingeniería de Software

Exige un enfoque sistemático y secuencial del desarrollo de software. Abarca las siguientes actividades:

- **Ingeniería y Análisis del Sistema:** abarca los requisitos globales a nivel del sistema con una pequeña cantidad de análisis y diseño a un nivel superior.
- **Análisis de los Requisitos del Software:** para comprender la naturaleza de los programas que hay que construir, el ingeniero de software (“analista”) debe interpretar el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridas. Los requisitos, tanto del sistema como del software, se documentan y se revisan con el software.
- **Diseño:** se enfoca sobre cuatro atributos distintos del programa (la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz). El proceso de diseño traduce los requisitos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes que comience la codificación.

- **Codificación:** el diseño debe traducirse en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente.
- **Prueba:** una vez generado el código, comienza la prueba de programa. La prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.
- **Mantenimiento:** no suele representar una actividad específica, sino que consiste normalmente en repetir o rehacer partes de las actividades de las fases anteriores para introducir cambios en una aplicación de software ya entregada al cliente y puesta en explotación.

2.2.10 Proceso Unificado (Jacobson, 2000)

Un proceso define quien está haciendo Qué, Cuándo y Cómo alcanzar un determinado objetivo. En la Ingeniería de Software el objetivo es construir un producto de software o mejorar uno existente. Un proceso efectivo proporciona normas para el desarrollo eficiente de software de calidad, captura y presenta las mejoras prácticas que el estado actual de la tecnología permite. En consecuencia, reduce el riesgo y hace el proyecto más predecible.

Dentro del proceso unificado estas preguntas son presentadas en cuatro elementos:

- Trabajadores: El quién

Los trabajadores definen los puestos que las personas pueden optar, es decir, un trabajador es un papel que un individuo puede desempeñar durante el desarrollo del software.

Cada trabajador tiene un conjunto de responsabilidades y lleva a cabo un conjunto de actividades en desarrollo del software. Un trabajador puede representar a un conjunto de personas que trabajan juntas.

- Actividades: El cómo

Una actividad es una parte del trabajo que un trabajador realiza, produciendo con ello un trabajo significativo, lo que representa una unidad con límites bien definidos para facilitar la asignación de tareas.

- Artefactos: El qué

Un artefacto es una pieza de información que es producida, modificada o utilizada en un proceso, en sí, es cualquier tipo de información que los trabajadores pueden usar.

- Flujos de Trabajo: El cuándo

  Un flujo de trabajo es un conjunto de actividades, y es el modo en que describe un proceso desarrollo. Ya se han mencionado los cinco flujos principales del proceso unificado.

2.2.10.1 Definición

Es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software. Sin embargo, el proceso unificado es

más que un simple proceso; es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños del proyecto.

El Proceso Unificado tiene dos dimensiones (*ver* **Figura 2.1**):

- Un eje horizontal que representa el tiempo y muestra los aspectos del ciclo de vida del proceso a lo largo de su desenvolvimiento.
- Un eje vertical que representa las disciplinas, las cuales agrupan actividades de una manera lógica de acuerdo a su naturaleza.

La primera dimensión representa el aspecto dinámico del proceso conforme se va desarrollando, se expresa en términos de fases, iteraciones e hitos. La segunda dimensión representa el aspecto estático del proceso: cómo es descrito en términos de componentes del proceso, disciplinas, actividades, flujos de trabajo, artefactos y roles.

El proceso unificado está basado en componentes lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.

2.2.10.2 Principios Básicos del Proceso Unificado

- **Proceso Unificado Dirigido por Casos de Uso:** Al estar dirigido por casos de usos significa que el proceso se centra en lo que debe hacer el sistema, comenzando por especificar sus requisitos. Los casos de uso representan los requisitos funcionales el sistema y el conjunto de todos estos casos de usos forman por lo que se conoce como

modelos de casos de usos, el cual contiene la descripción de todo el sistema. Los casos de usos además de ayudar a especificar los requisitos, sirven de guía durante todo el proceso, ya que se toman como base para el diseño, la implementación y las pruebas.

- **Un Proceso Centrado en la Arquitectura:** Al estar centrado en la arquitectura, el PU describe el sistema en varios puntos de vista. La arquitectura es la forma que tendrá el sistema y sirve como base para comprender como quedara terminado. Se utiliza como artefacto básico para conceptualizar, construir, gestionar y hacer evolucionar el sistema en desarrollo.
- **El Proceso es Iterativo e Incremental:** Un proceso iterativo es aquel que involucra la gestión de un flujo de versiones ejecutables. Un proceso incremental es aquel que implica la integración continua de la arquitectura del sistema para producir versiones, donde cada nuevo ejecutable incorpora mejoras incrementales sobre los otros. Es un proceso iterativo e incremental ya que divide el proyecto en pequeños mini-proyectos, donde a cada uno de estos se le conoce como iteraciones, las cuales una vez terminada significa un incremento en el proyecto.

2.2.10.3 Vida del Proceso Unificado

El Proceso unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema, los cuales constan de cuatro fases: Inicio, Elaboración, Construcción y Transición. Cada fase se subdivide a su vez en iteraciones que se repiten a lo largo de una serie de ciclos, donde el término de cada ciclo se obtiene una versión del software listo para ser entregado al cliente (*ver* **Figura 2.1**).

Una iteración puede pasar por los cinco flujos de trabajo fundamentales en el proceso unificado, los cuales son: requisitos, análisis, diseño, implementación y prueba.

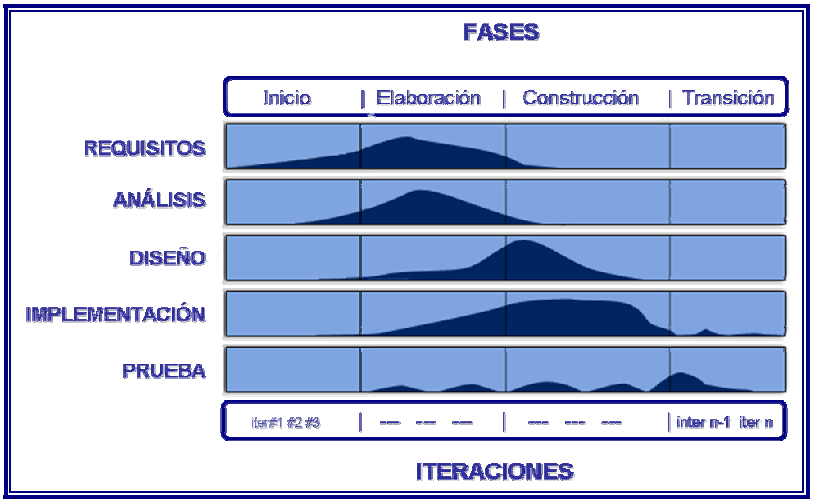

Figura 2.1. El Proceso Unificado de Desarrollo de Software.

Fuente: Jacobson, 2000.

Durante la fase de inicio, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocio para el producto. En esta fase se obtiene básicamente la descripción del problema a resolver mediante los casos de uso. Se describen las principales funciones que el sistema debe desempeñar, su arquitectura y un plan de tiempo y costo del proyecto.

Durante la fase de elaboración, se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La arquitectura se expresa en forma de vista de todos los modelos del sistema, los cuales juntos representan el sistema entero. Durante esta fase del desarrollo, se realizan los casos de usos más críticos que se identificaron en

la fase de comienzo, dando como resultado una línea base de la arquitectura. Al término de esta fase se planifican las actividades y los recursos necesarios para la realización del proyecto.

Durante la fase de construcción se crea el producto, se añaden los músculos (software terminado) al esqueleto (la arquitectura). En esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo. Al termino de esta fase, el sistema realizará las funciones especificadas por los casos de uso definidos en la fase de elaboración, pero es probable que en esta fase el sistema tenga errores los cuales se irán detectando y resolviendo en la fase de transición.

La fase de transición cubre el periodo durante el cual el producto se convierte la versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras sugeridas en una versión general dirigida a la totalidad de la comunidad de usuarios. La fase de transición conlleva a actividades como la fabricación, formación del cliente, el proporcionar una línea de ayuda y asistencia, y la corrección de los defectos que se encuentran tras la entrega.

2.2.11 Lenguaje Unificado de Modelado (UML) (Jacobson, 2000)

El proceso unificado utiliza el UML para expresar gráficamente todos los esquemas de un software.

El Lenguaje Unificado de Modelado (*ver* **Figura 2.2**) emergió en los '90, luego de la búsqueda de un lenguaje de modelado que unificara, que siguió a la “guerra de métodos” de los '70 y '80. A pesar que UML evolucionó

primeramente de varios métodos orientados al objeto de segunda generación UML no es simplemente un lenguaje para el modelado orientado a objetos de tercera generación. Su alcance extiende su uso más allá de sus predecesores. Y es la experiencia, experimentación y una gradual adopción del estándar lo que revelará su verdadero potencial y posibilitará a las organizaciones darse cuenta de sus beneficios.

Figura 2.2. Logo de UML

Fuente: Jacobson, 2000.

2.2.11.1 Definición

UML es un lenguaje de modelado unificado basado en una notación gráfica que permite: especificar, construir, visualizar y documentar los objetos de un sistema.

Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. UML capta la información sobre la estructura estática y el comportamiento dinámico de un sistema. Un sistema se modela como una colección de objetos discretos que interactúan para realizar un trabajo que finalmente beneficia a un usuario externo.

UML es un lenguaje de propósito general para el modelado orientado a objetos; es también un lenguaje de modelamiento visual que permite una abstracción del sistema y sus componentes.

Un modelo captura una vista de un sistema del mundo real. Es una abstracción de dicho sistema, considerando un cierto propósito. Así, el modelo describe completamente aquellos aspectos del sistema que son relevantes al propósito del modelo, y a un apropiado nivel de detalle.

Un proceso de desarrollo de software debe ofrecer un conjunto de modelos que permitan expresar el producto desde cada una de las perspectivas de interés. El código fuente del sistema es el modelo más detallado del sistema (y además es ejecutable). Sin embargo, se requieren otros modelos. Cada modelo es completo desde su punto de vista del sistema, sin embargo, existen relaciones de trazabilidad entre los diferentes modelos.

A través de un diagrama se puede realizar la representación gráfica de una colección de elementos de modelado, a menudo dibujada como un grafo conexo de arcos (relaciones) y vértices (otros elementos del modelo). Un diagrama no es un elemento semántico, un diagrama muestra representaciones de elementos semánticos del modelo, pero su significado no se ve afectado por la forma en que son representados.

La mayoría de los diagramas de UML y algunos símbolos complejos son grafos que contienen formas conectadas por rutas. La información está sobre todo en la topología, no en el tamaño o la colocación de los símbolos. Hay tres clases importantes de relaciones visuales: conexión, contención y adhesión. Así como también existen cuatro clases de construcciones gráficas

que se usan en la notación de UML: íconos, símbolos bidimensionales, rutas y cadenas.

2.2.11.2 Utilidad de UML

UML es un lenguaje para modelamiento de propósito general evolutivo, ampliamente aplicable, debe de ser soportado por herramientas e industrialmente estandarizado. Se aplica a una multitud de diferentes tipos de sistemas, dominios, y métodos o procesos.

- Como lenguaje de propósito general, se enfoca en el corazón de un conjunto de conceptos para la adquisición, compartición, y utilización de conocimientos emparejados con mecanismos de extensión.
- Como un lenguaje de modelamiento ampliamente aplicable, puede ser aplicado a diferentes tipos de sistemas (software y no-software), dominios (negocios versus software) y métodos o procesos.
- Como un lenguaje para modelamiento soportable por herramientas, las herramientas ya están disponibles para soportar la aplicación del lenguaje para especificar, visualizar, construir y documentar sistemas.
- Como un lenguaje para modelamiento industrialmente estandarizado, no es un lenguaje cerrado, propiedad de alguien, sino más bien, un lenguaje abierto y totalmente extensible reconocido por la industria.

UML posibilita captura, comunicación y nivelación de conocimiento estratégico, táctico y operacional para facilitar el incremento de valor, aumentando la calidad, reduciendo costos y reduciendo el tiempo de presentación al mercado; manejando riesgos y siendo proactivo para el posible aumento de complejidad o cambio.

2.2.11.3 Bloques Básicos de UML

Elementos del lenguaje: Estructurales, comportamiento, agrupación, anotación.

Relaciones entre elementos: Dependencia, asociación, generalización, realización.

Diagramas: Clases, objetos, casos de uso, secuencia, colaboración, estados, actividades, componentes, despliegue, capas.

a. Elementos de Lenguaje

- **Elementos estructurales**: Son los nombres de los modelos de UML. En su mayoría son las partes estáticas de un modelo, y representan conceptos o cosas materiales. Colectivamente, los elementos estructurales se denominan clasificadores.
  Entre los elementos estructurales tenemos:

  - **Actores**: Un actor es "algo" o "alguien" que puede interaccionar con el sistema que se está desarrollando (*ver* **Figura 2.3**).

Figura 2.3. Figura 2.3. **Representación de un Actor.**

Fuente: Rumbaugh, 2004.

- **Casos de uso:** Un caso de uso es una descripción de un conjunto de secuencias de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular (*ver* **Figura 2.4**).

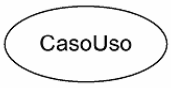

Figura 2.4. Estereotipo de un Caso de Uso

Fuente: Rumbaugh, 2004.

- **Clases**: Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica (*ver* **Figura 2.5**).

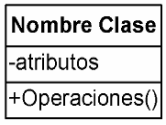

Figura 2.5. Representación de una Clase.

Fuente: Rumbaugh, 2004.

- **Objetos**: Un objeto es una instancia de alguna clase (*ver* **Figura 2.6**).

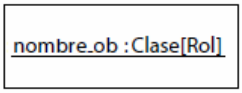

Figura 2.6. Representación de un Objeto.

Fuente: Rumbaugh, 2004.

- **Elementos de Comportamiento**: Son las partes dinámicas de los modelos UML. Éstos son los verbos de un modelo, y representan comportamiento en el tiempo y el espacio.

    - **Mensaje**: Los mensajes se usan para especificar una comunicación entre objetos (*ver* **Figura 2.7**). Se utilizan en los diagramas de secuencia.

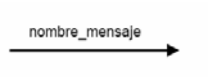

Figura 2.7. Representación de un Mensaje.
Fuente: Rumbaugh, 2004.

- **Elementos de Agrupación**: Son las partes organizativas de los modelos UML. Éstos son las cajas en las que puede descomponerse un modelo. Hay un tipo principal de elementos de agrupación: los paquetes.

    - **Paquete**: Sirve para organizar elementos en grupos. Un paquete es puramente conceptual (sólo existe en tiempo de desarrollo), como se puede observar en la **Figura 2.8**.

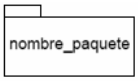

Figura 2.8. Representación de un Paquete.
Fuente: Rumbaugh, 2004.

b. Relaciones entre Elementos

- **Dependencia**: Es una relación semántica entre dos elementos (o dos conjuntos de elementos), en la cual un cambio en un elemento puede afectar a la semántica de otro elemento (*ver* **Figura 2.9**).

Figura 2.9. Representación de la Relación de Dependencia entre dos Clases.

Fuente: Rumbaugh, 2004.

Existen varios tipos de dependencia predefinidas que se indican mediante estereotipos, por ejemplo: «extend», e «include» para casos de uso.

- **Asociación:** Es una relación estructural entre dos elementos, que describe las conexiones entre ellos (suele ser bidireccional). Es la única relación permitida entre los actores y los casos de uso (refleja la comunicación existente entre un actor y un caso de uso) (*ver* **Figura 2.10**).

Figura 2.10. Representación de la Relación de Asociación entre Dos Clases.

Fuente: Rumbaugh, 2004.

- **Agregación**: Es una relación estructural entre un todo y sus partes. Se denota por una línea terminada en un "diamante" en el extremo de la clase que representa el todo (*ver* **Figura 2.11**).

Figura 2.11. Representación de una Relación de Agregación entre Dos Clases.

Fuente: Rumbaugh, 2004.

- **Generalización:** Es una relación taxonómica entre un elemento más general (el padre) y un elemento más específico (el hijo). Se usa tanto en diagramas de clases como en diagramas de casos de uso (*ver* **Figura 2.12**).

Figura 2.12. Representación de una Relación de Generalización entre Dos Clases.
Fuente: Rumbaugh, 2004.

c. Diagramas (Maldonado, 2008)

Los elementos de UML se muestran mediante diagramas que presentan múltiples vistas del sistema, ese conjunto de vistas son conocidos como modelos. UML presenta varios diagramas donde cada uno representa un aspecto del sistema, los cuales están conformados por Diagramas de estructura, Diagramas de comportamiento y Diagramas de interacción.

A partir de la versión 2.0 de los diagramas de UML, existen trece tipos diferentes de diagramas, de hecho están agrupados por categoría para ser más fácil entenderlos y aplicarlos.

Entre las categorías y sus Diagramas se tienen las siguientes:

Los Diagramas de Estructura, muestran los elementos que existen en el modelo:

- Diagrama de Clases.

- Diagrama de Componentes.
- Diagrama de Objetos.
- Diagrama de Estructura Compuesta.
- Diagrama de Despliegue.
- Diagrama de Paquetes.

Los Diagramas de Comportamiento, muestra lo que puede suceden dentro del modelo:

- Diagrama de Actividades.
- Diagrama de Casos de Uso.
- Diagrama de Estados.

Los Diagrama de interacción, también conocidos como subtipos de diagramas de comportamiento y tiene como fin mostrar los flujos de control.

- Diagrama de Secuencia.
- Diagrama de Colaboración.
- Diagrama de Tiempos.
- Diagrama de Vistas de interacción.

- **Diagrama de Casos de Uso**

El diagrama de casos de uso se emplea para capturar información de cómo un sistema o negocio trabaja, o de cómo se desea que trabaje (*ver* **Figura 2.13**). Los casos de uso no son artefactos orientados a objetos, es una técnica para captura de requisitos. Captura la funcionalidad del sistema vista por los usuarios.

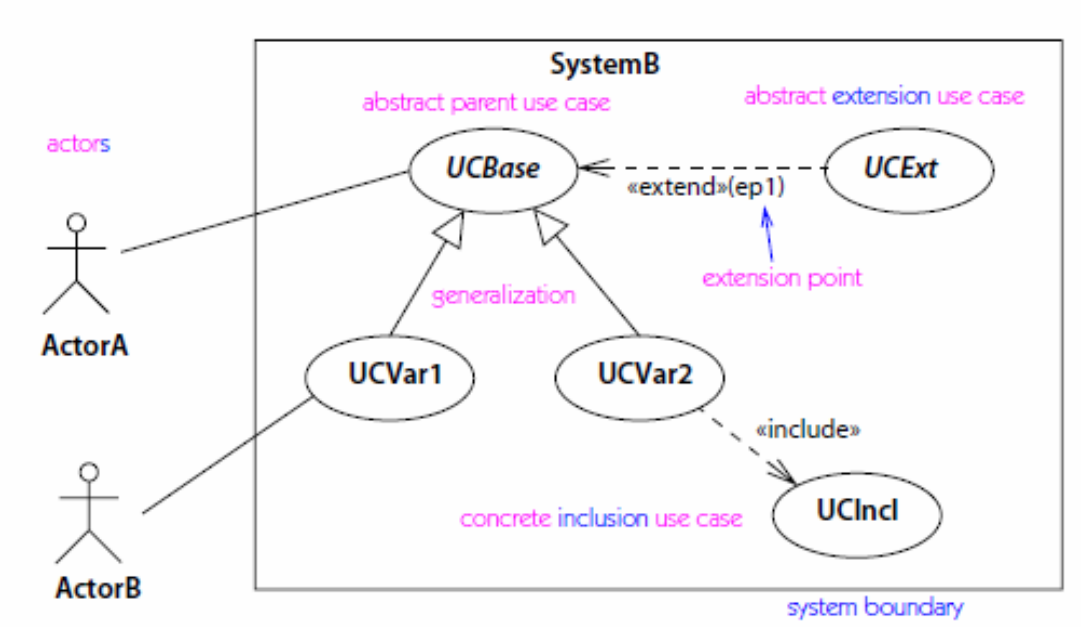

Figura 2.13. Diagrama de Casos de Uso.

Fuente: Rumbaugh, 2004.

- **Diagrama de Secuencia**

El diagrama de secuencia muestra las interacciones entre los objetos organizadas en una secuencia temporal. En particular muestra los objetos participantes en la interacción y la secuencia de mensajes intercambiados (*ver* **Figura 2.14**). Representa una interacción, un conjunto de comunicaciones entre objetos organizadas visualmente por orden temporal. A diferencia de los diagramas de colaboración, los diagramas de secuencia incluyen secuencias temporales pero no incluyen las relaciones entre objetos. Pueden existir de forma de descriptor (describiendo todos los posibles escenarios) y en forma de instancia (describiendo un escenario real).

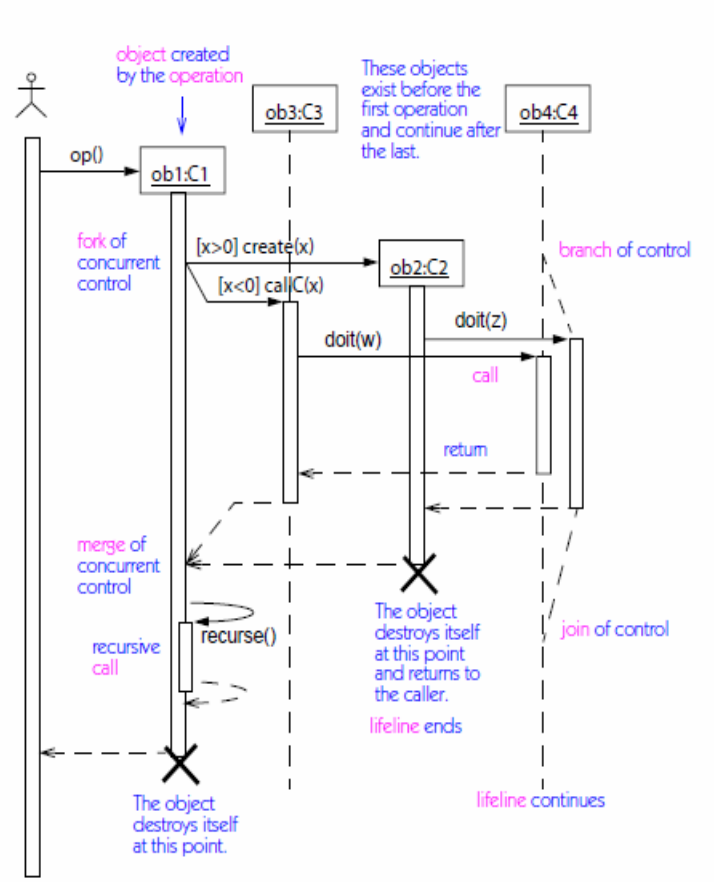

Figura 2.14. Diagrama de Secuencia.

Fuente: Rumbaugh, 2004.

- **Diagrama de Colaboración**

Un diagrama de colaboración muestra la implementación de la operación (*ver* **Figura 2.15**). La colaboración muestra los parámetros y las variables locales de la operación, así como operaciones más permanentes; modela los objetos y los enlaces significativos dentro de una interacción. Un rol describe un objeto, y un rol en la asociación describe un enlace dentro de una colaboración. Un diagrama de colaboración muestra los roles en la interacción en una disposición geométrica.

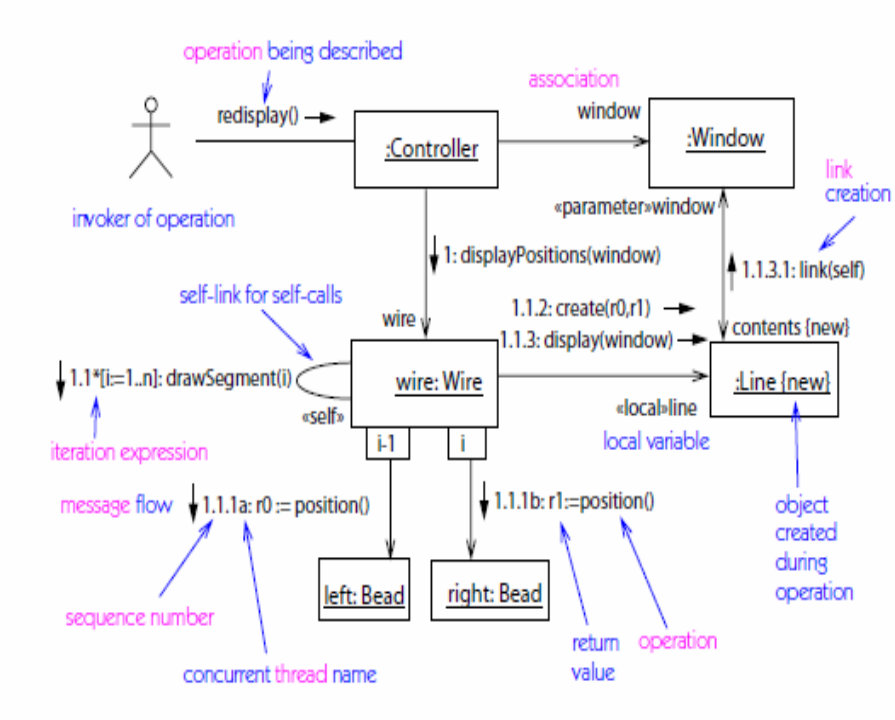

Figura 2.15. Diagrama de Colaboración.

Fuente: Rumbaugh, 2004.

- **Diagrama de Clases**

Es un diagrama que muestra un conjunto de clases y sus colaboraciones y relaciones. Estos diagramas sirven para visualizar las relaciones existentes entre las distintas clases y las formas en que colaboran unas con otras (ver **Figura 2.16**). Las relaciones entre las distintas clases son las relaciones comunes existentes en UML aunque con matices: Una asociación se traduce como que desde los objetos de una clase se puede acceder a los objetos de otra. Una dependencia se puede visualizar como que la clase utilizada es un parámetro de un método de la clase que la utiliza. Una generalización se traduce como una herencia entre clases. Además de las clases en el análisis, se encuentran tres estereotipos fundamentales a saber: de interfaz, de control y de entidad.

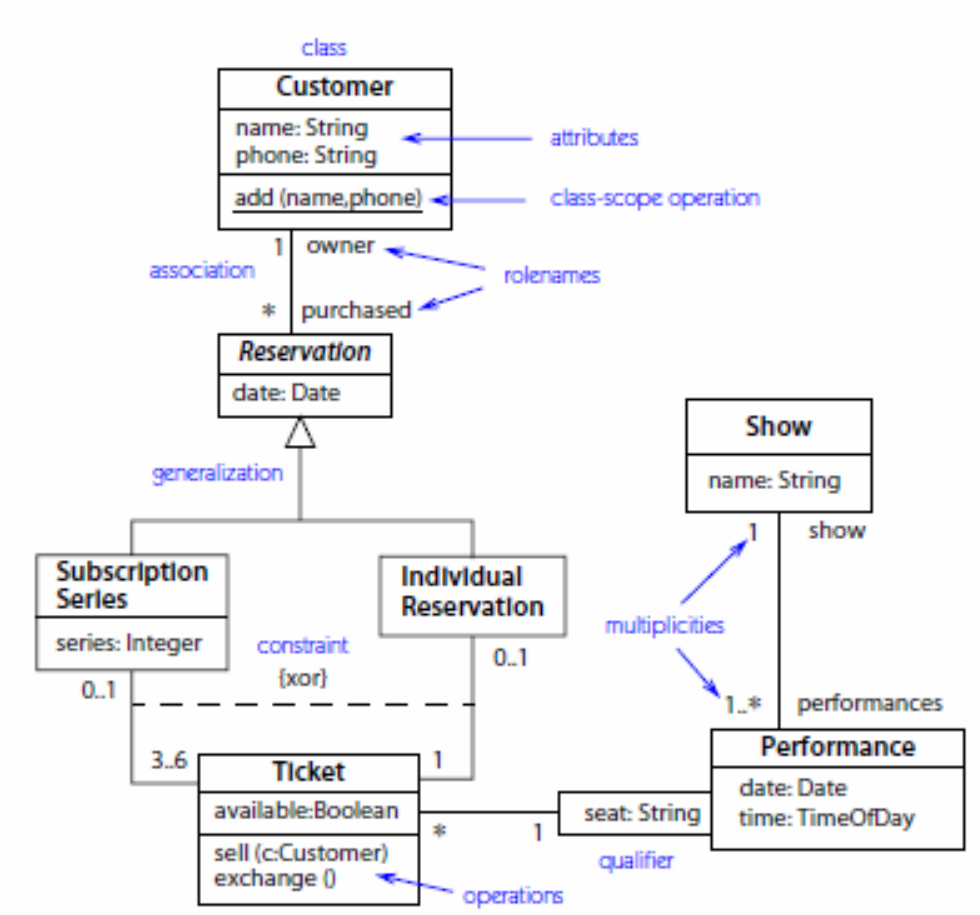

Figura 2.16. Diagrama de Clases.

Fuente: Rumbaugh, 2004.

- **Diagrama de Despliegue**

Los diagramas de despliegue muestran la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. Existen dos niveles, el nivel de Descriptor (*ver* **Figura 2.17**) y el nivel de Instancia (*ver* **Figura 2.18**). La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

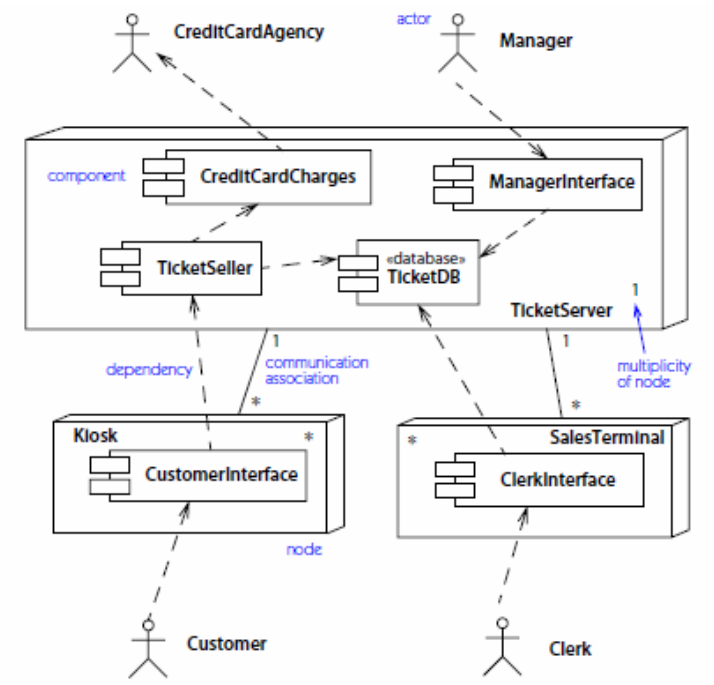

Figura 2.17. Diagrama de Despliegue (Nivel de Descriptor).

Fuente: Rumbaugh, 2004.

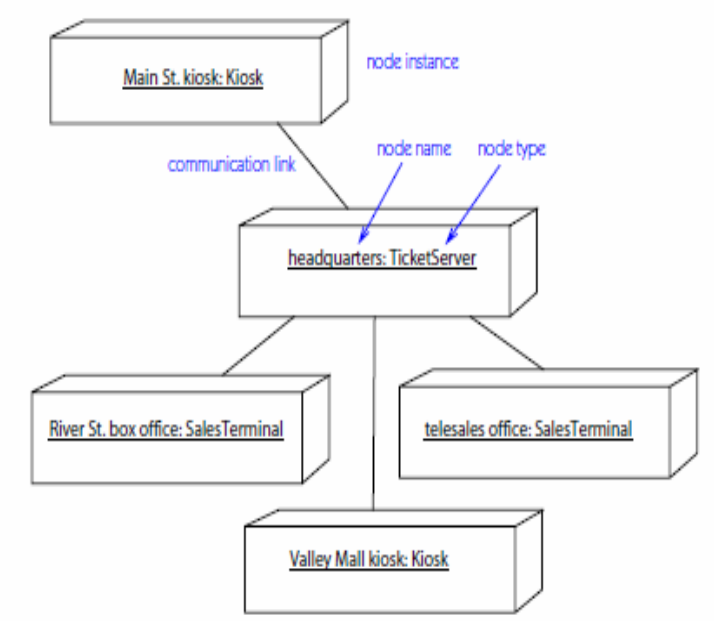

Figura 2.18. Diagrama de Despliegue (Nivel de Instancia).

Fuente: Rumbaugh, 2004.

- **Diagrama de Componentes**

Es un diagrama que muestra un conjunto de componentes y sus relaciones (*ver* **Figura 2.19**). Los diagramas de componentes muestran los componentes de un sistema desde un punto de vista estático.

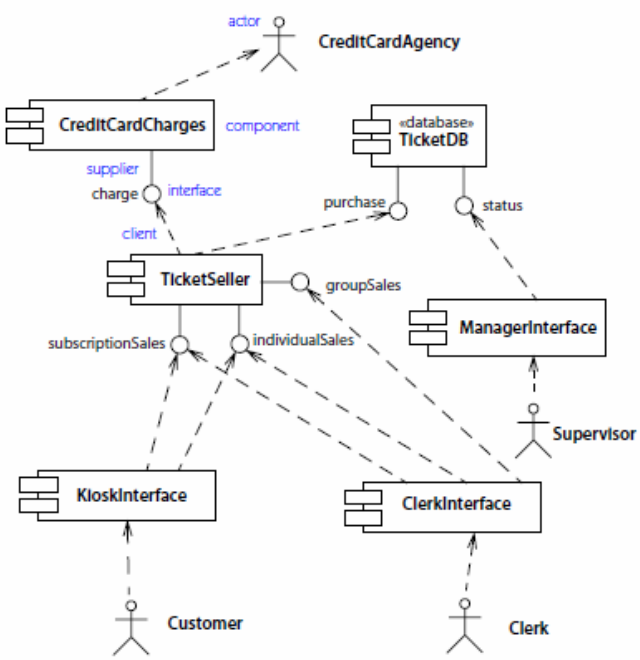

Figura 2.19. Diagrama de Componentes.

Fuente: Rumbaugh, 2004.

- **Diagrama de Objetos**

Un diagrama de objetos representa un conjunto de objetos y sus relaciones (*ver* **Figura 2.20**). Se utilizan para describir estructuras de datos, instantáneas estáticas de las instancias de los elementos existentes en los diagramas de clases. Los diagramas de objetos abarcan la vista de diseño estática o la vista de procesos estática de un sistema al igual que los diagramas de clases, pero desde la perspectiva de casos reales o prototípicos.

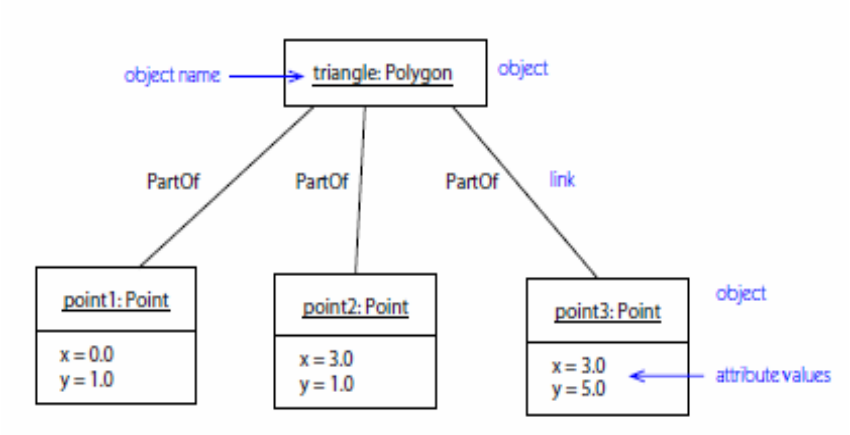

Figura 2.20. Diagrama de Objetos.

Fuente: Rumbaugh, 2004.

- **Diagrama de Estados**

Un diagrama de estados representa una máquina de estados, constituida por estados, transiciones, eventos y actividades (*ver* **Figura 2.21**). Los diagramas de estados se utilizan para describir la vista dinámica de un sistema. Son especialmente importantes para modelar el comportamiento de una interfaz, una clase o una colaboración. Los diagramas de estados resaltan el comportamiento dirigidos por eventos de un objeto, lo que es especialmente útil al modelar sistemas reactivos.

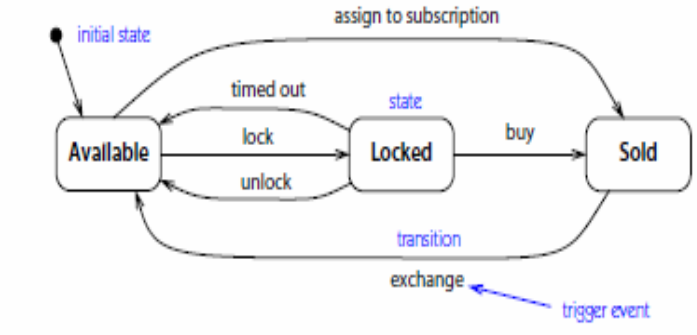

Figura 2.21. Diagrama de Estados.

Fuente: Rumbaugh, 2004.

- **Diagrama de Actividades**

Un diagrama de actividades muestra el flujo paso a paso en una computación (*ver* **Figura 2.22**). Una actividad muestra un conjunto de acciones, el flujo secuencial o ramificado de acción en acción, y los valores que son producidos o consumidos por las acciones. Los diagramas de actividades se utilizan para ilustrar la vista dinámica de un sistema.

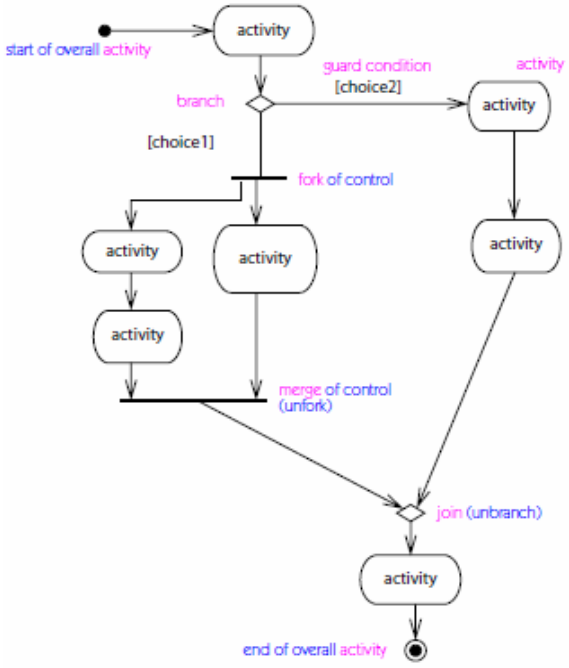

Figura 2.22. Diagrama de Actividades.
Fuente: Rumbaugh, 2004.

2.2.12 Interfaz de Usuario **(Myers, 1996)**

2.2.12.1 Definición

Una interfaz es un conjunto de comandos o menús a través de los cuales el usuario se comunica con el programa. Esta es una de las partes

más importantes de cualquier programa ya que determina que tan fácilmente es posible que el programa haga lo que el usuario quiere hacer. Un programa muy poderoso con una interfaz pobremente elaborada tiene poco valor para un usuario no experto.

2.2.12.2 Principales Funciones de las Interfaces de Usuario

Sus principales funciones son:

- Puesta en marcha y apagado
- Control de las funciones manipulables del equipo
- Manipulación de archivos y directorios
- Herramientas de desarrollo de aplicaciones
- Comunicación con otros sistemas
- Información de estado
- Configuración de la propia interfaz y entorno
- Intercambio de datos entre aplicaciones
- Control de acceso
- Sistema de ayuda interactivo.

2.2.12.3 Tipos de Interfaces de Usuario

a. Según la forma de interactuar del usuario, atendiendo a como el usuario puede interactuar con una interfaz, nos encontramos con varios tipos de interfaces de Usuario:

- Interfaces alfanuméricas (intérpretes de mandatos) que solo presentan texto.

- Interfaces gráficas de usuario (*GUI, Graphics User Interfaces*), las que permiten comunicarse con el ordenador de una forma muy rápida e intuitiva representando gráficamente los elementos de control y medida.
- Interfaces táctiles, que representan gráficamente un "panel de control" en una pantalla sensible que permite interaccionar con el dedo de forma similar a si se accionara un control físico.

b. Según su Construcción

Pueden ser de hardware o de software:

- Interfaces hardware: Se trata de un conjunto de controles o dispositivos que permiten la interacción hombre-máquina, de modo que permiten introducir o leer datos del equipo, mediante pulsadores, reguladores e instrumentos.
- Interfaces software: Son programas o parte de ellos, que permiten expresar nuestros deseos al ordenador o visualizar su respuesta.

CAPÍTULO III
FASE DE INICIO

INTRODUCCIÓN

Para llevar a cabo la fase de inicio se abarcarán los flujos de trabajo, modelación de negocio, requerimientos y análisis (*ver* **Figura 3.1**).

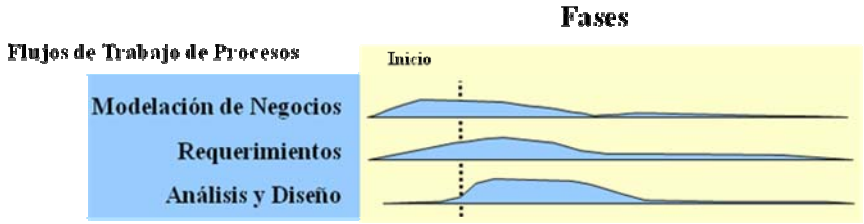

Figura 3.1. Flujos de trabajo de la fase de inicio.
Fuente: Rumbaugh, 2004.

En esta fase se definirá y acordará el alcance del proyecto con las personas involucradas y responsables del sistema propuesto, identificando los requisitos, desarrollando el modelo de dominio y el modelo de casos de uso para los principales procesos, así como también se identificarán los riesgos asociados al proyecto. A partir del modelo de casos de uso y la lista de riesgos, se determinará qué casos de uso deben ejecutarse primero para atacar los riesgos críticos. Con base en la información previa se realizará el proceso de planificación general y un plan de trabajo detallado para la siguiente fase, así como el plan para la siguiente iteración de ser necesario.

3.1 FLUJO DE TRABAJO DE REQUISITOS

El flujo de trabajo de requisitos es uno de los más importantes, ya que en él se establece qué tiene que hacer exactamente el sistema que se va a

construir, éste tiene como propósito guiar el desarrollo del proyecto hacia el sistema correcto hasta obtener una arquitectura sólida. Se basa en la comprensión del contexto del sistema usando el diagrama del modelo de dominio, estudio del contexto del sistema, requerimientos funcionales, lista de riesgos, modelos de casos de uso, identificación de actores, descripción de casos de uso y requerimientos adicionales.

Los requisitos están comprendidos en dos grupos. Los requisitos funcionales que representa la funcionalidad primaria del sistema, se modelan mediante diagramas de casos de uso. Los requisitos no funcionales que representan aquellos atributos que debe exhibir el sistema, pero que no son una funcionalidad específica.

Para el establecimiento de los requerimientos del sistema, se utilizará el modelado de diagramas de casos de usos, el cual es una herramienta del lenguaje unificado de modelado (*UML*), que permite representar el comportamiento general del sistema desde una figura externa, donde se especifica qué debería hacer el sistema. Este modelo también sirve para definir cuáles son los límites del sistema, quienes van a ser los usuarios del sistema, cuáles son sus funciones o roles, así como son los casos de uso que interactuarán con los actores identificados.

3.1.1 Modelo de Dominio

La captura de los requisitos funcionales demanda un firme conocimiento del contexto en el que se ubica el sistema. Para tal fin, es necesario comprender los procesos y las actividades de la organización relacionadas con el sistema actual, de tal forma que esto permita identificar los casos de uso y las entidades relevantes que debe soportar el sistema.

El modelo del dominio es un modelo conceptual empleado para comprender de una forma intuitiva el contexto del sistema y representar conceptos concernientes al dominio de una forma clara. Estos conceptos surgen de una investigación del dominio del problema.

UML, cuenta con una notación en diagramas de estructura estática que explican gráficamente los modelos conceptuales. Estos representan cosas del mundo real, no componentes del software.

El modelo de dominio del sistema propuesto el cual refleja las entidades y sus relaciones dentro del contexto del mismo y que interactúa dentro del proceso a automatizar, el sistema tendrá el nombre de “SCAAMP” (*Sistema para el Control Administrativo de Afiliados y Monitoreo de Paquetes*), cabe destacar que este no representa ninguna funcionalidad, sino las entidades más importantes de éste, y las asociaciones entre cada una.

En el diagrama del Modelo de Dominio (*ver* **figura 3.2**), se observa el proceso para efectuar un envío y su seguimiento, desde el momento en el cual el cliente lo realiza y es tramitado por el usuario del sistema hasta que llega a su destino. Cada uno de los envíos genera una guía, seguidamente estas son asignadas a una unidad de transporte que se encargará de hacer llegar el paquete a su destino. Todas las guías asignadas a una unidad son vinculadas en una remesa (*Control de Encomiendas*), y posteriormente el administrador procesa la cuenta asociada a cada afiliado y sus unidades de transporte.

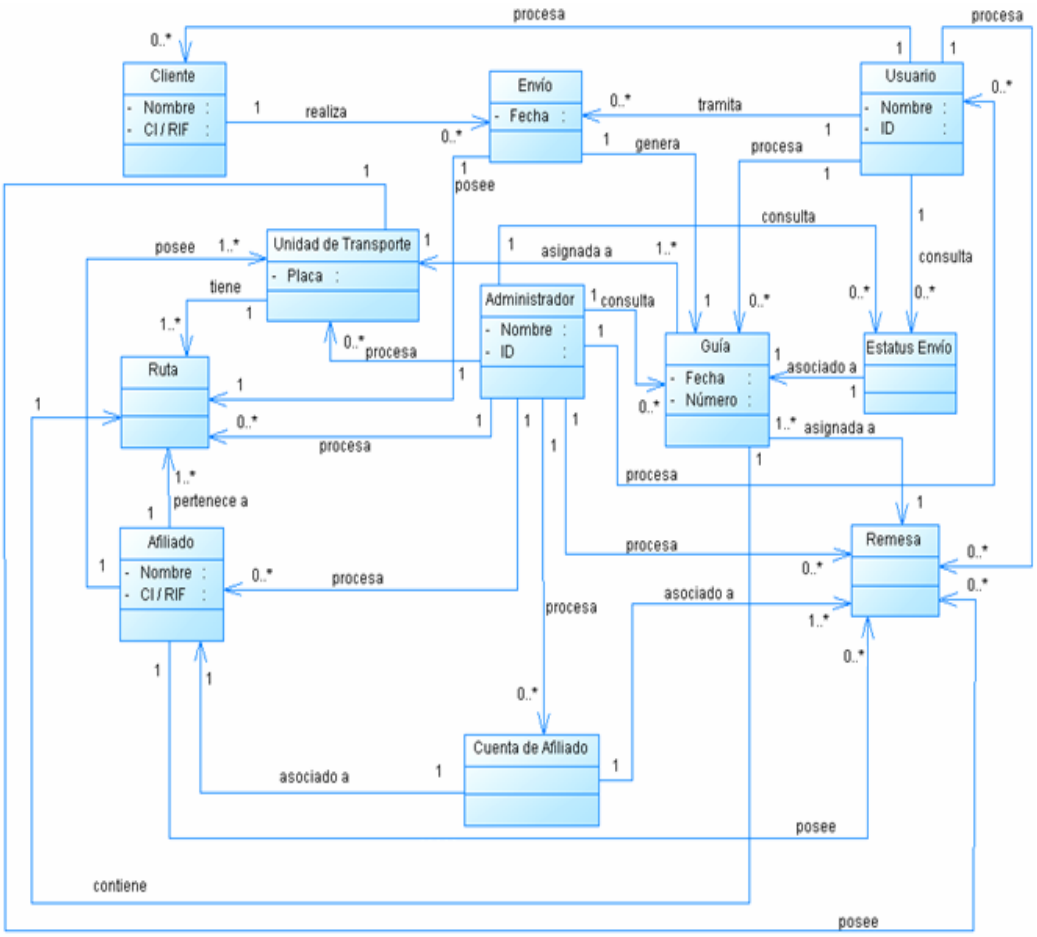

Figura 3.2. Modelo de Dominio del Sistema

Fuente: Elaboración Propia

3.1.1.1 Glosario de Términos Modelo de Dominio

Es necesario contar con un lenguaje común para facilitar el intercambio de ideas. El glosario o diccionario (*ver* **Tabla 3.1**) modelo incluye y define todos los términos que son de vital importancia para la comprensión del análisis que se realizará en las fases de desarrollo del presente estudio, los cuales se mencionan a continuación:

Tabla 3.1. Glosario de Términos

TÉRMINO	DESCRIPCIÓN
Administrador	Representa a la persona encargada del procesamiento de las unidades de transporte, remesas, rutas, usuarios, afiliados y sus cuentas.
Afiliado	Simboliza la persona que posee unidades de transporte.
Cliente	Es la persona que realiza el envío.
Cuenta de Afiliado	Se refiere a la administración del afiliado por unidades de transporte.
Envío	Significa la solicitud realizada por un cliente para el traslado de una encomienda.
Estatus de Envío	Estado en el que se encuentra en un momento dado el envío
Guía	Contempla toda la información concerniente al envío.
Remesa	Relación de guías transportadas por unidad.
Ruta	Simboliza el área a la cual se realiza el envío.
Usuario	Personifica a quien se encarga del procesamiento de la información de los clientes, remesas, envíos y sus estatus.
Unidad de Transporte	Donde es trasladada la encomienda desde el lugar de origen hasta su destino.

Fuente: Elaboración Propia

3.1.2 Contexto del Sistema

Para determinar el contexto se debe realizar un acercamiento con el entorno del sistema que se desea desarrollar. Partiendo del conocimiento de las Empresas de Encomiendas, para luego analizar las actividades de control del administrativo de afiliados y monitoreo de paquetes, a través de la observación directa con la finalidad de comprender los procesos de mayor relevancia dentro del área de interés, y ofrecer una visión representativa del contexto del sistema.

Cuando se habla de contexto se hace referencia a las circunstancias y condiciones que rodean un evento determinado, haciendo uso de este concepto se determinó el contexto del presente sistema partiendo del conocimiento de las Empresas de Encomiendas, para luego analizar las actividades de control del administrativo de afiliados y monitoreo de paquetes.

Este análisis se logró realizando visitas y encuestas al personal de la empresa, esto con la finalidad de conocer directamente todos los procesos que se llevan a cabo e identificar las debilidades que poseen, para realizar mejoras mediante su automatización. Luego de conocer el contexto en el cual se ubica cada proceso, se procede a establecer o indicar los diversos requisitos que servirán de base como lineamiento a seguir durante el desarrollo del sistema propuesto.

El sistema SCAAMP será el encargado de automatizar los procesos de recopilación y análisis de la información requerida para aplicar control administrativo a los afiliados de la empresa, así como de tramitar el envío y monitoreo de los paquetes. Con lo cual se pretende reducir el tiempo de respuesta.

3.1.3 Requerimientos Esenciales del Sistema

Los requisitos de información del sistema, es una de las actividades claves para el desarrollo del mismo, y por lo tanto, reflejan las necesidades del sistema para procesar la información. Estos requisitos son esenciales e importantes ya que determinan específicamente las necesidades.

El sistema propuesto (*SCAAMP*), que permitirá la automatización de las actividades relacionadas al control administrativo de afiliados y monitoreo de paquetes utilizados en una empresa de encomiendas privada, debe cumplir con los siguientes requerimientos funcionales y no funcionales.

3.1.3.1 Requisitos Funcionales

Los requisitos funcionales determinan las necesidades de información y automatización de los procesos de negocios, que tienen los usuarios y que el sistema debe cumplir luego de su desarrollo. A continuación se muestra la lista de los requisitos que deben implementarse de manera funcional en el posterior desarrollo del sistema:

- Registro, consulta y actualización de clientes.
- Registro, consulta y actualización de usuarios.
- El software debe permitir ingresar datos para las consultas que realizarán los usuarios.
- El software debe ser diseñado con restricciones de entrada de usuarios, esto con la finalidad de mantener la seguridad de la información almacenada en la base de datos.
- Registro, consulta y actualización de afiliados.
- El software debe permitir el procesar la cuenta de cada afiliado.
- El sistema debe permitir procesar las remesas de cada afiliado asociadas a las unidades de transporte.
- El sistema debe contar con mecanismos de control que validen la información ingresada por el usuario, evitando asi que se introduzcan datos erroneos o no válidos.

- El software diseñado debe permitir el mantenimiento sin ninguna complicación.

3.1.3.2 Requisitos No Funcionales

Los requisitos no funcionales representan aquellos aspectos del sistema que no cumplen una función específica, pero que ayudan a la interacción entre los actores y el sistema. Son las restricciones de la aplicación, los atributos de calidad, los limites de memoria, requerimientos de seguridad, restricciones de software, restricciones de hardware, etc. En este caso se encontraron los siguientes:

- La arquitectura de software a usar debe ser la siguiente: Sistema operativo Windows o Linux, donde debe estar el Java Virtual Machine, PostgreSQL como servidor de base de datos, NetBeans IDE o Eclipse como herramientas de desarrollo, el lenguaje de programación debe ser Java.
- El acceso al sistema sólo puede realizarse a través de los equipos de la empresa, ya que es una aplicación de escritorio.
- La información del password o clave del usuario debe almacenarse encriptado en la base de datos para brindar seguridad e integridad de los datos, ya que la organización maneja volúmenes de información de considerable importancia por lo que se hace necesario software de seguridad para ello.
- La interfaz de usuario debe ser agradable e intuitiva, que sea fácil de operar, ya que existen usuarios de nivel intermedio en computación.

3.1.4 Identificación de Riesgos

La identificación de riesgo se basa en el conocimiento y recopilación de aquellos factores que pueden influir directamente en el proceso de desarrollo del proyecto y que logren poner en peligro su progreso y desenvolvimiento, desde el inicio hasta su culminación. Por estas razones se hace obvio que para el Proceso Unificado, los riesgos dirigen la viabilidad del sistema.

Una vez identificados pueden ser tratados de diversas maneras. Se cuenta fundamentalmente con cuatro elecciones: evitarlos, limitarlos, eliminarlos o controlarlos. Algunos riesgos pueden evitarse mediante una replanificación del proyecto o un cambio en los requisitos; otros deberían restringirse de modo que sólo afecten una parte del proyecto; otros atenuarse ejercitándolos y observándolos; aunque existen riesgos que no pueden atenuarse, solamente pueden controlarse y observar si aparecen.

Por esto la reducción de riesgos es un punto importante en las iteraciones de las primeras fases, de esta forma en las fases posteriores gran parte de los riesgos se logran reducir a lo usual y su manejo significará una práctica habitual. A continuación se presenta la lista de riesgos críticos donde se hace una breve descripción de los riesgos (*ver* **tabla 3.2**), se indica que parte del sistema es afectada, se señala al responsable de tratarlo y por último las acciones a llevar a cabo para mitigarlo en caso de que se presente.

Tabla 3.2. Lista de riesgos críticos.

DESCRIPCIÓN	IMPACTO	RESPONSABILIDAD	CONTINGENCIA
Escogencia de herramientas inadecuadas para el desarrollo del Sistema	Software	Desarrollador	Estudiar las capacidades de las herramientas seleccionadas y sus posibles reemplazos
Falta de dominio del contexto del sistema	Software	Desarrolladores	Evaluar junto con los usuarios el contexto
Incapacidad de uso del sistema por parte de usuarios finales	Empresa	Desarrollador	Verificar la tecnología utilizada en la empresa y correcto diseño de interfaces y manual de usuario
No lograr una interfaz adecuada para las necesidades del sistema	Software	Desarrolladores	Profundizar el análisis del flujo de actividades con apoyo del futuro usuario
Fallas en el acceso a la base de datos	Consultas	Desarrollador	Establecer un correcto diseño de la base de datos y de las consultas que se realizan
Falta de acceso a información fundamental durante el levantamiento de información.	Software, Empresa	Empresa	Exponer a la empresa las razones por las cuales esta información es vital para el desarrollo del proyecto
Datos errados en la información suministrada	Consultas, Empresa	Desarrollador	Verificar la información documentada por parte de la empresa

Fuente: Elaboración Propia

3.1.5 Modelo de Casos de Uso

El modelo de casos de uso tiene como finalidad la captura de todos o parte de los requisitos funcionales del sistema, ya que describe la funcionalidad propuesta del nuevo sistema. El recurso principal del cual se vale este modelo es el diagrama de casos de uso, que muestra las distintas operaciones que se esperan de una aplicación o sistema, y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

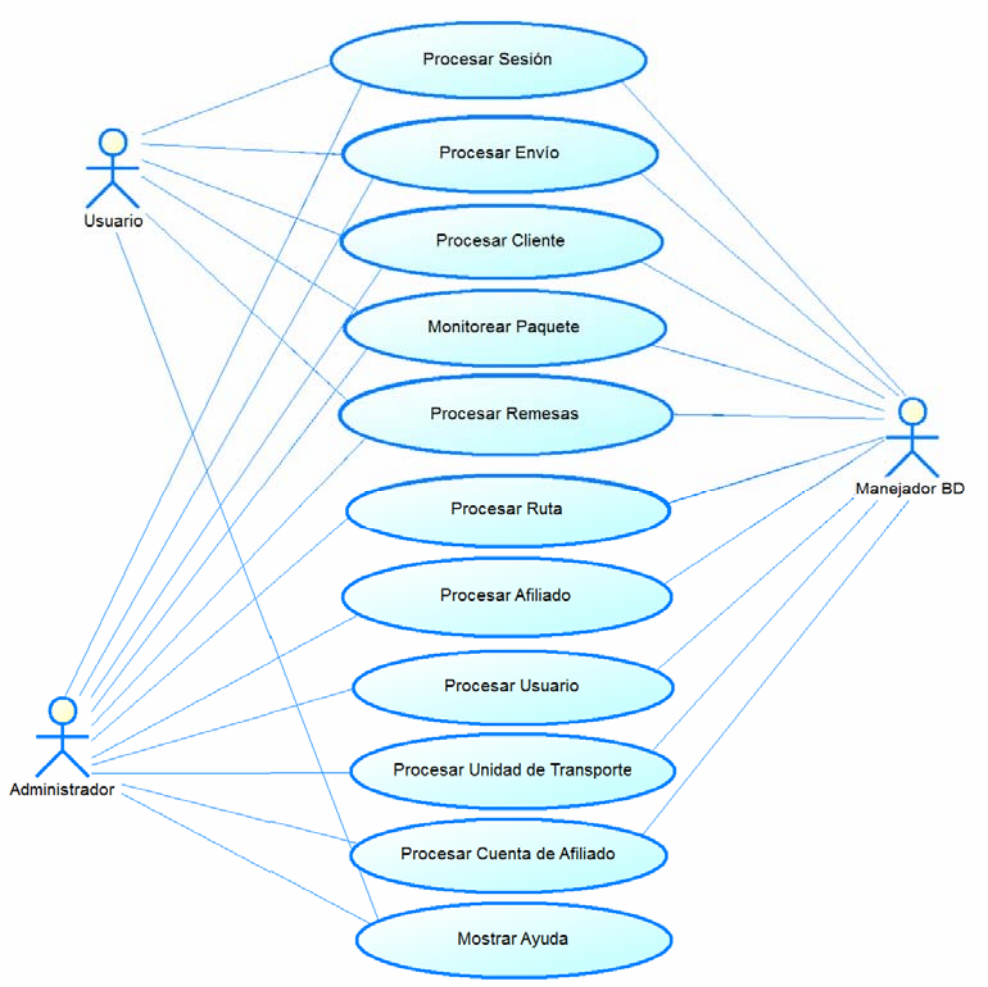

Figura 3.3. Diagrama de Casos de Uso del Sistema

Fuente: Elaboración Propia

En la **figura 3.3** se muestra el diagrama de casos de uso general realizado para el sistema que se está proponiendo, se puede apreciar que serán dos actores los que van a interactuar con el sistema.

3.1.5.1 Identificación de los Casos de Uso

Al hacer referencia a este punto se habla de la funcionalidad del sistema, requisitos que el sistema debe manejar y presentar a los usuarios en forma de funciones y de manera automatizada. Cada usuario va a interactuar con el sistema mediante uno o varios casos de uso, determinados por el tipo de acceso que tenga cada uno.

Para desarrollar los casos de uso correctamente es necesario entender el contexto del sistema y los requisitos esenciales, para así poder aproximarse realmente a las necesidades del usuario, y de esta manera tener un acercamiento al entorno que el proceso implica.

En la **Tabla 3.3** se definen los casos de uso existentes en el sistema en desarrollo.

Tabla 3.3. Definición de los Casos de Uso (1/2).

CASO DE USO	DESCRIPCIÓN
Procesar Cliente	Permite ingresar, modificar, consultar y eliminar toda la información referente a los clientes de la empresa.
Procesar Envío	Permite registrar, consultar y cancelar toda la información referente a la realización de un envío por parte del cliente.
Monitorear Paquete	Permite consultar y modificar el estatus de un envío.
Procesar Remesa	Permite procesar la relación de guías transportadas por unidad.

Tabla 3.3. Definición de los Casos de Uso (2/2).

CASO DE USO	DESCRIPCIÓN
Procesar Ruta	Permite ingresar y eliminar una ruta del sistema.
Procesar Afilado	Permite ingresar, modificar, consultar y eliminar toda la información referente a los socios de la empresa.
Procesar Usuario	Permite registrar y manipular la información concerniente a los usuarios que tienen acceso al sistema.
Procesar Cuenta de Afiliado	Permite realizar todo el control administrativo de cada socio de la empresa.

Fuente: Elaboración Propia

3.1.5.2 Identificación de Actores

Los actores son personas, sistemas o hardware externo que interactúa con el sistema en cuestión, es decir, representan a terceros fuera del sistema que colaboran con éste. Los actores pueden utilizar funciones propias del sistema, y de igual forma pueden proveer otras distintas al sistema, obteniendo o ingresando información en el mismo.

Para el Sistema que efectuará el Control Administrativo de Afiliados y Monitoreo de Paquetes (*SCAAMP*) se identificaron dos actores fundamentales como lo son: Usuario y Administrador (*ver* **Tabla 3.4**).

Tabla 3.4. Descripción de Actores (1/2).

ACTOR	DESCRIPCIÓN
Usuario	Personifica a quien se encarga del procesamiento de la información de los clientes, remesas, envíos y sus estatus.
Administrador	Representa a la persona encargada del procesamiento de las unidades de transporte, remesas, rutas, usuarios, afiliados y sus cuentas; además puede cubrir las funciones del usuario.

Tabla 3.4. Descripción de Actores (2/2).

ACTOR	DESCRIPCIÓN
Manejador BD	Representa el ente encargado de la manipulación de los datos y del resguardo e integridad de los mismos así como también de la conexión con la base de datos.

Fuente: Elaboración Propia

3.1.5.3 Descripción de Casos de Uso

La descripción de los casos de uso mostrados en la **Tabla 3.3** se presenta a continuación desde la **Tabla 3.5** hasta la **Tabla 3.13**, especificando los actores involucrados, las precondiciones, los flujos principales y los alternos existentes, así como también los caso de uso detallados para cada uno.

Tabla 3.5. Procesar Envío (1/2).

PROCESAR ENVÍO	
Actor	Usuario, Administrador
Descripción	Permite al actor recopilar toda la información necesaria para el procesamiento de un envío
Pre-condición	El sistema debe cargar la interfaz para el procesamiento del envío.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción Insertar Envío: 1.1. El actor ingresa la información del envío 1.2. El sistema verifica que se haya ingresado la información necesaria y la almacena	

Tabla 3.5. Procesar Envío (2/2).

Escenario de Éxito (Flujo Principal) (Continuación)
2. Si el actor selecciona la opción Consultar Envío:
2.1. El actor ingresa el número de guía
2.2. El sistema muestra la información del envío
3. Si el actor selecciona la opción Modificar Envío:
3.1. El actor ingresa el número de guía
3.2. El sistema muestra la información del envío
3.3. El actor modifica los campos deseados
3.4. El sistema verifica que estén completos los datos y los actualiza
Extensiones (Flujo Alterno)
1. El actor puede cancelar en cualquiera de los pasos
2. Si el actor ingresa un número de guía inválido se produce un error

Fuente: Elaboración Propia

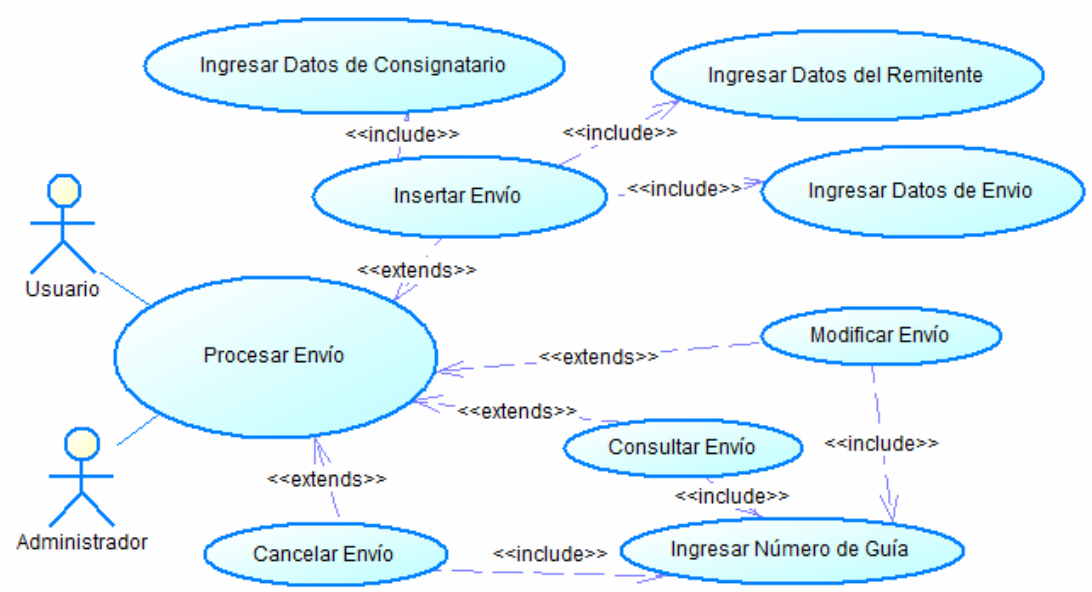

Figura 3.4. Diagrama de Caso de Uso de Procesar Envío

Fuente: Elaboración Propia

Tabla 3.6. Procesar Cliente.

PROCESAR CLIENTE	
Actor	Usuario, Administrador
Descripción	Este proceso permite al actor seleccionar las opciones disponibles para procesar la información asociada a los clientes, como insertar, modificar, eliminar y consultar.
Pre-condición	El sistema debe cargar la interfaz para el procesamiento de los clientes.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción Insertar Cliente: 1.1. El actor rellena los campos con la información del cliente 1.2. El sistema verifica que estén completos los datos y los almacena	
2. Si el actor selecciona la opción Consultar Cliente: 2.1. El actor ingresa la CI / RIF del cliente 2.2. El sistema muestra la información del cliente	
3. Si el actor selecciona la opción Modificar Cliente: 3.1. El actor ingresará la CI / RIF del cliente 3.2. El sistema muestra la información del cliente 3.3. El actor modifica los campos deseados 3.4. El sistema verifica los datos y almacena las modificaciones	
4. Si el actor selecciona la opción Eliminar Cliente: 4.1. El actor inserta la CI / RIF cliente 4.2. El sistema elimina al cliente	
Extensiones (Flujo Alterno)	
1. Si el actor puede cancelar en cualquiera de los pasos 2. Si el actor ingresa un cliente existente se produce un error 3. Si el actor ingresara una CI / RIF inválida se produce un error	

Fuente: Elaboración Propia

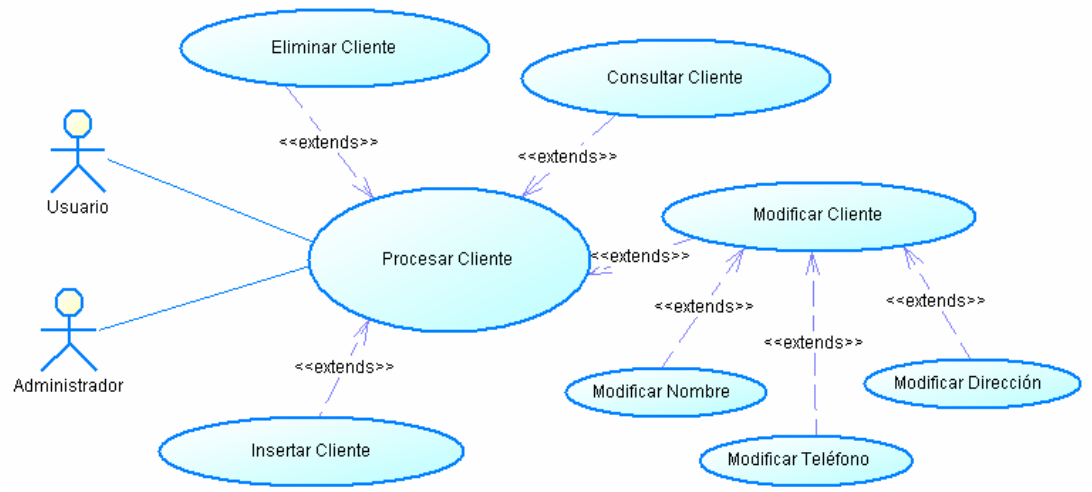

Figura 3.5. Diagrama de Caso de Uso de Procesar Cliente

Fuente: Elaboración Propia

Tabla 3.7. Monitorear Paquete (1/2).

MONITOREAR PAQUETE	
Actor	Usuario, Administrador
Descripción	Permite al actor consultar o modificar el estatus de un paquete en cualquier momento.
Pre-condición	El sistema debe cargar la interfaz para el monitoreo de paquetes.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción de Consultar Estatus del Paquete: 1.1. El actor ingresa el número de guía 1.2. El sistema muestra el estatus del paquete	
2. Si el actor selecciona la opción de Modificar Estatus del Paquete: 2.1. El actor ingresa el número de guía 2.2. El sistema muestra la información del estatus del envío 2.3. El actor modifica el estatus del envío 2.4. El sistema almacena la modificación	

Tabla 3.7. Monitorear Paquete (2/2).

Extensiones (Flujo Alterno)
1. Si el actor puede cancelar en cualquiera de los pasos
2. Si el actor ingresara un número de guía inválido se produce un error

Fuente: Elaboración Propia

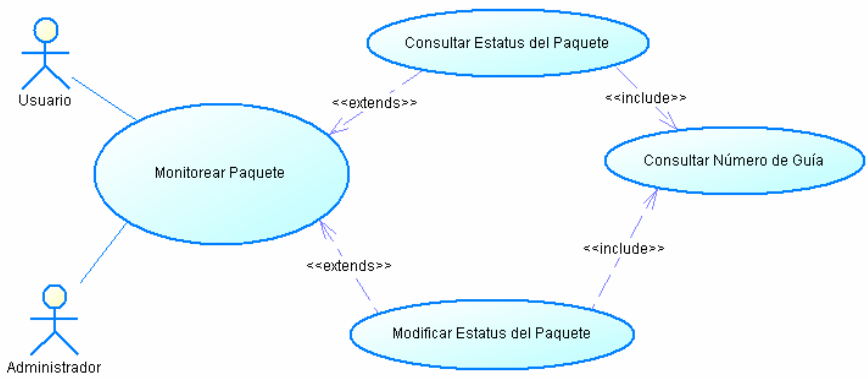

Figura 3.6. Diagrama de Caso de Uso de Monitorear Paquete

Fuente: Elaboración Propia

Tabla 3.8. Procesar Remesa (1/2).

PROCESAR REMESA	
Actor	Usuario, Administrador
Descripción	Permite al administrador llevar la relación de las guías que han sido transportadas por cada unidad.
Pre-condición	El sistema debe cargar la interfaz para el procesamiento de remesas.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción Crear Remesa: 1.1. El actor inserta la placa de la unidad de transporte del afiliado 1.2. El actor inserta los números de guías que desee 1.3. El sistema almacena la información de la remesa creada	

Tabla 3.8. Procesar Remesa (2/2).

Escenario de Éxito (Flujo Principal) (Continuación)
2. Si el actor selecciona la opción Modificar Remesa: 2.1. El actor ingresa el ID de la remesa 2.2. El sistema muestra la información de la remesa 2.3. El actor modifica la información deseada 2.4. El sistema almacena las modificaciones realizadas
3. Si el actor selecciona la opción Eliminar Remesa: 3.1. El actor inserta el ID de la remesa 3.2. El sistema elimina la remesa
Extensiones (Flujo Alterno)
1. Si el actor puede cancelar en cualquiera de los pasos
2. Si el actor ingresa una placa que no está registrada se produce un error
3. Si el actor ingresa un número de guía inválido se produce un error
4. Si el actor ingresa un ID de remesa inválido se produce un error

Fuente: Elaboración Propia

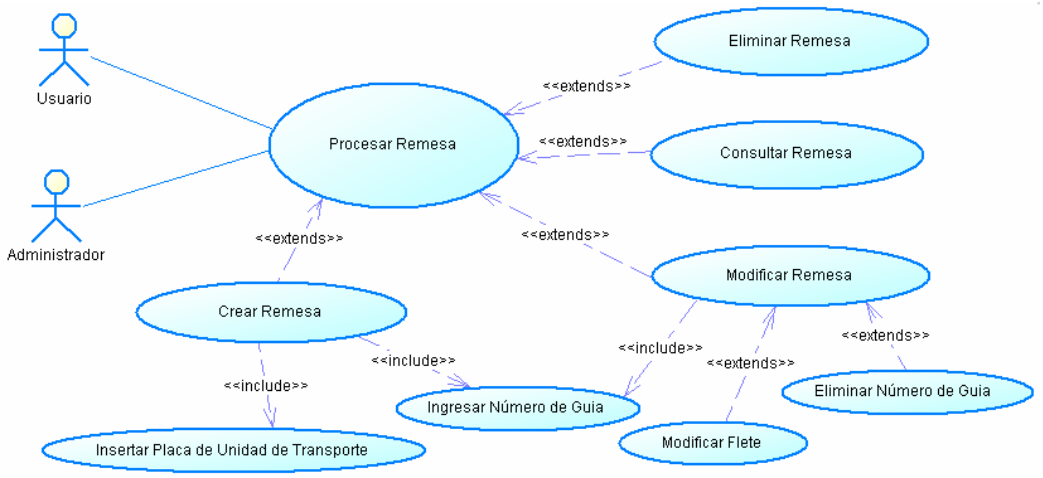

Figura 3.7. Diagrama de Caso de Uso de Procesar Remesa

Fuente: Elaboración Propia

Tabla 3.9. Procesar Ruta.

PROCESAR RUTA	
Actor	Administrador
Descripción	Este proceso permite al administrador ingresar, modificar, consultar y eliminar toda la información referente a las rutas que cubre la empresa.
Pre-condición	El sistema debe cargar la interfaz para el procesamiento de rutas.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción de Ingresar Ruta: 1.1. El actor rellena los campos con la información de la ruta 1.2. El sistema almacena la información de la nueva ruta	
2. Si el actor selecciona la opción de Consultar Ruta: 2.1. El actor inserta el ID de la ruta 2.2. El sistema muestra la información de la ruta	
3. Si el actor selecciona la opción de Eliminar Ruta: 3.1. El actor inserta el ID de la ruta 3.2. El sistema elimina la ruta	
4. Si el actor selecciona la opción de Modificar Ruta: 4.1. El actor inserta el ID de la ruta 4.2. El actor modifica los campos deseados 4.3. El sistema verifica los datos y almacena las modificaciones	
Extensiones (Flujo Alterno)	
1. Si el actor presiona cancelar en cualquiera de los pasos 2. Si el actor ingresa una ruta existente se produce un error 3. Si el actor ingresa un ID de ruta inválido se produce un error	

Fuente: Elaboración Propia

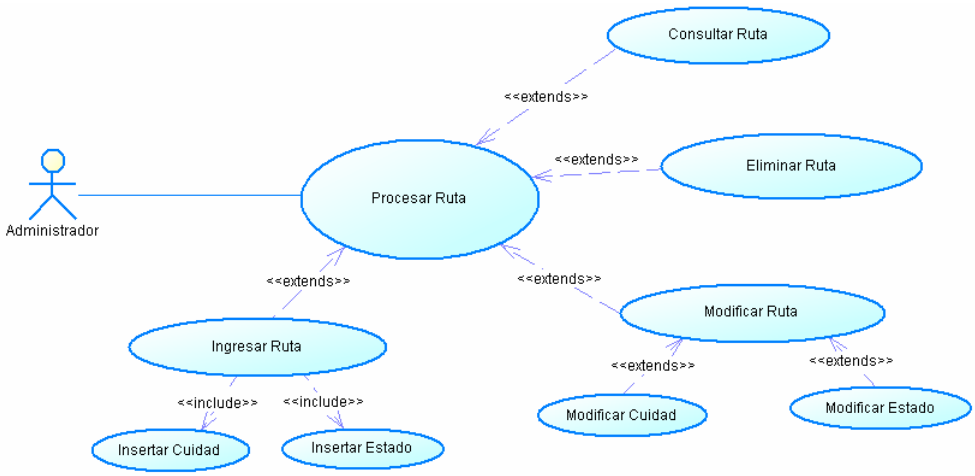

Figura 3.8. Diagrama de Caso de Uso de Procesar Ruta

Fuente: Elaboración Propia

Tabla 3.10. Procesar Afiliado (1/2).

PROCESAR AFILIADO	
Actor	Administrador
Descripción	Este proceso permite al administrador ingresar, modificar, consultar y eliminar toda la información referente a los socios de la empresa.
Pre-condición	El sistema debe cargar la interfaz para el procesamiento de afiliados.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción de Ingresar Afiliado: 1.1. El actor ingresa los datos del nuevo afiliado 1.2. El sistema verifica que la información esté completa y la almacena	
2. Si el actor selecciona la opción de Consultar Afiliado: 2.1. El actor ingresa la CI del afiliado 2.2. El sistema muestra la información del afiliado	

Tabla 3.10. Procesar Afiliado (2/2).

Escenario de Éxito (Flujo Principal) (Continuación)
3. Si el actor selecciona la opción de Eliminar Afiliado: 3.1. El actor ingresa la CI del afiliado 3.2. El sistema elimina el registro del afiliado
Extensiones (Flujo Alterno)
1. Si el actor puede cancelar en cualquiera de los pasos 2. Si el actor ingresa un afiliado existente se produce un error 3. Si el actor ingresa una CI inválida se produce un error

Fuente: Elaboración Propia

Figura 3.9. Diagrama de Caso de Uso de Procesar Afiliado

Fuente: Elaboración Propia

Tabla 3.11. Procesar Usuario.

PROCESAR USUARIO	
Actor	Administrador
Descripción	Le permite al actor ingresar, modificar, eliminar y consultar la información referente a cada usuario del sistema.
Pre-condición	Cargar la interfaz para el procesamiento de los usuarios.
Escenario de Éxito (Flujo Principal)	
1. Si el actor selecciona la opción de Insertar Usuario: 1.1. El actor rellena los campos con la información del nuevo usuario 1.2. El sistema almacena el nuevo registro	
2. Si el actor selecciona la opción de Consultar Usuario: 2.1. El actor ingresará el login del usuario 2.2. El sistema muestra la información de usuario	
3. Si el actor selecciona la opción de Eliminar Usuario: 3.1. El actor ingresa el login del usuario 3.2. El sistema eliminará el registro del usuario	
2. Si el actor selecciona la opción de Modificar Usuario: 2.1. El actor ingresa el login del usuario 2.2. El actor modifica los campos deseados 2.3. El sistema almacena las modificaciones realizadas	
Extensiones (Flujo Alterno)	
1. Si el actor puede cancelar en cualquiera de los pasos 2. Si el actor ingresa un usuario existente se produce un error 3. Si el actor ingresa un login inválido se produce un error	

Fuente: Elaboración Propia

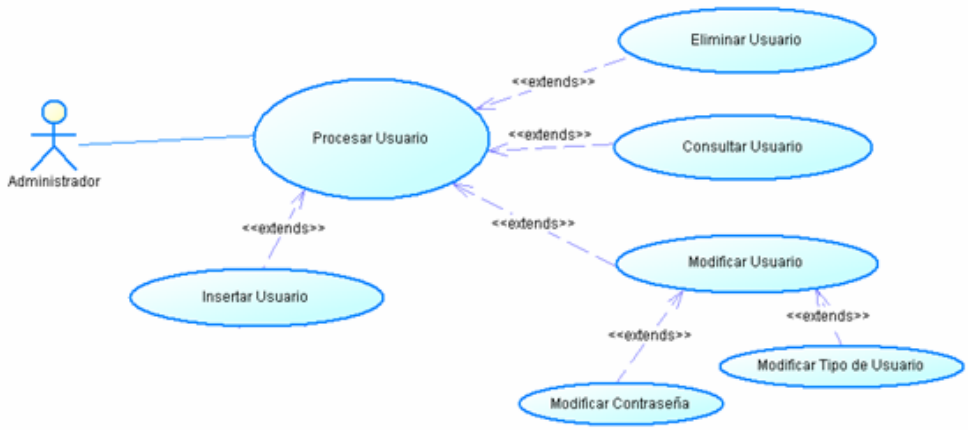

Figura 3.10. Diagrama de Caso de Uso de Procesar Usuario

Fuente: Elaboración Propia

Tabla 3.12. Procesar Unidad de Transporte (1/2).

PROCESAR UNIDAD DE TRANSPORTE	
Actor	Administrador
Descripción	Le permite al actor ingresar, modificar, eliminar y consultar la información referente a las unidades de transporte de los afiliados.
Pre-condición	Cargar la interfaz para el procesamiento de las unidades de transporte.
Escenario de Éxito (Flujo Principal)	
1. Si el actor presiona la opción de Ingresar Unidad de Transporte: 1.1. El actor rellena los campos con la información de la unidad 1.2. El sistema almacena la información de la nueva unidad	
2. Si el actor presiona la opción de Consultar Unidad de Transporte: 2.1. El actor inserta la placa de la unidad 2.2. El sistema muestra la información de la unidad	
3. Si el actor selecciona la opción de Modificar Unidad de Transporte: 3.1. El actor inserta la placa de la unidad 3.2. El actor modifica los campos deseados 3.3. El sistema almacena las modificaciones realizadas	

Tabla 3.12. Procesar Unidad de Transporte (2/2).

Escenario de Éxito (Flujo Principal) (Continuación)
4. Si el actor selecciona la opción de Eliminar Unidad de Transporte: 4.1. El actor inserta la placa de la unidad 4.2. El sistema eliminará la unidad
Extensiones (Flujo Alterno)
1. Si el actor puede cancelar en cualquiera de los pasos 2. Si el actor ingresa una unidad existente se produce un error 3. Si el actor ingresa una placa inválida se produce un error

Fuente: Elaboración Propia

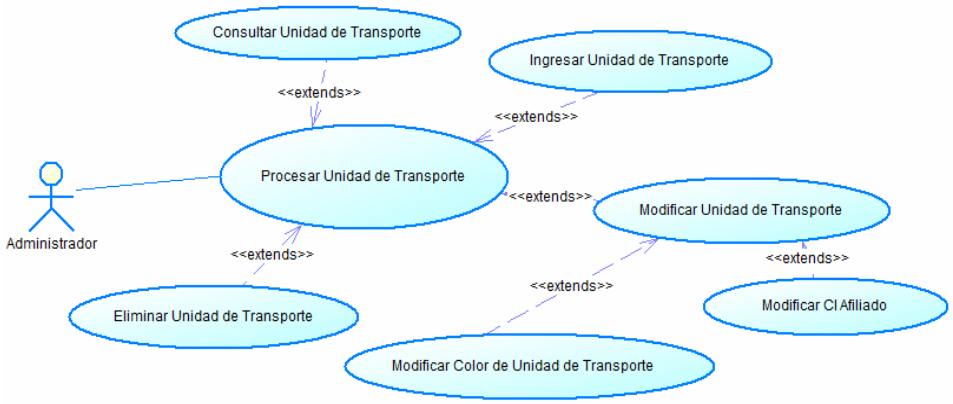

Figura 3.11. Diagrama de Caso de Uso de Procesar Unidad de Transporte

Fuente: Elaboración Propia

Tabla 3.13. Procesar Cuenta de Afiliado (1/2).

PROCESAR CUENTA DE AFILIADO	
Actor	Administrador
Descripción	Le permite al administrador consultar y modificar la cuenta de cada afiliado de la empresa, así como también, realizar pagos a los mismos.
Pre-condición	Cargar la interfaz para el procesamiento de las cuentas de los afiliados.

Tabla 3.13. Procesar Cuenta de Afiliado (2/2).

Escenario de Éxito (Flujo Principal)
1. Si el actor selecciona la opción de Consultar Cuenta de Afiliado: 1.1. El actor ingresa la CI del afiliado 1.2. El sistema muestra la información de la cuenta
2. Si el actor selecciona la opción de Eliminar Cuenta de Afiliado: 2.1. El actor ingresa la CI del afiliado 2.2. El sistema muestra la información de la cuenta consultada 2.3. El sistema elimina la cuenta del afiliado
3. Si el actor selecciona la opción de Realizar Pago: 3.1. El actor ingresa la CI del afiliado 3.2. El actor selecciona el tipo de pago que desea realizar 3.3. El sistema almacena la información sobre el pago realizado a la cuenta
Extensiones (Flujo Alterno)
1. Si el actor puede cancelar en cualquiera de los pasos 2. Si el actor ingresa una CI inválida se produce un error

Fuente: Elaboración Propia

Figura 3.12. Diagrama de Caso de Uso de Procesar Cuenta de Afiliado

Fuente: Elaboración Propia

3.2 FLUJO DE TRABAJO DE ANÁLISIS

El objetivo de este flujo de trabajo es traducir los requisitos a detalle de manera que especifique cómo implementar el sistema. En el análisis se obtiene una visión del sistema que se preocupa de “ver qué hace”, de modo que sólo se interesa por los requisitos funcionales.

A diferencia del lenguaje intuitivo utilizado en la captura de requisitos, en el flujo de trabajo análisis se utiliza un lenguaje basado en modelo de casos de uso y clases de análisis, que permite refinar el flujo anterior y comenzar a indagar en aspectos internos del sistema. En este flujo también se describe el diagrama de paquetes de análisis, que permite precisar y fundamentar la línea base de la arquitectura del sistema.

3.2.1 Análisis de los Casos de Uso

El análisis de los requisitos capturados en forma de casos de uso es un flujo de trabajo que permite de manera iterativa e incremental la construcción del modelo de análisis del sistema. Este modelo de análisis crece a medida que se analizan más casos de uso, ofreciendo una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que ayude a estructurar el sistema entero.

3.2.2 Identificación de las Clases de Análisis

Las clases de análisis, también llamados objetos de análisis, son clases estereotipadas que representan un modelo conceptual para los elementos del sistema que tienen responsabilidad y comportamiento. Hay tres tipos de clases de análisis, las cuales son usadas en todo el modelo de análisis:

- Clases de interfaz
- Clases de entidad
- Clases de control

Una clase de análisis representa una abstracción de una o varias clases y/o subsistemas del diseño del sistema, y constituye una colección de clases que describe los requisitos funcionales del sistema.

3.2.2.1 Clases de Interfaz

Una interfaz es una colección de operaciones que representan servicios ofrecidos por una clase o componente. La clase realiza (o suministra una realización de) varias interfaces. El prototipo de clase de interfaz se muestra en la **Figura 3.13**, a continuación se detallan las clases de interfaz que se identificaron en el modelo de análisis, para el sistema SCAAMP (*ver* **Tabla 3.14**).

Figura 3.13. Prototipo de Clase de Interfaz

Fuente: Elaboración Propia

Tabla 3.14. Clases de interfaz (1/3).

CLASES DE INTERFAZ	DESCRIPCIÓN
UI Insertar Cliente	Permite al administrador y usuario insertar un nuevo cliente en el sistema.
UI Modificar Cliente	Permite al administrador y usuario modificar la información de un cliente.
UI Eliminar Cliente	Permite al administrador y usuario eliminar un cliente del sistema.
UI Consultar Cliente	Permite al administrador y usuario consultar los datos de un cliente.

Tabla 3.14. Clases de interfaz (2/3).

CLASES DE INTERFAZ	DESCRIPCIÓN
UI Insertar Envío	Permite al administrador y usuario tramitar toda la información necesaria para la realización de un nuevo envío.
UI Cancelar Envío	Permite al administrador y usuario cancelar un envío que ya ha sido tramitado.
UI Consultar Envío	Permite al administrador y usuario consultar la información de un envío.
UI Modificar Envío	Permite al administrador y usuario modificar los datos de un envío.
UI Consultar Estatus del Paquete	Permite al administrador y usuario consultar la estatus de un paquete en cualquier momento.
UI Modificar Estatus del Paquete	Permite al administrador y usuario modificar el estatus de un envío en un momento dado.
UI Ingresar Afiliado	Permite al administrador insertar un afiliado de la empresa al sistema.
UI Modificar Afiliado	Permite al administrador modificar la información de un afiliado.
UI Eliminar Afiliado	Permite al administrador eliminar un afiliado del sistema.
UI Consultar Afiliado	Permite al administrador consultar los datos de un afiliado.
UI Ingresar Unidad de Transporte	Permite al administrador insertar una nueva unidad de transporte asociada a un afiliado.
UI Modificar Unidad de Transporte	Permite al administrador modificar la información de una unidad de transporte.
UI Eliminar Unidad de Transporte	Permite al administrador eliminar una unidad de transporte del sistema.
UI Consultar Unidad de Transporte	Permite al administrador consultar los datos de una unidad de transporte.
UI Insertar Usuario	Permite al administrador insertar un nuevo usuario del sistema.
UI Modificar Usuario	Permite al administrador modificar la información de un usuario.
UI Eliminar Usuario	Permite al administrador eliminar un usuario del sistema.
UI Consultar Usuario	Permite al administrador consultar los datos de un usuario.
UI Ingresar Ruta	Permite al administrador insertar una nueva ruta al sistema.
UI Modificar Ruta	Permite al administrador modificar la información de una ruta.

Tabla 3.14. Clases de interfaz (3/3).

CLASES DE INTERFAZ	DESCRIPCIÓN
UI Eliminar Ruta	Permite al administrador eliminar una ruta del sistema.
UI Consultar Ruta	Permite al administrador consultar los datos de una ruta.
UI Crear Remesa	Permite al administrador y usuario crear nuevas relaciones guías transportadas por unidad (remesa)
UI Modificar Remesa	Permite al administrador y usuario modificar las remesas.
UI Eliminar Remesa	Permite al administrador y usuario eliminar las remesas.
UI Consultar Remesa	Permite al administrador y usuario consultar las remesas.
UI Consultar Cuenta de Afiliado	Permite al administrador consultar las cuentas asociadas a los afiliados.
UI Eliminar Cuenta de Afiliado	Permite al administrador consultar las cuentas de los afiliados.
UI Realizar Pago	Permite al administrador realizar pagos a las cuentas de los afiliados

Fuente: Elaboración Propia

3.2.2.2 Clase de Control

Una operación es un servicio que una instancia de la clase puede realizar. En la **Figura 3.14** se puede observar el estereotipo de la clase de control, a continuación se especificarán las clases que se encontraron en el diseño de análisis, para el sistema SCAAMP (*ver* **Tabla 3.15**).

Figura 3.14. Prototipo de Clase de Control

Fuente: Elaboración Propia

Tabla 3.15. Clases de control (1/2).

CLASES DE CONTROL	DESCRIPCIÓN
Gestor Procesar Cliente	Actualiza los registros de clientes en el sistema.
Gestor Insertar Cliente	Inserta los datos de los nuevos clientes.
Gestor Modificar Cliente	Modifica los datos de los clientes.
Gestor Eliminar Cliente	Elimina los datos de los clientes
Gestor Consultar Cliente	Consulta los datos de los clientes.
Gestor Procesar Envío	Actualiza los registros de los envíos en el sistema.
Gestor Insertar Envío	Procesa la información para tramitar nuevos envíos.
Gestor Cancelar Envío	Cancela envíos del sistema.
Gestor Consultar Envío	Consulta la información de los envíos.
Gestor Modificar Envío	Modifica información de los envíos.
Gestor Procesar Estatus del Paquete	Actualiza los registros del estatus de un paquete en el sistema.
Gestor Consultar Estatus Del Paquete	Consulta el estatus de los paquetes.
Gestor Modificar Estatus Del Paquete	Modifica el estatus de los paquetes en un momento dado.
Gestor Procesar Unidad de Transporte	Actualiza los registros de las unidades de transporte en el sistema.
Gestor Ingresar Unidad De Transporte	Ingresa los datos de las nuevas unidades de transporte al sistema.
Gestor Modificar Unidad De Transporte	Modifica datos de las unidades de transporte.
Gestor Eliminar Unidad De Transporte	Elimina del sistema los datos de las unidades de transporte.
Gestor Consultar Unidad De Transporte	Consulta los datos de las unidades de transporte registradas en el sistema.
Gestor Procesar Afiliado	Actualiza los registros de los afiliados en el sistema.
Gestor Ingresar Afiliado	Inserta los datos de los nuevos afiliados.
Gestor Modificar Afiliado	Modifica los datos de los afiliados.
Gestor Eliminar Afiliado	Elimina los datos de los afiliados.
Gestor Consultar Afiliado	Consulta los datos de los afiliados.
Gestor Procesar Usuario	Actualiza los registros de los usuarios en el sistema.
Gestor Insertar Usuario	Inserta los datos de los nuevos usuarios del sistema.
Gestor Modificar Usuario	Modifica los datos de los usuarios del sistema.

Tabla 3.15. Clases de control (2/2).

CLASES DE CONTROL	DESCRIPCIÓN
Gestor Eliminar Usuario	Elimina los datos de los usuarios del sistema.
Gestor Consultar Usuario	Consulta los datos de los usuarios del sistema.
Gestor Procesar Ruta	Actualiza los registros de las rutas en el sistema.
Gestor Ingresar Ruta	Inserta los datos de las nuevas rutas.
Gestor Modificar Ruta	Modifica los datos de las rutas.
Gestor Eliminar Ruta	Elimina los datos de las rutas.
Gestor Consultar Ruta	Consulta los datos de las rutas.
Gestor Procesar Remesa	Actualiza los registros de las remesas en el sistema.
Gestor Crear Remesa	Crea nuevas remesas en el sistema.
Gestor Modificar Remesa	Modifica las remesas existentes en el sistema.
Gestor Eliminar Remesa	Elimina la información de las remesas.
Gestor Consultar Remesa	Consulta las remesas en el sistema.
Gestor Procesar Cuenta de Afiliado	Actualiza los registros de las cuentas de los afiliados en el sistema.
Gestor Consultar Cuenta de Afiliado	Consulta el estado de las cuentas de los afiliados.
Gestor Eliminar Cuenta de Afiliado	Elimina las cuentas de los afiliados del sistema.
Gestor Realizar Pago	Realiza pagos a las cuentas de los afiliados.

Fuente: Elaboración Propia

3.2.2.3 Clases de Entidad

La clase de entidad permite modelar la información. La **Figura 3.15** muestra el prototipo de la clase entidad, a continuación de especifican las clases encontradas en el modelo de análisis, para el sistema SCAAMP (*ver* **Tabla 3.16**).

Figura 3.15. Prototipo de Clase de Entidad.

Fuente: Elaboración Propia

Tabla 3.16, Clases de Entidad.

CLASES DE ENTIDAD	DESCRIPCIÓN
Cliente	Representa toda la información relacionada con los clientes de la empresa.
Cuenta de Afilado	Representa la información que permite llevar el control administrativo de cada afiliado.
Envío	Hace referencia a la información almacenada de cada envío tramitado a través del sistema.
Afiliado	Constituye toda la información perteneciente a los socios de la empresa.
Usuario	Representa a cada uno de los empleados que desempeñan un rol en el sistema.
Unidad de Transporte	Significa los datos concernientes a las unidades de transporte que posee cada afiliado de la empresa.
Ruta	Representa los detalles de cada ruta que cubren las unidades de transporte pertenecientes a la empresa.
Remesa	Representa toda la información sobre las guías que han sido transportadas por cada unidad.

Fuente: Elaboración Propia

3.2.3 Diagramas de Clases de Análisis

Una vez identificado los casos de uso del sistema y luego de puntualizar cada uno de ellos, se procedió a generar el Diagrama de Clases de Análisis del mismo. Para ello, se analizó cada caso de uso representando al sistema como una estructura de clasificadores que revela como debería ser diseñada la estructura del sistema, mediante una abstracción de una o varias clases básicas, lo que constituye una colección de clases que describe los requisitos funcionales del sistema.

Los diagramas de clases de análisis se utilizan para representar y modelar los atributos y asociaciones que poseen las clases, bien sea del sistema en general o de un caso de uso en concreto.

3.2.3.1 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Envío

El diagrama de clases de análisis para el caso de uso Procesar Envío se describe la forma en que se maneja la información relacionada con los envíos en el sistema. En este diagrama tanto el usuario, como el administrador del sistema usan las interfaces a través del gestor procesar envío para insertar, cancelar, modificar o consultar los registros de los envíos tramitados por la empresa.

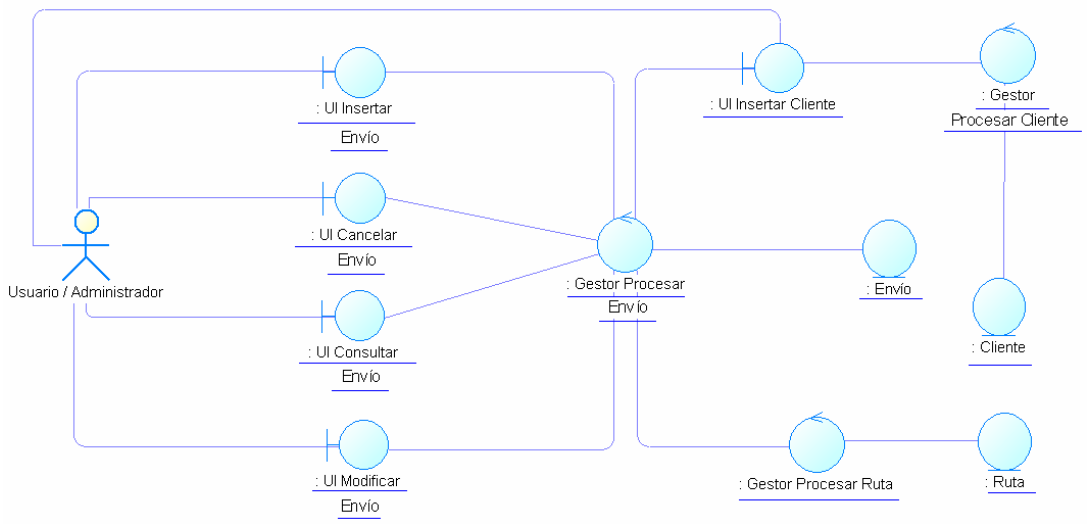

Figura 3.16. Diagrama de Clase de Análisis para el Caso de Uso Procesar Envío

Fuente: Elaboración Propia

Como se puede observar en la Figura 3.16, el análisis del caso de uso Procesar Envío destaca las clases interfaz denominada Interfaz Insertar Envío, que permite procesar un nuevo envío. La interfaz Cancelar Envío que permitirá cancelar cualquier envío que haya sido tramitado previamente. La interfaz Consultar Envío que permitirá buscar en el sistema los envíos procesados. La interfaz Modificar Envío que permitirá modificar cualquier

dato; las clases de control Gestor Procesar Envío, donde se ejecutan todos los procesos concernientes a un envío, Gestor Procesar Cliente, que permite insertar los datos del consignatario (cliente), Gestor Procesar Ruta, por medio del cual se busca la ruta del envío. Por último las clases entidad Envío, Cliente y Ruta, donde se almacenan los datos del envío, los datos de los clientes, y las rutas respectivamente.

3.2.3.2 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Remesa

Las actividades relacionadas con la administración de las remesas pueden ser realizadas por el Usuario o Administrador, como se observa en la **Figura 3.17** con el diagrama de clases de análisis de los casos de uso Procesar Remesa, donde se especificaron cuatro clases de interfaz las cuales interactúan con ambos actores por medio del proceso Gestor Procesar Remesa. La Interfaz Crear Remesa, permite crear las relaciones de guías transportadas por cada unidad asociada a los afiliados. La clase de interfaz Modificar Remesa, que permite modificar la información de las remesas. La interfaz Eliminar Remesa a través de la cual los actores pueden eliminar las remesas registradas en el sistema. La interfaz Consultar Remesa, que permite consultar toda la información de cada remesa registrada. Además se especificaron las clases entidad Remesa, donde se registra todo los datos de dichas relaciones; Envío, donde se almacenan los registros de los envíos; Afiliado y Unidad de transporte, en las cuales se almacenan los registros de los afiliado y las unidad de transporte asociadas a él, respectivamente.

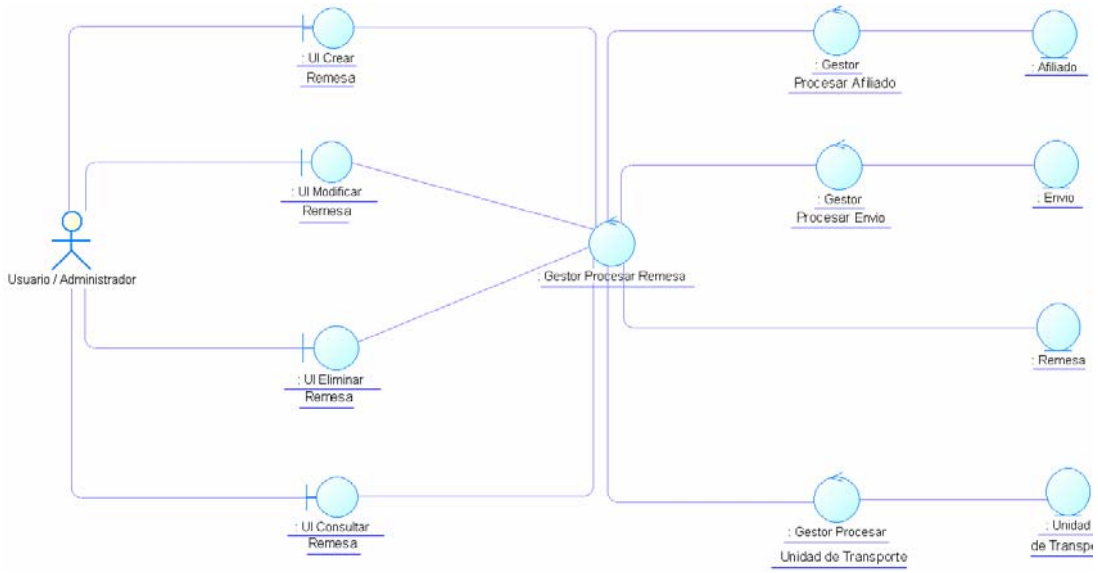

Figura 3.17. Diagrama de Clase de Análisis para el Caso de Uso Procesar Remesa
Fuente: Elaboración Propia

3.2.3.3 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Cuenta de Afiliado

En el caso de uso Procesar Cuenta de Afiliado (*ver* **Figura 3.18**) se presenta el actor con los privilegios correspondientes para este caso, la pantalla de interacción para la administración de las cuentas de los afiliados, así como también pone a la disposición del actor un menú con las opciones: consultar y eliminar cuenta de afiliado, y realizar pago.

Para realizar el análisis del caso de uso procesar cuenta de afiliado se identificaron en primer lugar las clases de análisis: Interfaz Consultar Cuenta de Afiliado, que es la interfaz que le permitirá revisar la ficha administrativa de cada afiliado, Interfaz Eliminar Cuenta de Afiliado, donde podrá suprimir del sistemas los registros de las cuentas. Se denotan las clases de control Gestor de Cuenta de Afiliado, que se encarga de manera general de los

procesos relacionados con la administración de las mismas, y los Gestores Procesar Afiliado y Procesar Remesa que permitirán procesar los datos de cada afiliado y de las remesas que posee.

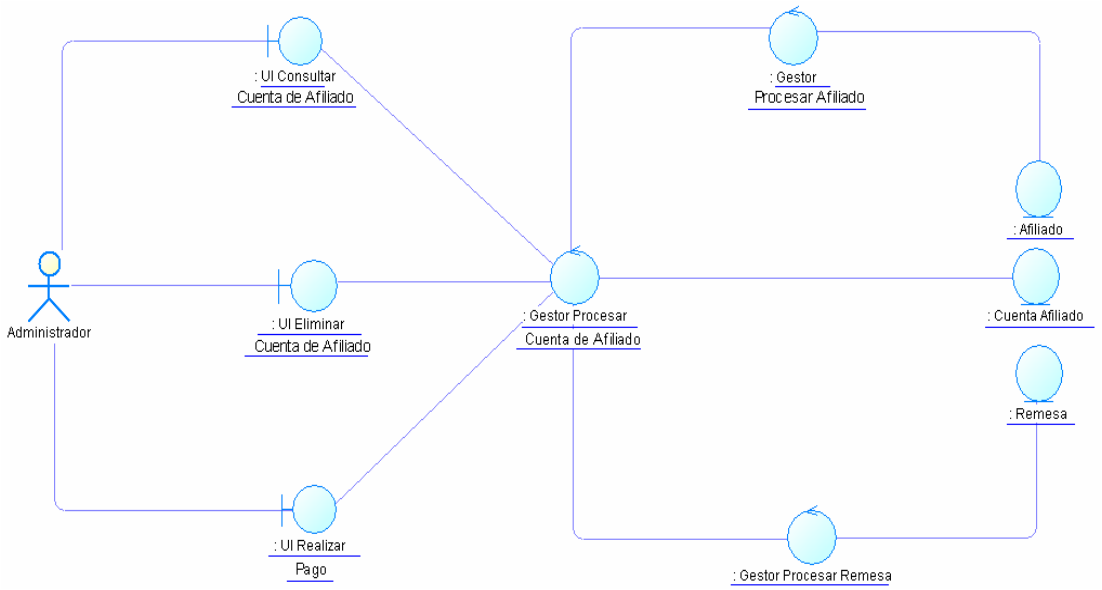

Figura 3.18. Diagrama de Clase de Análisis para el Caso de Uso Procesar Cuenta de Afiliado

Fuente: Elaboración Propia

3.2.4 Diagramas de Colaboración

Los diagramas de colaboración muestran las interacciones entre objetos creando enlaces entre ellos y añadiendo mensajes a esos enlaces. Describen el proceso a detalle señalando la interacción y comunicación entre las clases, actores y entidades. El nombre de un mensaje debería denotar el propósito del objeto invocante en la interacción con el objeto invocado.

3.2.4.1 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Envío

En el diagrama de colaboración del caso de uso Procesar Envío, como se muestra en la **Figura 3.19**, el usuario o administrador tiene cuatro flujos básicos que representan las opciones que puede realizar, las cuales son: Solicitar insertar envío, solicitar cancelar envío, solicitar consultar envío y solicitar modificar envío.

En el caso de solicitar un nuevo envío, se despliega la interfaz Insertar Envío, en la cual se procede a insertar los datos correspondientes al envío (1), para luego procesarlos con el Gestor Procesar Envío (2), el cual además verifica a través de la CI / RIF ingresada los datos del cliente, si este no se encuentra registrado, se despliega la interfaz Insertar Cliente, y se procede a insertar los datos (3), los cuales son procesados y validados por el gestor Procesar Cliente (4), y almacenados en la base de datos (5), luego a través del Gestor Procesar Ruta se gestionan los datos de la ruta (6), se buscan en la base de datos (7), y se procesan por el Gestor Procesar Rutas (8), para enviarlos al Gestor Procesar Envío (9) y finalmente se almacenan todos estos datos en el sistema (10), y culmina el registro del nuevo envío.

Si el usuario solicita cancelar un envío, se despliega la interfaz Cancelar Envío, en la cual se inserta el número de guía (11), luego a través del Gestor Procesar Envío se procede a buscar los datos del envío a anular (12), y finalmente se cancela (13).

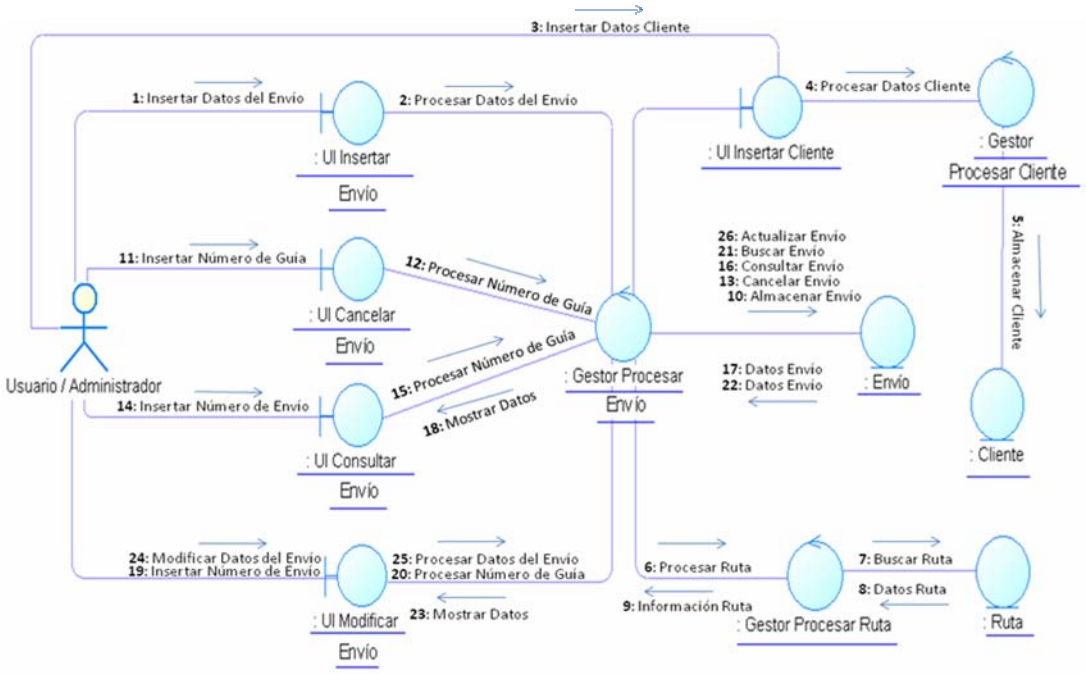

Figura 3.19. Diagrama de Colaboración para el Caso de Uso Procesar Envío

Fuente: Elaboración Propia

Para consultar un envío, el usuario extiende la interfaz Consultar Envío, para seleccionar que envío desea consultar insertando el número de guía (14), éste es analizado por el Gestor Procesar Envío (15), el cual se encargará de solicitar (16) y procesar el resultado (17), para luego mostrar de toda la información referente a la consulta realizada (18).

Cuando el usuario selecciona modificar un envío, se activa la interfaz Modificar Envío, en la cual se inserta el número de guía (19), éste se procesa a través del Gestor Procesar Envío (20), para llevar a cabo la consulta de los datos correspondientes al envío (21), y retornarlos al gestor para ser procesados (22) y mostrados (23), seguidamente el usuario modifica los datos (24), y estos son analizados nuevamente por el Gestor Procesar Envío (25), para ser actualizados en el registro (26).

3.2.4.2 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Remesa

En el diagrama de colaboración del caso de uso Procesar Remesa, como se muestra en la **Figura 3.20**, el usuario tiene las opciones de Crear Remesa, Consultar Remesa, Modificar Remesa y Eliminar Remesa.

Si el usuario solicita crear remesa, se activa la interfaz Crear Remesa, donde se ingresan los datos correspondientes de la remesa (1), luego éstos son procesados por el Gestor Procesar Remesa (2), seguidamente por medio del Gestor Procesar Unidad de Transporte se gestionan las placas de la unidad introducida (3), para realizar la consulta a la base de datos (4), y luego retornar el resultado (5) al Gestor Procesar Remesa (6); posteriormente a través del Gestor Procesar Afiliado se procesan los datos del afiliado (7) y se solicitan a la base de datos del sistema (8), la cual devuelve el resultado (9) al Gestor Procesar Remesa (10); de igual manera el Gestor Procesar Envíos, analiza los datos (11), que son solicitados (12), y regresados por la base de datos (13), para que los reciba el Gestor Procesar Remesa (14); luego toda esa información es mostrada al usuario (15), para que la verifique y proceda a almacenar el registro de las remesas (16).

En el caso que el usuario desee modificar una remesa, se despliega la interfaz Modificar Remesa, en la cual se ingresar el ID de la Remesa (17), el cual es procesado por el Gestor Procesar Remesa (18), para luego realizar la consulta a la base de datos (19), y analizar la información que esta le devuelva (20), a continuación se procesa el número de guía que se quiere añadir a la remesa a través del Gestor Procesar Envíos (21), este a su vez solicita la información a la base de datos (22), la cual devuelve los datos de la remesa requerida (23), para hacerlos llegar al Gestor Procesar Remesa

(24), posteriormente el usuario visualiza la información (25), luego modifica la información sobre el flete de cada envió registrado o agregado en dicha remesa (26), para que finalmente el Gestor Procesar Remesa se encargue de procesar la nueva información (27) y actualizarla en los registros (28).

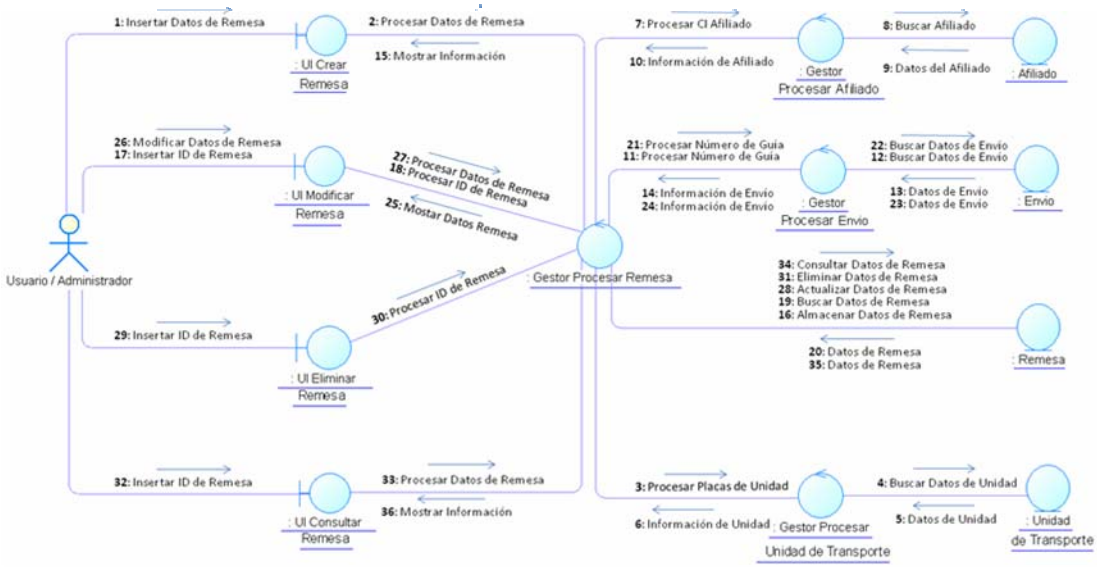

Figura 3.20. Diagrama de Colaboración para el Caso de Uso Procesar Remesa

Fuente: Elaboración Propia

Para eliminar una remesa, el usuario activa la interfaz Eliminar Remesa; en ella se le solicita el ID de la Remesa que desea eliminar (29), para luego ser procesado por el Gestor Procesar Remesa (30), y ser eliminada del sistema (31).

Al seleccionar la opción de consultar remesa, el sistema despliega la interfaz Consultar Remesa, en la cual el usuario inserta el ID de la Remesa (32), para que el Gestor Procesar Remesa analice y (33), realice con esta información la consulta a la base de datos (34), para que ésta le retorne el registro correspondiente (35), para su posterior visualización (36).

3.2.4.3 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Cuenta de Afiliado

La **Figura 3.21**, representa el diagrama de colaboración del caso de uso Procesar Cuenta de Afiliado, en el cual se muestra que el administrador tiene tres flujos básicos representando a las opciones que éste puede realizar.

Si el administrador solicita consultar la cuenta de un asociado, el sistema activa la interfaz Consultar Cuenta de Afiliado, en ésta se inserta el ID de la Cuenta (1), el cual se procesa por medio del Gestor Procesar Cuenta Afiliado (2); además este gestor se encarga de procesar los datos de los afiliados registrados (3), llevando a cabo la solicitud de los mismos al sistema (4), para su posterior análisis (5) y enviarlos al Gestor Procesar Cuenta Afiliado (6); de igual manera, el Gestor Procesar Remesa, se encarga del gestionamiento de los datos de las remesas asociadas al afiliado (7), los cuales son invocados (8), y retornados a través de una solicitud a la base de datos (9), hasta el Gestor Procesar Cuenta Afiliado (10), finalmente se procede a consultar la información de la cuenta en la base de datos (11) para su procesamiento a través del Gestor Cuenta de Afiliado (12) y posteriormente visualizarla (13).

Cuando el administrador solicite la eliminación de una cuenta perteneciente a un afiliado, se despliega la interfaz Eliminar Cuenta de Afiliado, allí se ingresa el ID de la Cuenta (14), seguidamente se gestiona esta información a través del Gestor Procesar Cuenta de Afiliado (15), quien a su vez se encarga de solicitar los datos del afiliado por medio del Gestor Procesar Afiliado (16), para realizar su búsqueda (17), análisis (18), y retornar los datos al Gestor Procesar Cuenta Afiliado (19); a continuación, el

Gestor Procesar Remesa, será el encargado de procesar los datos de las remesas asociadas al afiliado (20), para ejecutar su consulta en el registro (21), y examinar el resultado de la misma (22), para retornarlos al Gestor Procesar Cuenta Afiliado (23), a continuación se solicitan los datos de la cuenta a la base datos (24), para procesarlos a través del Gestor Procesar Cuenta Afiliado (25), para luego visualizarlos (26); seguidamente, el usuario selecciona las remesas que desea eliminar (27), para que sean procesadas por el Gestor Cuenta de Afilado (28), y finalmente actualizar los registros de la cuenta (29).

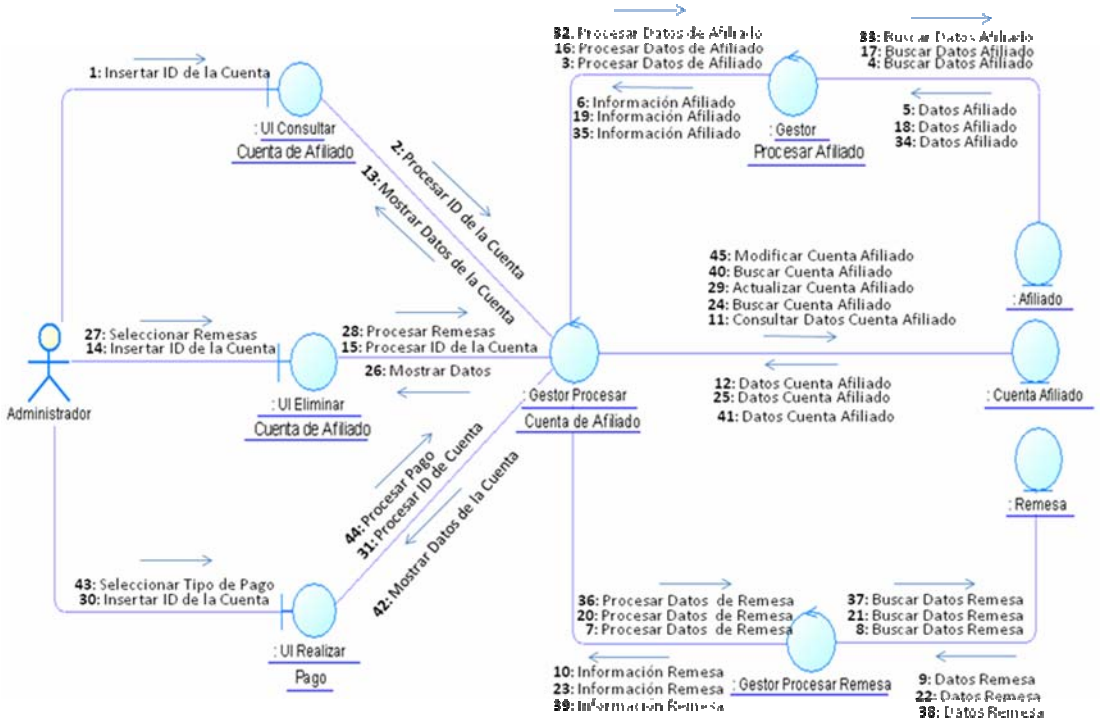

Figura 3.21. Diagrama de Colaboración para el Caso de Uso Procesar Cuenta de Afiliado
Fuente: Elaboración Propia

Para realizar un pago, se activa la interfaz Realizar Pago, en la cual el administrador ingresa el ID de la Cuenta (30), para que éste sea gestionado

por el Gestor Procesar Cuenta de Afiliado (31), éste además solicita los datos del afiliado a través del Gestor Procesar Afiliado (32), quién realiza la consulta al sistema (33), para su procesamiento (34), y retorno al Gestor Procesar Cuenta Afiliado (35); a continuación, el Gestor Procesar Remesa, se encarga de gestionar la información de las remesas asociadas al afiliado (36), para ejecutar su búsqueda en el registro (37), y luego analizarla (38), para enviarla al Gestor Procesar Cuenta Afiliado (39); seguidamente, se busca toda la información asociada a la cuenta del afiliado (40), para procesarla (41) y mostrarla al usuario (42); finalmente se selecciona el tipo de pago que se desea realizar (43), para su procesamiento (44) y modificar la cuenta del afiliado con el nuevo monto cancelado o abono (45).

3.2.5 Paquetes de Análisis

El propósito del análisis de la arquitectura es esbozar el modelo de análisis y la arquitectura mediante la identificación de paquetes de análisis. La descripción de la arquitectura contiene una vista de la arquitectura del modelo de análisis, que muestra sus artefactos significativos para la arquitectura como la descomposición del modelo de análisis en paquetes de análisis y sus dependencias.

3.2.5.1 Identificación de los Paquetes de Análisis

Un paquete es una forma de agrupar clases en modelos grandes, pueden tener asociaciones de dependencias o de generalizaciones entre ellos. Los paquetes de análisis proporcionan un medio para organizar los artefactos del modelo de análisis en piezas manejables. En general, los paquetes de análisis deben ser cohesivos (fuertemente relacionados), y deben ser débilmente acoplados (con mínima dependencia).

Los paquetes de análisis pueden constar de clases de análisis, de realizaciones de casos de uso, y de otros paquetes de análisis.

A continuación desde la **Figura 3.22** hasta la **Figura 3.31** se presentan cada uno de los paquetes del sistema con sus respectivas clases de análisis.

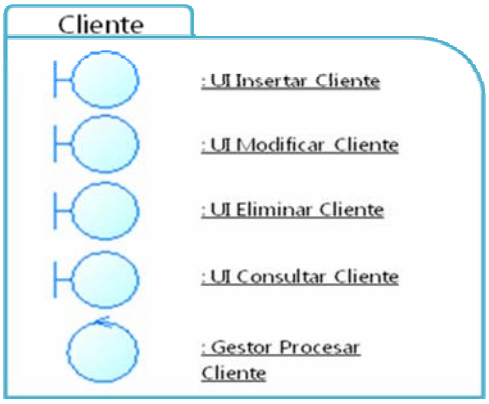

Figura 3.22. Paquete de Análisis: Cliente

Fuente: Elaboración Propia

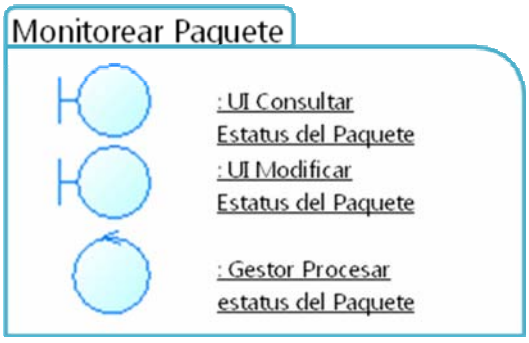

Figura 3.23. Paquete de Análisis: Monitorear Paquete

Fuente: Elaboración Propia

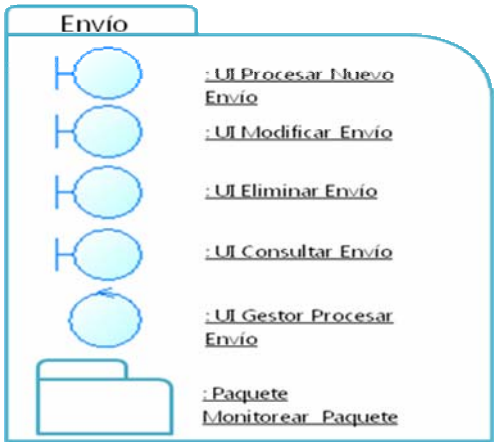

Figura 3.24. Paquete de Análisis: Envío

Fuente: Elaboración Propia

Figura 3.25. Paquete de Análisis: Usuario

Fuente: Elaboración Propia

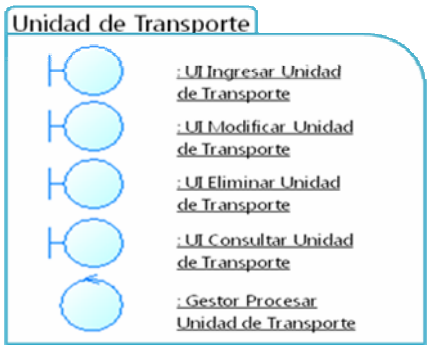

Figura 3.26. Paquete de Análisis: Unidad de Transporte

Fuente: Elaboración Propia

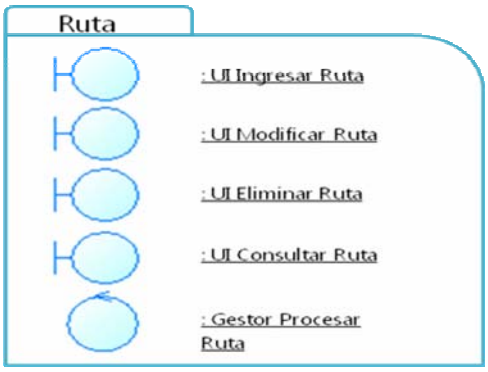

Figura 3.27. Paquete de Análisis: Ruta

Fuente: Elaboración Propia

Figura 3.28. Paquete de Análisis: Cuenta de Afiliado

Fuente: Elaboración Propia

Figura 3.29. Paquete de Análisis: Afiliado

Fuente: Elaboración Propia

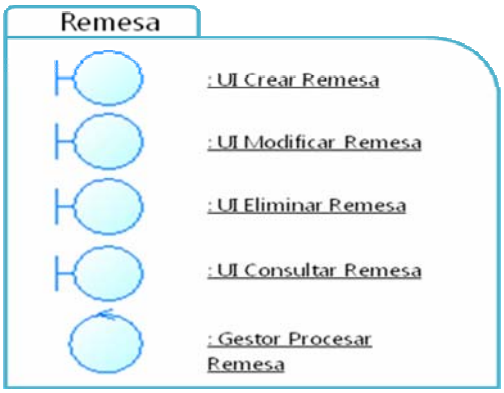

Figura 3.30. Paquete de Análisis: Remesa

Fuente: Elaboración Propia

Figura 3.31. Paquete de Análisis: Base de Datos

Fuente: Elaboración Propia

La identificación de paquetes de análisis permite organizar el modelo de análisis en unidades que, básicamente, engloban los requerimientos funcionales del sistema definidos con anterioridad como casos de uso en el modelo de casos de uso.

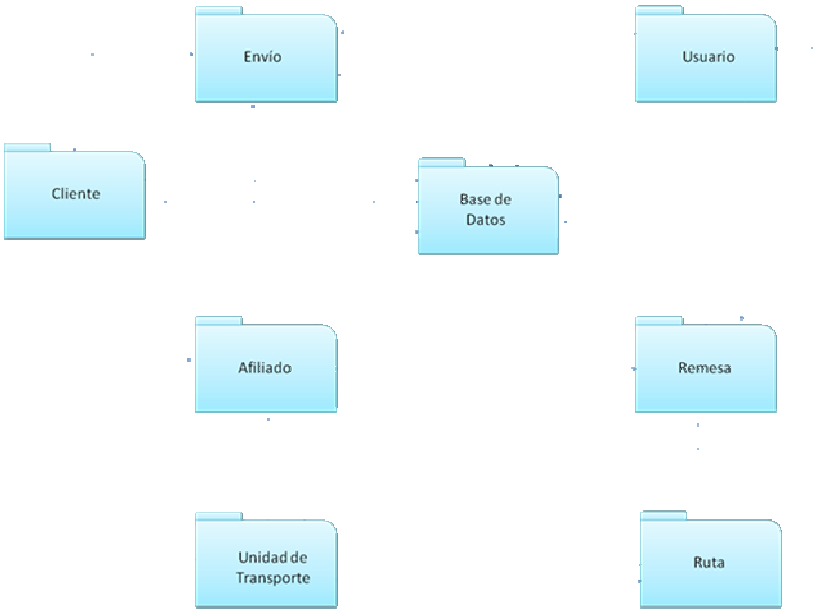

Figura 3.32. Diagrama de Paquete de Análisis del Sistema SCAAMP

Fuente: Elaboración Propia

Siguiendo las directrices del Proceso Unificado de Desarrollo de Software, se realizan una serie de actividades cuyos resultados constituyen la versión inicial del modelo de análisis (*ver* **Figura 3.32**), el cual contiene los paquetes Envío, Usuario, Cliente, Ruta, Unidad de Transporte, Base de Datos, Afiliado y Remesa.

3.3 FLUJO DE TRABAJO DE DISEÑO

3.3.1 Interfaz de Usuario

La interfaz de usuario, (*ver* **Figura 3.33)**, es aquella a la que accede cualquiera de los actores para llevar a cabo el procesamiento de información.

Esta se encuentra dividida en tres áreas; en la parte superior izquierda se encuentra la identificación del sistema, en ella se observa el nombre y la versión del software; a continuación se encuentra la barra de opciones, en donde están ubicados todos los botones a través de los cuales se accede a los diferentes casos de uso; por último se tiene el área de trabajo, en la cual se van desplegando las distintas interfaces que se van activando.

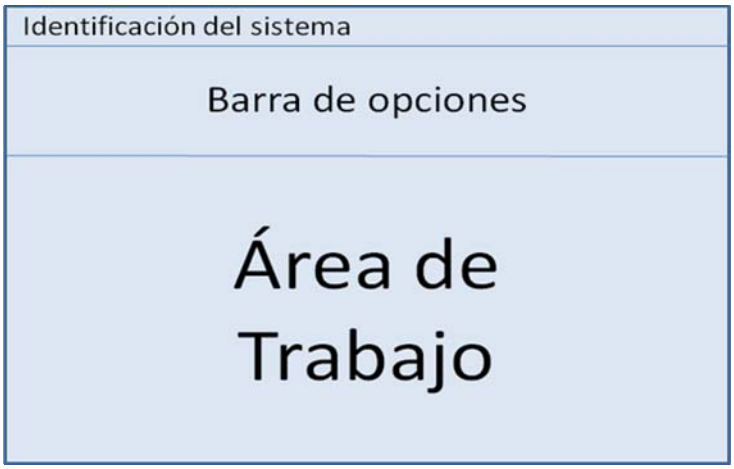

Figura 3.33. Interfaz Principal
Fuente: Elaboración Propia

3.4 EVALUACIÓN DE LA FASE DE INICIO

En este capítulo se llevó a cabo la fase de inicio, la cual es la primera fase del Proceso Unificado de Desarrollo de Software, en donde se estableció el concepto inicial del sistema, creando un esquema general del mismo, se identificaron y describieron los requisitos de sistema bajo estudio mediante los llamados casos de uso, con el fin de elaborar un conjunto de modelos iníciales que permitieron capturar la semántica y comportamiento del sistema.

La fase de inicio es la que reviste más importancia en todo el proceso, dado que es en ella donde se efectuó el análisis del sistema propuesto, se describió el contexto del sistema, representado a través del modelo de dominio, lo que permitió identificar los riesgos críticos que pondrían en peligro el desarrollo del proyecto y proponer una arquitectura candidata factible. Los casos de uso identificados con el flujo de trabajo de los requisitos fueron estudiados en detalle con diagramas de colaboración en el flujo de análisis, para representar las interacciones entre los objetos. Además, se identificaron y describieron las clases de análisis, haciendo la refinación de los casos de uso, mediante el uso de diagramas de clases de análisis, y se construyó el diagrama de paquetes de análisis para encapsular las clases de análisis.

CAPÍTULO IV
FASE DE ELABORACIÓN

INTRODUCCIÓN

En el presente capítulo se contempla la fase de elaboración, en la cual se hace mayor énfasis en el análisis y diseño (*ver* **Figura 4.1**); en ella se refleja una perspectiva del sistema completo, se describen los elementos más importantes del mismo, y se tiene como objetivo principal construir una arquitectura sólida que sirva de base para edificar el sistema.

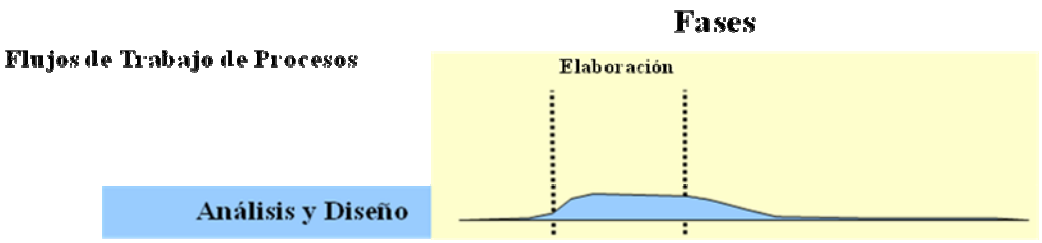

Figura 4.1. Flujos de trabajo de la fase de elaboración.
Fuente: Rumbaugh, 2004

La fase de elaboración tiene por objeto construir el núcleo central de la arquitectura, resolver los elementos de alto riesgo, definir los requerimientos y estimar los recursos necesarios. En esta fase, la arquitectura del sistema identifica las capacidades del sistema con las componentes de software, así como las relaciones entre estas componentes, se construye el núcleo central de la arquitectura incluyendo técnicas de diseño, se identifican los procesos, capas de software, paquetes, subsistemas, estableciendo sus responsabilidades, se efectúa la clarificación de las interfaces internas (entre componentes) y externas (con los actores), refinándolas e incluyendo parámetros y valores de retorno.

Los casos de uso seleccionados para desarrollarse en esta fase permiten definir la arquitectura del sistema, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar del problema y comienza la ejecución del plan de manejo de riesgos, según las prioridades definidas en él.

Al final de la fase, se determina la viabilidad de continuar el proyecto, dado que la mayor parte de los riesgos han sido mitigados, se obtiene una visión refinada, implementación iterativa del núcleo central de la aplicación, resolución de los riesgos más altos, identificación de nuevos requisitos y nuevos alcances, estimaciones más ajustadas.

4.1 FLUJO DE TRABAJO DE ANÁLISIS

En la fase de inicio se desarrolló el análisis de la arquitectura sólo hasta el punto de determinar que había una arquitectura del sistema factible. Ahora en esta fase se extiende el análisis hasta el punto de que pueda servir como base a la línea principal de la arquitectura ejecutable.

En la fase de análisis, se transforma el modelo de caso de usos en un modelo de análisis. El modelo de análisis de los requisitos permite lograr un método más concreto sobre los aspectos del sistema y una base más detallada de dichas exigencias obtenidas en el modelo de casos de uso para su mejor entendimiento.

Para el flujo de trabajo análisis se analizarán más a fondo los casos de uso identificados en los requisitos de la fase de inicio, planteando las clases de análisis de ellos, y estableciendo sus diagramas de colaboración y los

paquetes de análisis, lo cual permitirá precisar y fundamentar la línea base de la arquitectura del sistema.

4.1.1 Diagramas de Clase de Análisis

Para esta fase se presentan los diagramas de clase de análisis para los principales casos de uso del sistema. Las clases de análisis nos ayudan a representar la semántica y descripción más exacta de los casos de uso del sistema, ya que las mismas son un perfeccionamiento de estos casos de uso, basándose en las diferentes entidades o estereotipos de análisis que estos definen.

4.1.1.1 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Cliente

Para describir la forma en que se maneja la información relacionada con los clientes tenemos el diagrama de clases de análisis para el caso de uso Procesar Cliente. En este diagrama el administrador y usuario del sistema tienen acceso a las interfaces a través del gestor Procesar Cliente para insertar, modificar, eliminar o consultar los registros de los clientes.

Como se observa en la **Figura 4.2**, en el análisis del caso de uso Procesar Cliente destacan las clases interfaz denominadas Insertar Cliente, a través de la cual se registran nuevos clientes en el sistema; Modificar Cliente, que permitirá modificar los datos de cualquier cliente; Consultar Cliente, a través de la cual se podrán examinar los datos de cualquier cliente; y Eliminar Cliente, que permitirá eliminar el registro del cliente seleccionado.

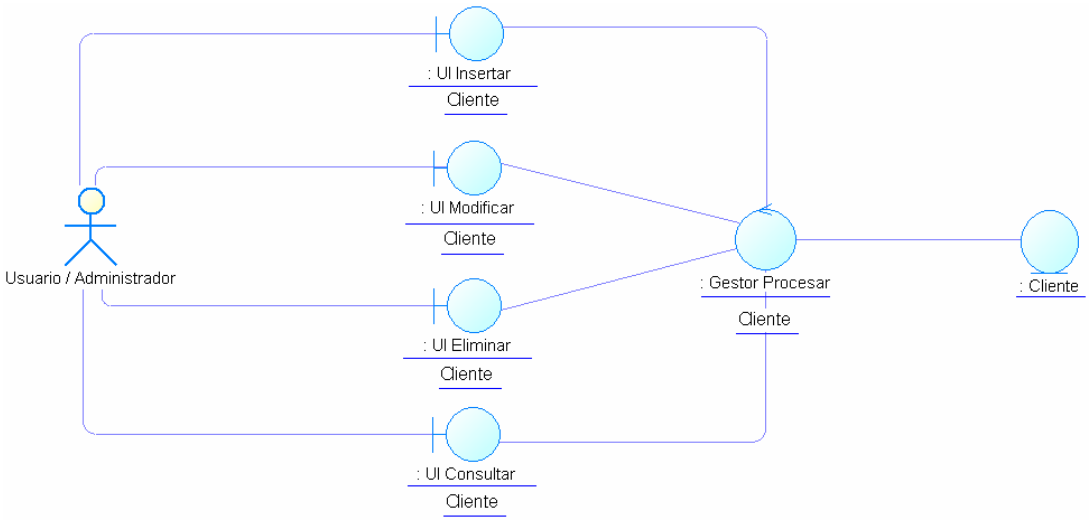

Figura 4.2. Diagrama de Clase de Análisis para el Caso de Uso Procesar Cliente

Fuente: Elaboración Propia

Además se pueden visualizar, la clase de control Gestor Procesar Cliente donde se procesará toda la información de los clientes; y la clase entidad Cliente donde se almacenan los datos de los clientes.

4.1.1.2 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Afiliado

En el diagrama de clase de análisis para el caso de uso Procesar Afiliado se observan (*ver* **Figura 4.3**) cuatro clases de interfaz denotadas de la siguiente manera, Interfaz Ingresar Afiliado, Interfaz Modificar Afiliado, Interfaz Eliminar Afiliado e Interfaz Consultar Afiliado; a través de las cuales se puede insertar en el sistema los datos de un nuevo afiliado, modificar los datos de un afiliado que ya ha sido registrado, eliminar del sistema cualquier afiliado que se desee, así como modificar los datos de éste, respectivamente.

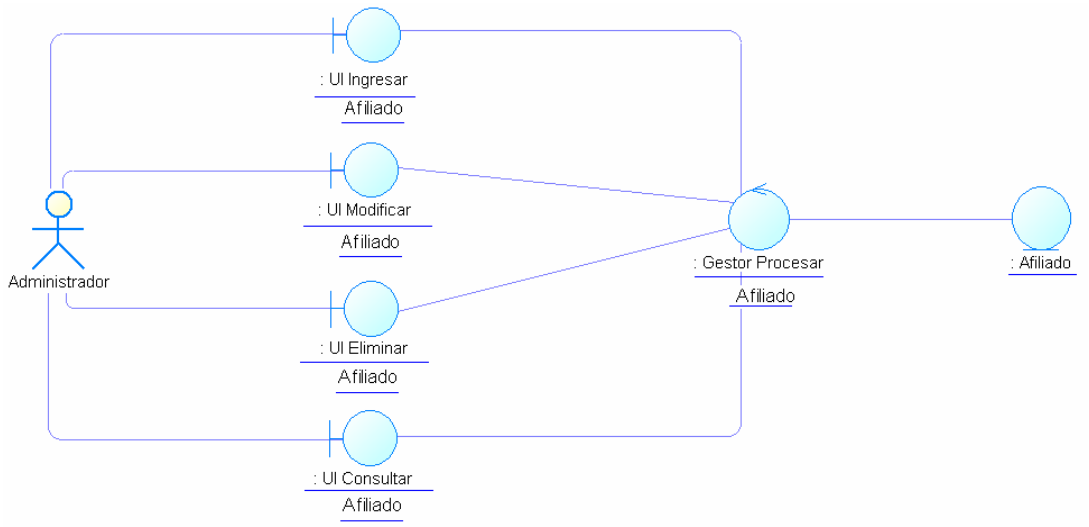

Figura 4.3. Diagrama de Clase de Análisis para el Caso de Uso Procesar Afiliado

Fuente: Elaboración Propia

De igual manera se puede ver la clase de control Gestor Procesar Afiliado, la cual interactúa con las clases de interfaz arriba mencionadas, permitiendo de ésta manera al administrador procesar la información correspondiente a los afiliados de la empresa.

Por último se tiene la clase de entidad Afiliado, ésta representa toda la información almacenada en el sistema de los afiliados.

4.1.1.3 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Monitorear Paquete

En el caso de uso Monitorear Paquete se presentan los actores con los privilegios correspondientes para este caso, la pantalla de interacción para monitoreo de los paquetes, así como también pone a la disposición de los actores un menú con las opciones: consultar y modificar estatus del paquete.

Para realizar el análisis del caso de uso Monitorear Paquete (*ver* **Figura 4.4)** se identificaron en primer lugar las clases de análisis: Interfaz Consultar Estatus del Paquete, que permitirá revisar el estatus en el que se encuentra un paquete en un momento dado, y la Interfaz Modificar Estatus del Paquete, donde se podrá actualizar el estatus del paquete, una vez que éste cambie. Se denota la clase de control Gestor Procesar Estatus del Paquete, que se encarga de manera general de los procesos relacionados con los estatus de los paquetes.

Además, se identifica la clase de entidad Envío, a la cual se accederá para visualizar o almacenar los estatus de los envíos registrados en el sistema.

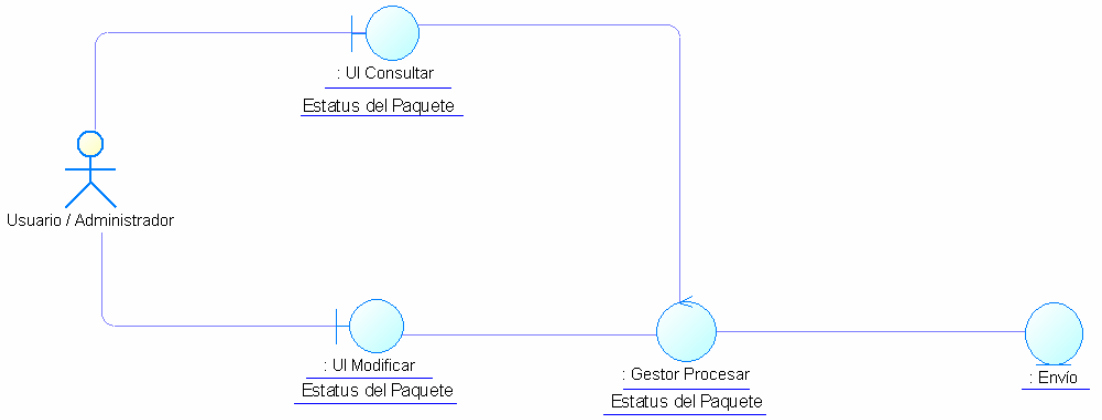

Figura 4.4. Diagrama de Clase de Análisis para el Caso de Uso Monitorear Paquete

Fuente: Elaboración Propia

4.1.1.4 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Ruta

El diagrama de clases de análisis para el caso de uso Procesar Ruta se describe la forma en que se maneja la información relacionadas con las rutas

en el sistema. En él, el administrador utiliza las interfaces a través del gestor Procesar Ruta para Ingresar, Eliminar, Modificar o Consultar los registros de las rutas utilizadas por las unidades de transporte de la empresa.

Como se puede observar en la **Figura 4.5**, el análisis del caso de uso Procesar Ruta destaca cuatro interfaces, Ingresar Ruta, Modificar Ruta, Eliminar Ruta y Consultar Ruta. La Interfaz Ingresar Ruta, permite añadir al sistema una nueva ruta. La interfaz Eliminar Ruta, permitirá eliminar cualquier ruta que haya sido registrada previamente. La interfaz Consultar Ruta, permitirá buscar en el sistema la información de cualquier ruta. La interfaz Modificar Ruta, permitirá modificar los datos de la ruta deseada.

De igual manera se visualiza la clase de control Gestor Procesar Ruta, donde se lleva a cabo el procesamiento de las rutas. Por último la clase entidad Ruta donde se almacenan los registros de las rutas procesadas.

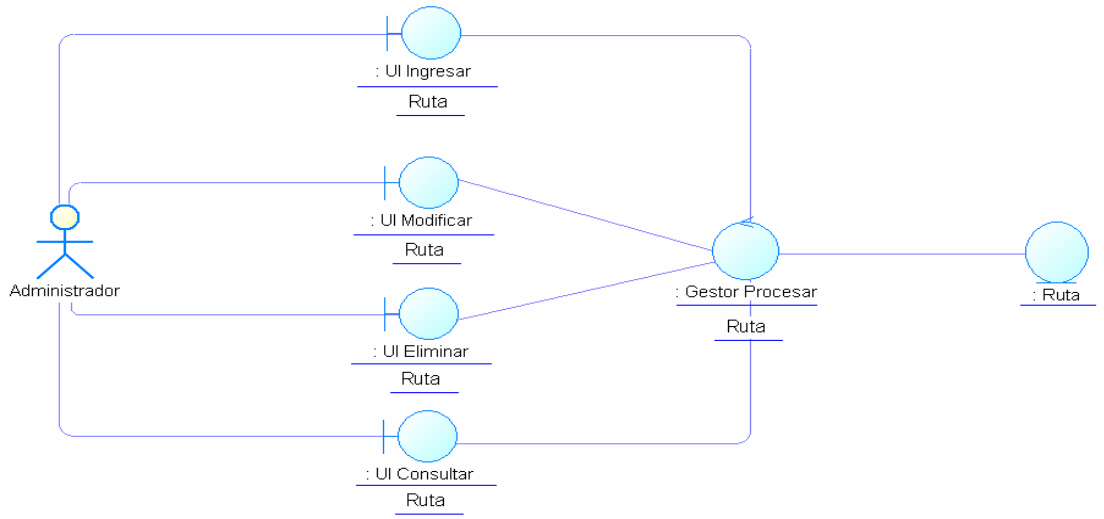

Figura 4.5. Diagrama de Clase de Análisis para el Caso de Uso Procesar Ruta

Fuente: Elaboración Propia

4.1.1.5 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Unidad de Transporte

Para el análisis del caso de uso Procesar Unidad de Transporte (*ver* **Figura 4.6**), se identificaron cuatro clases de interfaces con las cuales interactuará el administrador del sistema, la Interfaz Ingresar Unidad de Transporte, a través de la cuál ingresará los nuevos registros de unidades al sistema; la Interfaz Modificar Unidad de Transporte, que permitirá al actor modificar la información registrada de cada unidad de transporte; la Interfaz Eliminar Unidad de Transporte, por medio de ella se suprimirán del sistema los datos de la unidad de transporte deseada; y la Interfaz Consultar Unidad de Transporte, a través de ésta se podrán consultar las unidades de transporte y sus respectivos datos.

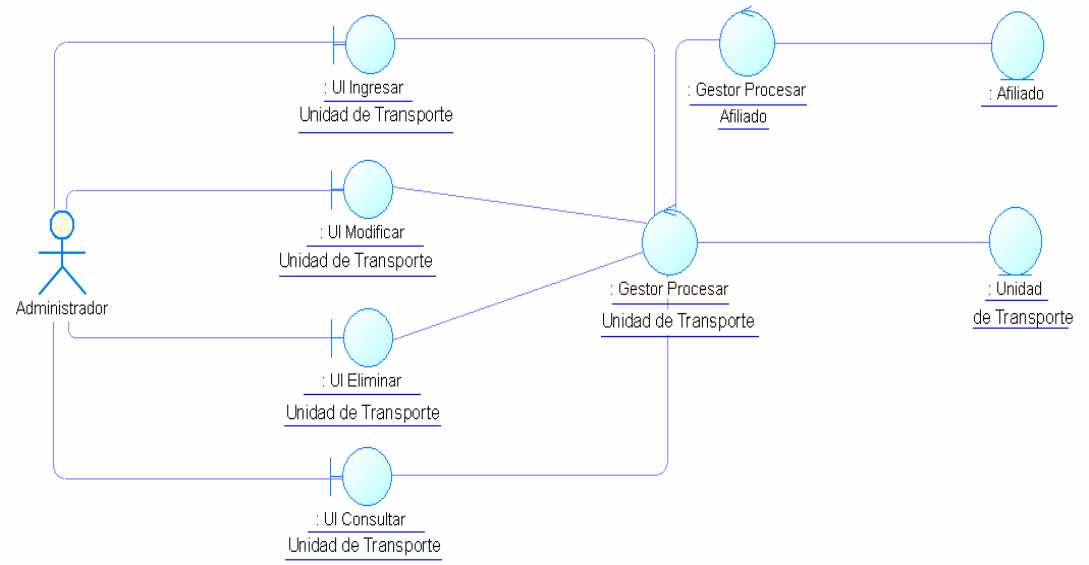

Figura 4.6. Diagrama de Clase de Análisis para el Caso de Uso Procesar Unidad de Transporte

Fuente: Elaboración Propia

Así mismo se definieron, las clases de control, Gestor Procesar Unidad de Transporte, la cuál será la encargada de ejecutar todos los procesos sobre la información de las unidades de transporte, y Gestor Procesar Afiliado, quien será la responsable de procesar la información de los afiliados asociados a las unidades.

Además se definen las clases de entidad, Afiliado, de la que se extrae la información relacionada con los dueños de dichas unidades; y Unidad de Transporte, en donde se almacenan los registros de las unidades.

4.1.1.6 Descripción del Diagrama de Clase de Análisis para el Caso de Uso Procesar Usuario

Para describir la forma en que se maneja la información relacionada con los usuarios tenemos el diagrama de clases de análisis para el caso de uso Procesar Usuario. En este diagrama el administrador del sistema tiene acceso a las interfaces a través del gestor Procesar Usuario para insertar, modificar, eliminar o consultar los registros de los usuarios.

Como se observa en la **Figura 4.7**, en el análisis del caso de uso Procesar Usuario destacan las clases interfaz denominadas Insertar Usuario, a través de la cual se registran nuevos usuarios del sistema; Modificar Usuario, que permitirá modificar los datos de los usuarios; Consultar Usuario, a través de la cual se podrán examinar los datos de cualquier usuario; y Eliminar Usuario, que permitirá eliminar el registro del usuario seleccionado.

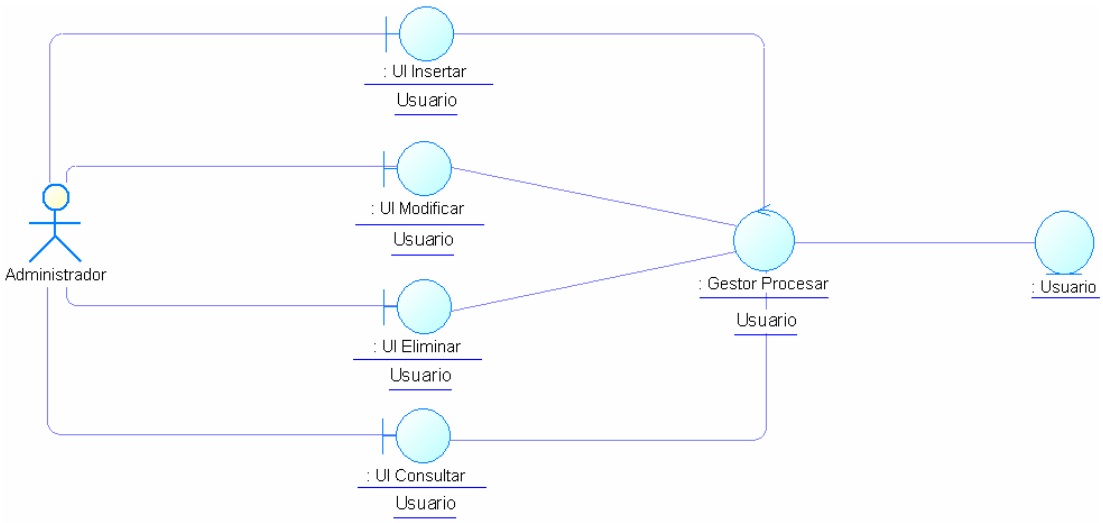

Figura 4.7. Diagrama de Clase de Análisis para el Caso de Uso Procesar Usuario

Fuente: Elaboración Propia

Además se pueden visualizar, la clase de control Gestor Procesar Usuario donde se procesará toda la información de los usuarios; y la clase entidad Usuario donde se almacenan los datos de los usuarios.

4.1.2 Diagramas de Colaboración

Los diagramas de clases de análisis permiten la apreciación de la estructura interna del mismo, la utilización de los diagramas de colaboración permitirán obtener la interacción con cada una de las entidades, gestores e interfaces del sistema.

En los diagramas de colaboración, la idea es ver los objetos en extenso, donde las interacciones muestran los flujos y los mensajes entre ellos, estos diagramas tienen como objetivo fundamental identificar los requisitos y responsabilidades sobre los objetos.

4.1.2.1 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Cliente

Para el caso de uso Procesar Cliente se identificaron dos actores, Usuario y Administrador del sistema, estos tendrán acceso a cuatro interfaces en las cuales manipularán los datos de los clientes registrados o que se deseen registrar en el sistema (*ver* **Figura 4.8**).

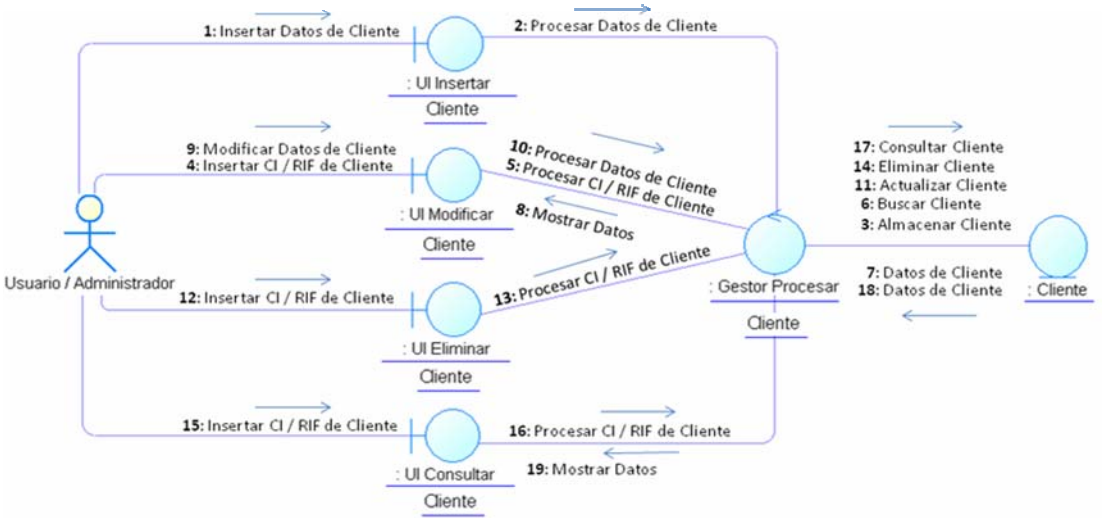

Figura 4.8. Diagrama de Colaboración para el Caso de Uso Procesar Cliente

Fuente: Elaboración Propia

En la interfaz Insertar Cliente, el actor llena los campos correspondientes para el registro del nuevo cliente (1), los cuales se procesan a través del Gestor Procesar Cliente (2), para luego ser almacenados en el sistema (3).

En la interfaz Modificar Cliente, inicialmente se ingresa la CI / RIF del Cliente (4), el Gestor Procesar Cliente se encarga de tramitarlo (5), para realizar la búsqueda en la base de datos (6), procesar el resultado de la

consulta (7), y luego visualizarlo (8), el actor modifica los campos deseados (9), para que finalmente el gestor se encargue de procesar la nueva información (10), y por último actualizar el registro del cliente en la base de datos (11).

En la interfaz Eliminar Cliente, primeramente el actor debe insertar la CI / RIF del Cliente (12), luego el Gestor Procesar Cliente procesa esta información (13), y subsiguientemente elimina el registro de la base de datos del sistema (14).

En la interfaz Consultar Cliente, de igual manera se debe ingresar la CI / RIF del Cliente (15), el Gestor Procesar Cliente la procesa (16), y realiza la consulta en la base de datos (17), para luego analizar la información extraída del sistema (18) y finalmente observarla (19).

4.1.2.2 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Afiliado

En el diagrama de colaboración del caso de uso Procesar Afiliado (ver **Figura 4.9**), se puede observar como el administrador del sistema a través de cuatro interfaces puede ingresar, modificar, eliminar y consultar, la información almacenada en el sistema de los afiliados.

Cuando el actor desea registrar un nuevo afiliado, despliega la interfaz Insertar Afiliado, en ella ingresa toda la información correspondiente al afiliado (1), toda esta información es procesada por el Gestor Procesar Afiliado (2), y luego son almacenados en la base de datos correspondiente (3).

En el caso de querer modificar la información de un afiliado, el actor activa la interfaz Modificar Afiliado, en la cual debe hacer la inserción de la CI del afiliado (4), para que el Gestor Procesar Afiliado la procese (5), y pueda solicitar los datos del afiliado a la base de datos (6); una vez realizada la consulta, el gestor examina el resultado de la misma (7), y luego lo muestra por pantalla (8), posteriormente el administrador modifica los datos del afiliado que desea (9), y luego el gestor analiza la nueva información (10), para finalmente actualizar en la base de datos el registro (11).

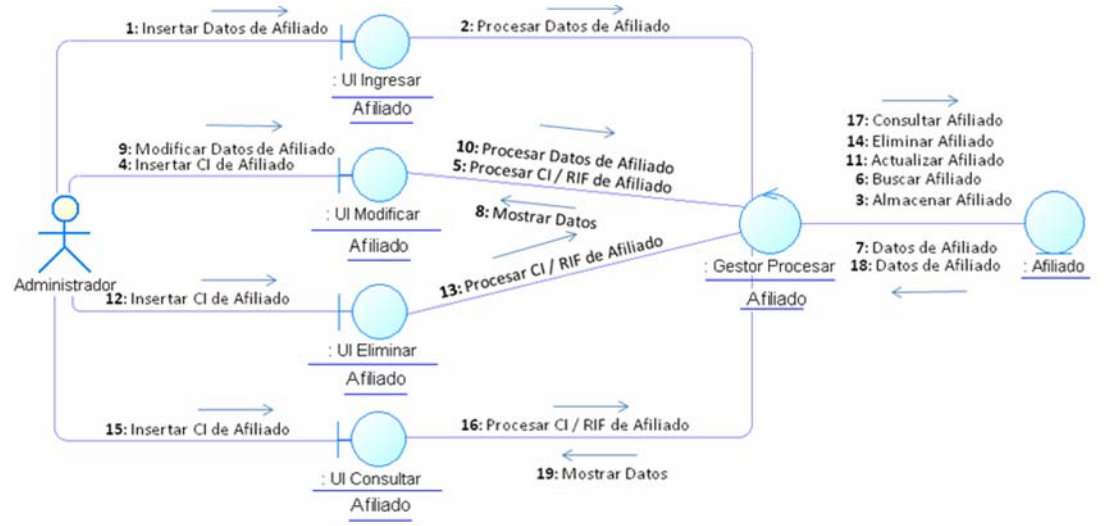

Figura 4.9. Diagrama de Colaboración para el Caso de Uso Procesar Afiliado

Fuente: Elaboración Propia

Por otra parte, si el administrador quiere eliminar a un afiliado del sistema, utiliza la interfaz Eliminar Afiliado; en primer lugar debe ingresar la CI del afiliado (12), para que éste, pueda ser procesado por el Gestor Procesar Afiliado (13), buscado en la base de datos, y finalmente eliminado de la misma (14).

Si en cambio, el actor desea consultar la información almacenada de un afiliado, activa la interfaz Consultar Afiliado, en la cual debe ingresar la CI del afiliado (15), para que el Gestor Procesar Afiliado pueda tramitarla (16), y solicitar la información a la base de datos (17); para finalmente poder procesarla (18) y visualizarla (19).

4.1.2.3 Descripción del Diagrama de Colaboración para el Caso de Uso Monitorear Paquete

En la **Figura 4.10,** se observa que el Usuario y Administrador pueden acceder a dos flujos de información, Consultar y Modificar Estatus del Paquete.

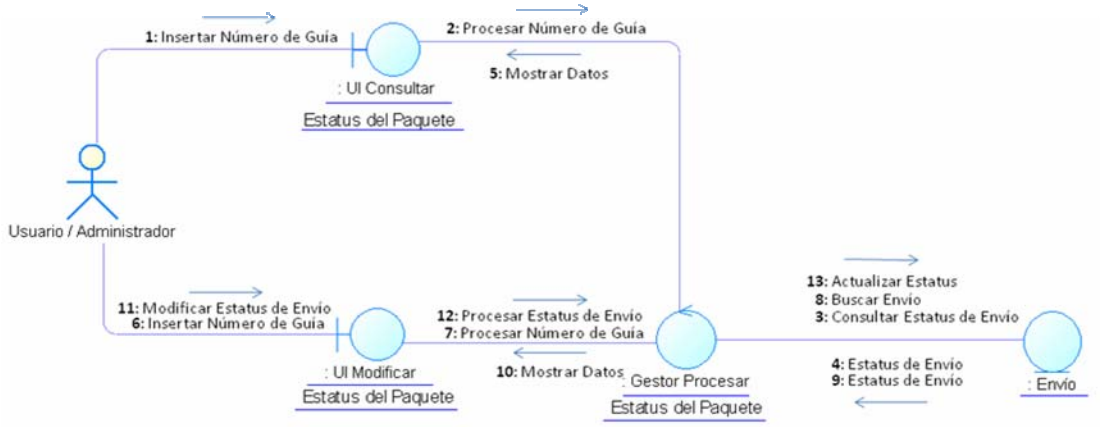

Figura 4.10. Diagrama de Colaboración para el Caso de Uso Monitorear Paquete

Fuente: Elaboración Propia

En el caso de ingresar a la interfaz Consultar Estatus del Paquete, el actor procede a insertar el Número de Guía del envío (1), éste luego es procesado por el Gestor Procesar Estatus del Paquete (2), posteriormente se ubica el registro en la base de datos con la información del paquete (3), para ser analizada por el gestor (4) y por ultimo ser visualizada por el actor (5).

Si por el contrario el actor activa la interfaz Modificar Estatus del Paquete, primero se inserta el Número de Guía (6), del cual el Gestor Procesar Estatus del Paquete se encarga de su procesamiento (7), luego se procede a buscar en la base de datos la información correspondiente (8), para que el gestor pueda analizarla (9), y posteriormente el usuario la verifique por pantalla (10) y modifique el estatus del paquete (11), para que el gestor gestione los nuevos datos (12); y finalmente actualizar el registro (13).

4.1.2.4 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Ruta

En la **Figura 4.11** se observa al actor identificado como Administrador, el cual tiene acceso a cuatro flujos básicos para el caso de uso Procesar Ruta, los cuales están identificados como: Ingresar Ruta, Modificar Ruta, Eliminar Ruta y Consultar Ruta.

Cuando el actor necesita insertar una nueva ruta, éste despliega la interfaz Ingresar Ruta, en la cual se llenan los campos correspondientes con la información de la misma (1), luego esta información es procesada por el Gestor Procesar Ruta (2), y posteriormente se almacena en la base de datos del sistema (3).

Al momento que el actor deseé modificar una ruta, éste activa la interfaz de Modificar Ruta, una vez desplegada, el administrador debe insertar el Id de Ruta (4), para que el Gestor de Procesar Ruta lo procese (5) y a través de él realice la consulta a la base de datos (6), la cual devuelve la información de la ruta (7), para ser visualizada (8), a continuación el actor modifica la información deseada (9), para que el gestor la tramite (10), y finalmente actualizar el registro (11).

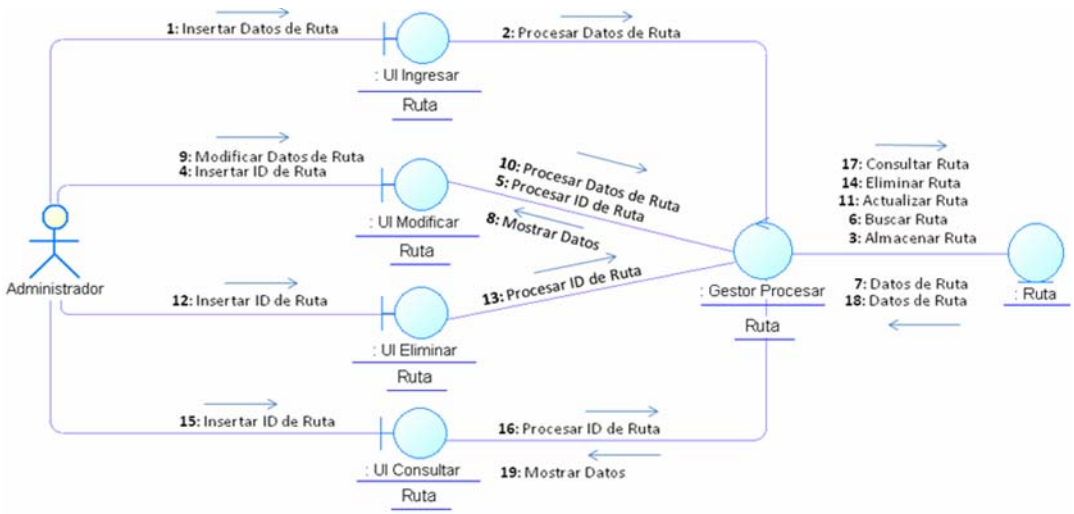

Figura 4.11. Diagrama de Colaboración para el Caso de Uso Procesar Ruta
Fuente: Elaboración Propia

Para eliminar una ruta, se selecciona la interfaz Eliminar Ruta, en ella se ingresa el Id de Ruta (12), luego el Gestor Procesar Ruta se encarga de procesar dicho Id (13), para finalmente eliminar del sistema la ruta (14).

Al momento de realizar la consulta de una ruta, el actor activa la interfaz Consultar Ruta, en la cual inserta el Id de la Ruta (15), para su posterior procesamiento por parte Gestor Procesar Ruta (16), éste posteriormente realiza la consulta a la base de datos (17), para analizar los datos de la ruta (18) y por último mostrarla por pantalla.

4.1.2.5 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Unidad de Transporte

En el caso de uso Procesar Unidad de Transporte se pueden identificar cuatro flujos de información, a los cuales el actor identificado puede acceder activando la interfaz correspondiente a cada caso (*ver* **Figura 4.12**).

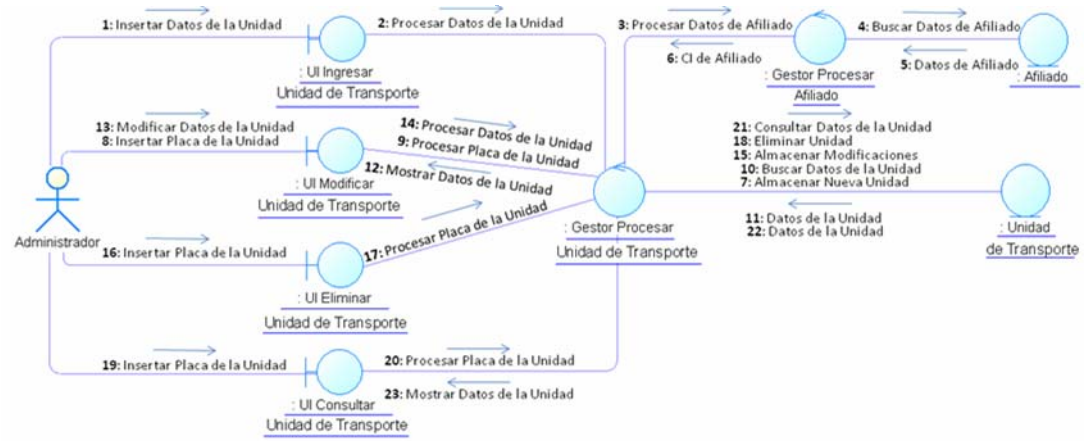

Figura 4.12. Diagrama de Colaboración para el Caso de Uso Procesar Unidad de Transporte

Fuente: Elaboración Propia

Para registrar una nueva unidad, el actor debe desplegar la interfaz Ingresar Unidad de Transporte, a continuación, ingresa todos los datos referentes a la unidad (1), para su posterior procesamiento por parte del Gestor Procesar Unidad de Transporte (2), éste a su vez, se encarga de solicitar a través del Gestor Procesar Afiliado los datos del socio (3), para que éste los solicite al sistema (4), quién a su vez devuelve los datos del afiliado (5), para que el Gestor procesar Afiliado envié la CI del afiliado al Gestor Procesar Unidad de Transporte que será registrado como propietario de la unidad (6), y finalmente se almacena toda esta información en la base de datos (7).

En el caso de necesitar modificar el color de una unidad, se ingresa la placa de la unidad de transporte a través de la interfaz Modificar Unidad de Transporte (8), y el Gestor Procesar Unidad de Transporte se encarga de procesar esta información (9), para luego realizar la consulta a la base de datos (10), posterior a esto, el gestor analiza el resultado de dicha consulta (11), luego esta información es visualizada por el usuario (12), y a continuación el actor procede a modificar el campo anteriormente nombrado

(13), el cual es procesado por el gestor (14), para que finalmente sea actualizado en el registro (15).

Cuando el administrador desea eliminar el registro de una unidad de transporte debe activar la interfaz Eliminar Unidad de Transporte, una vez que ésta se encuentra desplegada, el actor ingresa la placa que identifica a la unidad (16), el Gestor Procesar Unidad de Transporte se encarga a través de este dato (17) de eliminar de la base de datos la información correspondiente (18).

Para consultar los datos de una unidad de transporte, se utiliza la interfaz Consultar Unidad de Transporte, en la cual el administrador debe ingresar la placa de la unidad (19), para que el Gestor Procesar Unidad de Transporte se encargue de procesarla (20), y luego solicite a la base de datos la información correspondiente (21), para que esta pueda ser examinada por el gestor (22) y luego por el usuario (23).

4.1.2.6 Descripción del Diagrama de Colaboración para el Caso de Uso Procesar Usuario

Para el caso de uso Procesar Usuario se identificó un actor, Administrador del sistema, este tendrá acceso a cuatro interfaces en las cuales manipulará los datos de los usuarios registrados o que se desee registrar en el sistema (*ver* **Figura 4.13**).

En la interfaz Insertar Usuario, el actor llena los campos correspondientes para el registro del nuevo usuario (1), los cuales se procesan a través del Gestor Procesar Usuario (2), para luego ser almacenados en el sistema (3).

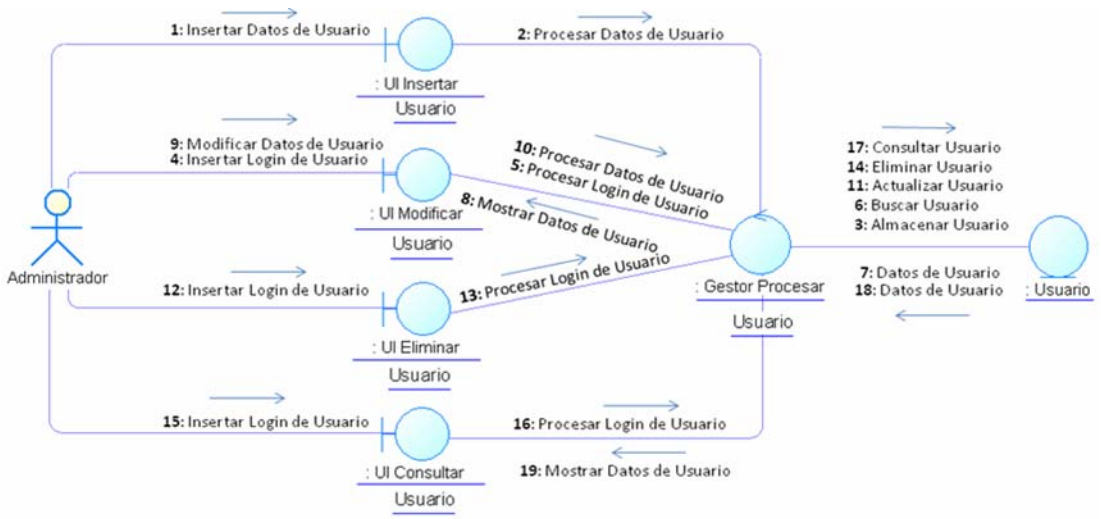

Figura 4.13. Diagrama de Colaboración para el Caso de Uso Procesar Usuario

Fuente: Elaboración Propia

En la interfaz Modificar Usuario, inicialmente se ingresa el login de usuario (4), el Gestor Procesar Usuario se encarga de procesarlo (5), para realizar la consulta al sistema (6), el cual devuelve todos los datos asociados al login (7), para que el actor los examine (8) y modifique los campos deseados (9), luego el gestor procesa esta nueva información (10), para finalmente actualizar el registro del usuario en la base de datos (11).

En la interfaz Eliminar Usuario, primeramente el actor debe insertar el login de usuario (12), luego el Gestor Procesar Usuario procesa esta información (13), y por último se elimina el registro de la base de datos (14).

En la interfaz Consultar Usuario, de igual manera se debe ingresar el login del usuario (15), el Gestor Procesar Usuario lo procesa (16), y realiza la consulta en la base de datos (17), para examinar la información extraída del sistema (18) y finalmente visualizarla (19).

4.1.3 Diagrama de Clase de Diseño

Una clase es un artefacto de modelado que describe un conjunto de objetos que comparten los mismos atributos (conocimiento), operaciones (responsabilidad), relaciones (entrelazamiento), y semántica (relevancia).

Para modelar los aspectos estáticos de un sistema se hace uso del diagrama de clases, el cual es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargarán del funcionamiento y la relación entre uno y otro.

El diagrama de clases es el principal diagrama de diseño para un sistema. Los diagramas de clases de diseño muestran la funcionalidad del sistema mediante clases relacionadas entre sí por medio de líneas, haciendo la especificación de las instancias que participan en la relación.

Las clases de diseño definen los atributos y operaciones con que debe contar cada clase de análisis para llevar a cabo sus responsabilidades, así como las relaciones existentes entre ellas. Los atributos y operaciones se encuentran en su mayoría inmersos dentro de los diagramas y artefactos obtenidos en los diferentes flujos de trabajo.

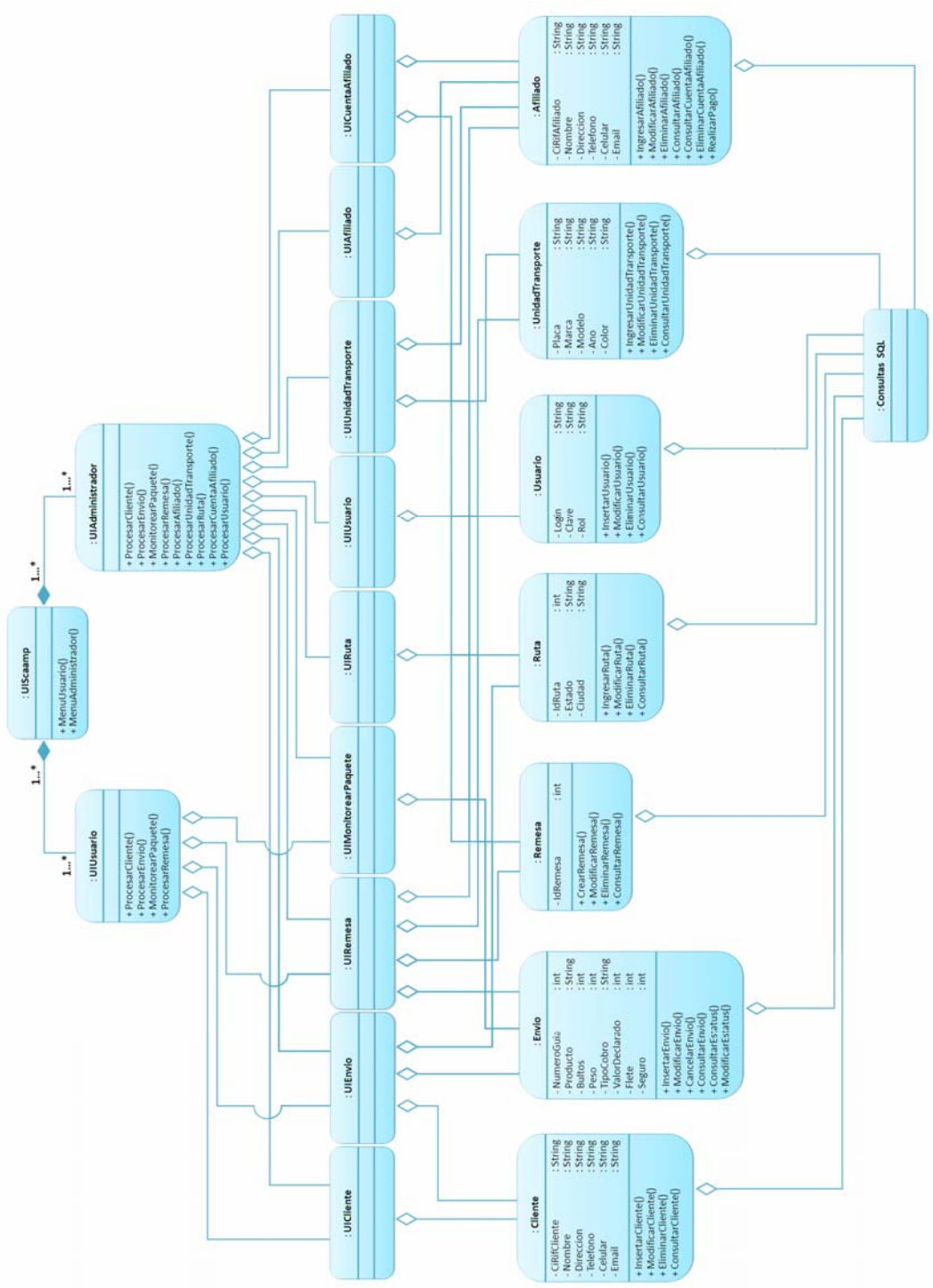

Figura 4.14. Diagrama de Clase de Diseño del Sistema

Fuente: Elaboración Propia

Una vez obtenido el resultado del flujo de análisis en la fase de inicio, en la **Figura 4.14** se puede observar el nivel de abstracción del sistema representado a través del Diagrama de clases de Diseño del Sistema SCAAMP, definiendo las clases presentes en el sistema.

La clase UIScaamp representa toda la aplicación como un conjunto, las clases UIUsuario y UIAdministrador son agregadas a la clase UIScaamp, y constituyen las interfaces de usuario a las que podrán acceder los actores para su interacción con el sistema.

Las clases UICliente, UIEnvio, UIRemesa y UIMonitorearPaquete, son agregadas a la clase UIUsuario, éstas representan ventanas de interfaz de usuario que se usan para el procesamiento de clientes, envíos y remesas, y para el monitoreo de paquetes, respectivamente.

A la clase UIAdministrador se agregan las interfaces de usuarios definidas como: clase UICliente, por medio de la cual se manipulará la información de los clientes; clase UIEnvio, que permitirá gestionar los envíos; clase UIRemesa, para administrar la relación de guías transportadas por cada unidad de transporte; clase UIMonitorearPaquete, para consultar y modificar el estatus de los paquetes enviados; clase UIAfiliado, a través de la cual se procesan los datos de los socios de la empresa; clase UIUsuario, para el procesamiento de los usuarios del sistema; clase UIUnidadTransporte, por medio de ella se manipularán los datos de las unidades de transporte que poseen los afiliados; clase UIRuta, esta permitirá procesar las rutas cubiertas por las unidades registradas; y clase UICuentaAfiliado, a través de la cual se llevará el control administrativo de cada afiliado de la empresa.

4.1.4 Diagrama de Secuencia

Los Diagramas de Secuencia son conocidos como Diagramas de Interacción dado que muestran la interacción que se establece en un conjunto de objetos y sus relaciones, además de los mensajes que estos se envían. Es decir, tratan la vista dinámica de un sistema.

Estos diagramas enfatizan como se realiza un caso de uso en términos de las clases del diseño, identifica las instancias de los actores, las interacciones entre los objetos del diseño y los mensajes que se envían y reciben estos objetos. Aquí lo importante es encontrar secuencias de interacciones ordenadas en el tiempo.

A continuación desde la **Figura 4.15** hasta la **Figura 4.17** se muestran los Diagramas de Secuencia de los Casos de Uso más importantes del sistema SCAAMP.

En la **Figura 4.15** se observa el diagrama de secuencia para el caso de uso Procesar Envío, en el cual los actores solicitan alguna de las operaciones disponibles en relación a los envíos como lo son, Insertar, Cancelar, Eliminar o Modificar, para esto el actor envía una solicitud a una interfaz principal la cual envía los datos a la interfaz adecuada para el actor, encargada de transmitir los datos a la clase correspondiente a la gestión de envíos según el actor y finalmente transmitir a la clases de conexión y consultas para realizar la operación seleccionada en la base de datos.

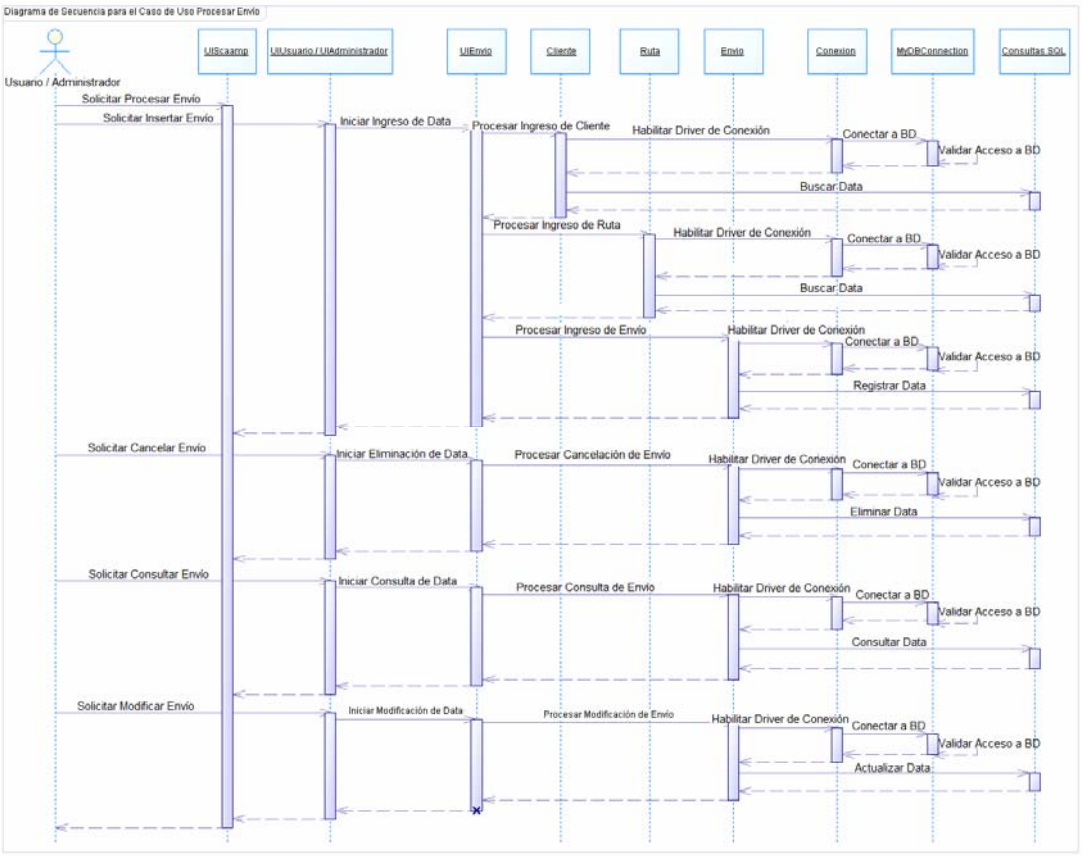

Figura 4.15. Diagrama de Secuencia para el Caso de Uso Procesar Envío

Fuente: Elaboración Propia

La **Figura 4.16** muestra el diagrama de secuencia para el caso de uso Procesar Remesa, en él observamos como los actores que intervienen solicitan a una interfaz principal las diferentes operaciones que pueden realizar, esta interfaz se encarga de transmitir estos datos a otra interfaz correspondiente al actor mediante la cual se envían los datos a la clase creada para las gestiones de remesas, en donde se realiza el proceso de transmisión hacia las clases encargadas de las consultas y de la conexión hacia la base de datos.

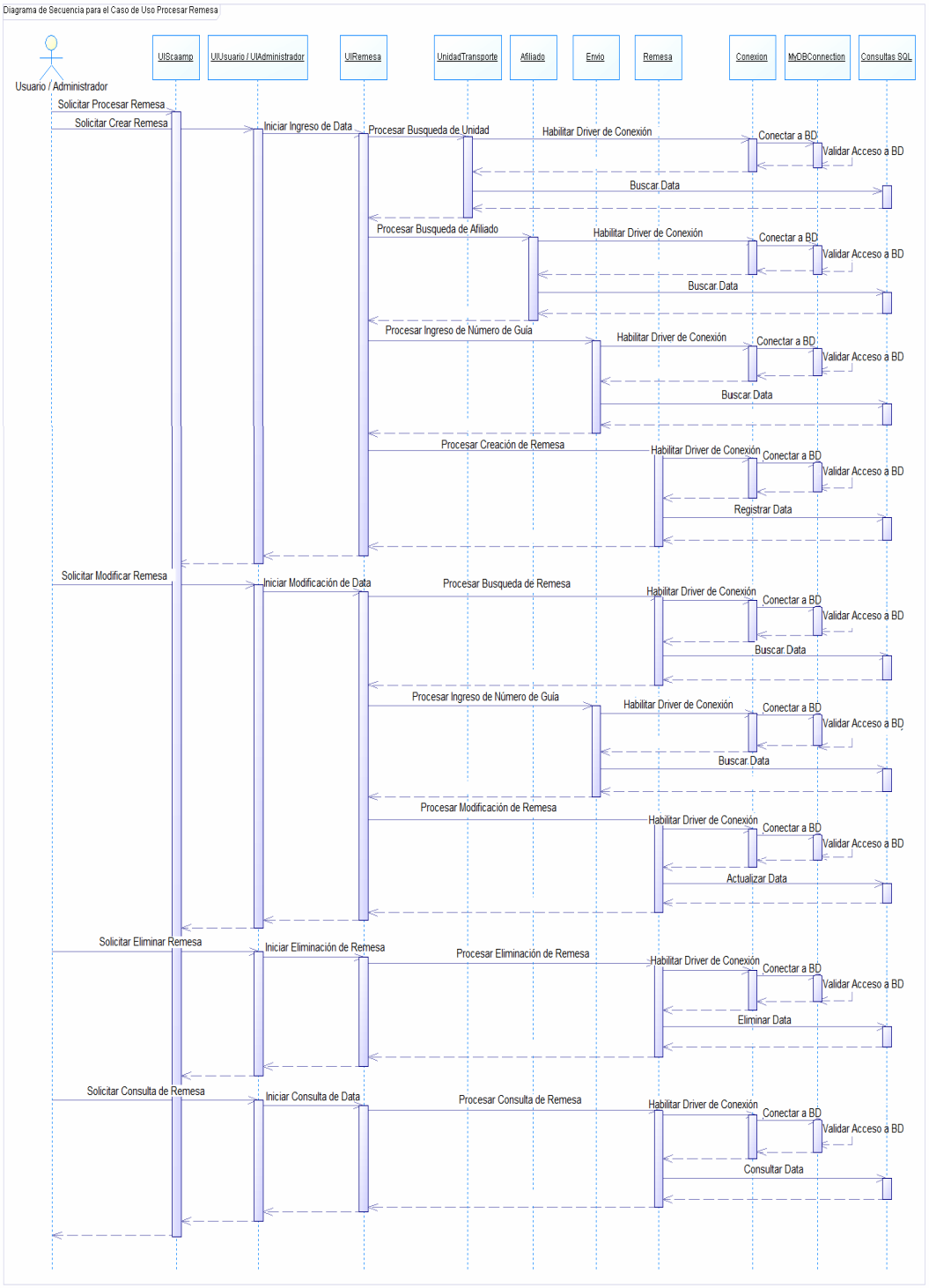

Figura 4.16. Diagrama de Secuencia para el Caso de Uso Procesar Remesa

Fuente: Elaboración Propia

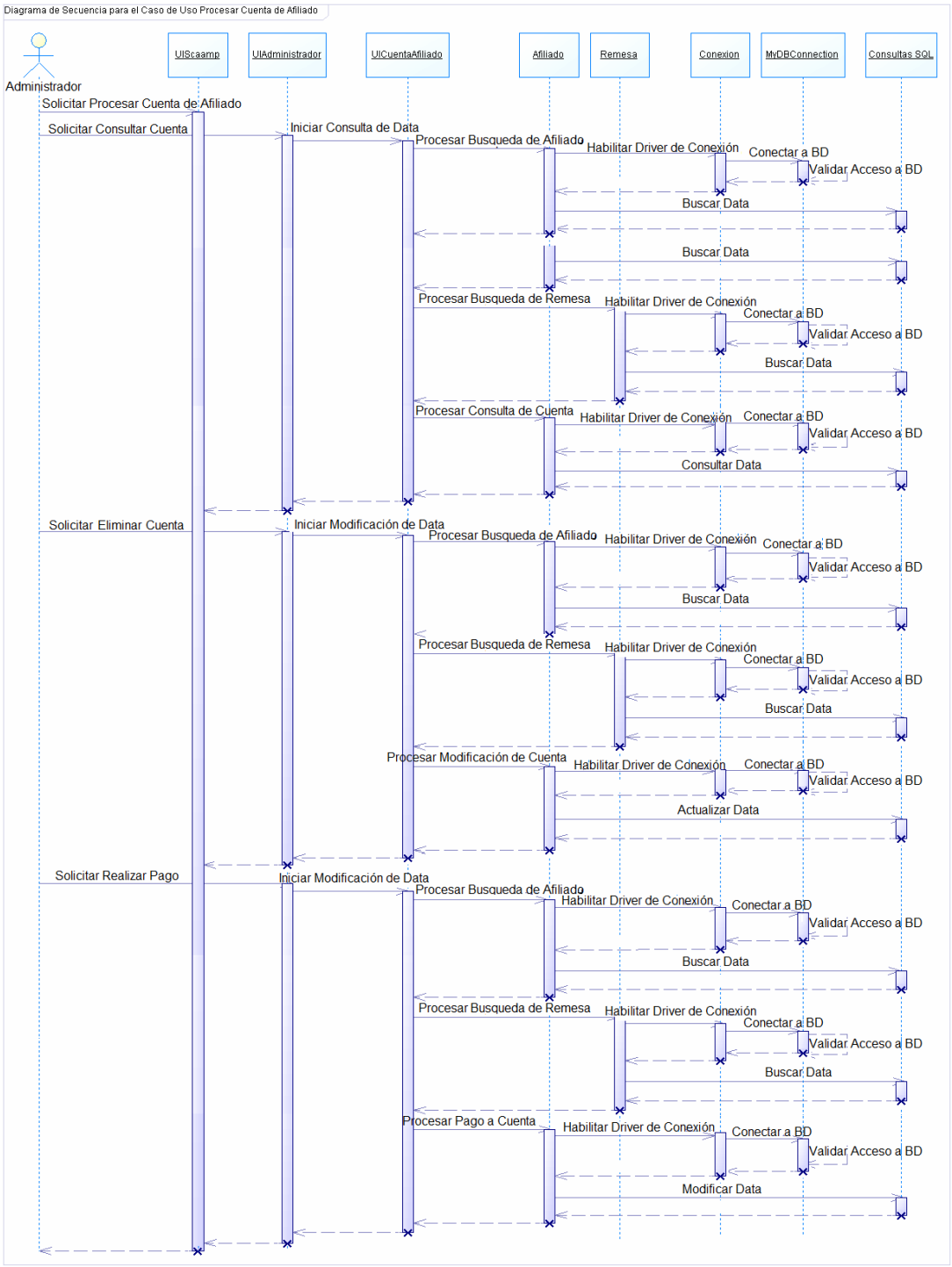

Figura 4.17. Diagrama de Secuencia para el Caso de Uso Procesar Cuenta de Afiliado

Fuente: Elaboración Propia

En el diagrama de secuencia para el caso de uso Procesar Cuenta de Afiliado (*ver* **Figura 4.17**), se puede observar como el actor solicita alguna de las operaciones disponibles en relación al procesamiento de las cuentas de los afiliados como son, Consultar, Eliminar o Realizar Pago, para esto el usuario envía una solicitud a una interfaz principal la cual envía los datos a la interfaz adecuada para el actor, esta se encarga de transmitir los datos a la clase correspondiente para la gestión de cuentas y finalmente transmitir a la clases de conexión y consultas para realizar la operación seleccionada en la base de datos.

4.1.5 Diagrama de Capas

El uso de la arquitectura de capas es aplicable a muchos tipos de sistemas. Este patrón define como organizar el modelo de diseño en capas, lo cual quiere decir que los componentes de una capa solo pueden hacer referencia a componentes en capas inmediatamente inferiores. Este patrón es importante porque simplifica la comprensión y la organización del desarrollo de sistemas complejos, reduciendo las dependencias de forma que las capas más bajas no son conscientes de ningún detalle o interfaz de las superiores. Además, ayuda a identificar que puede reutilizarse, y proporciona una estructura que facilita la toma de decisiones sobre que partes comprar y que partes construir.

La capa general de aplicación contiene los subsistemas que no son específicos de una sola aplicación, sino que pueden ser reutilizados por muchas aplicaciones diferentes dentro del mismo dominio o negocio. La arquitectura de las dos capas inferiores puede establecerse sin considerar los casos de uso de que no son dependientes del negocio. La arquitectura de

las dos capas superiores se crea a partir de los casos de uso significativos para la arquitectura (estas capas son dependientes del negocio).

Una capa es un conjunto de subsistemas que comparten el mismo grado de generalidad y de volatilidad en las interfaces: las capas inferiores son de aplicación general a varias aplicaciones y deben poseer interfaces más estables, mientras que las capas más altas son más dependientes de la aplicación y pueden tener interfaces menos estables.

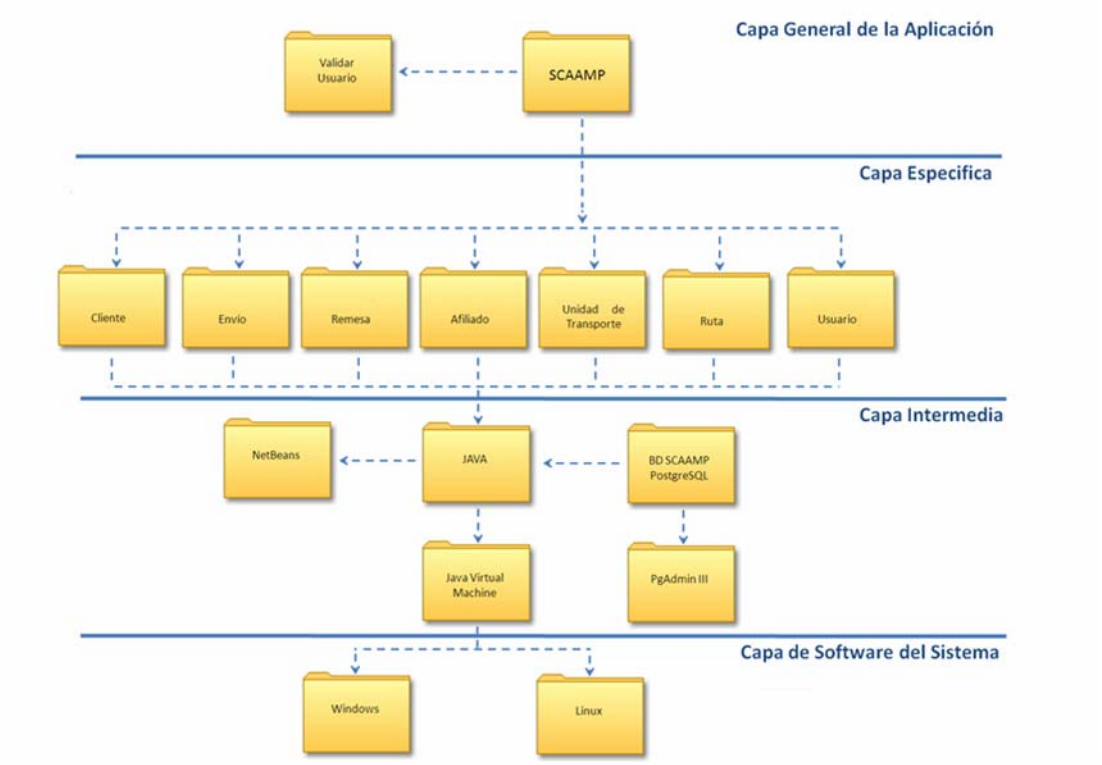

Figura 4.18. Diagrama de Capas del Sistema SCAAMP

Fuente: Elaboración Propia

La **Figura 4.18** muestra un diagrama de capas donde se describe el sistema informático en términos de varios niveles desde el mas externo (donde se encuentra el usuario) hasta el más interno (donde se encuentra el

hardware). La información fluye entre las diferentes capas a nivel real y a nivel lógico.

4.2 FLUJO DE TRABAJO DE DISEÑO

En el diseño se modelará el sistema y se encontrará la forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales y otras restricciones, que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, esto es, el modelo de análisis.

El modelo de análisis proporciona una comprensión detallada de los requisitos. Y lo que es más importante, impone una estructura del sistema, esta se debe conservar lo más fielmente posible cuando se dé forma al sistema.

4.2.1 Diseño de la Base de Datos

El diseño de la base de datos es un elemento fundamental durante el desarrollo de cualquier proyecto de software, ya que de esta manera se pueden evaluar el esquema de almacenamiento de la información antes de implementarla, esta etapa es muy importante ya que ayuda a garantizar en todo momento la integridad de los datos y evitar errores tales como lo son la conexión errada de las tablas y la duplicidad de la información.

Cada una de las tablas identificadas y que se describen están vinculadas entre sí por alguno de sus atributos. A esta relación se le llama dependencia funcional.

4.2.1.1 Modelo De Datos

El modelo de datos define la estructura de la base de datos. En la **Figura 4.19** pueden verse los atributos de cada entidad así como su relación con otras entidades.

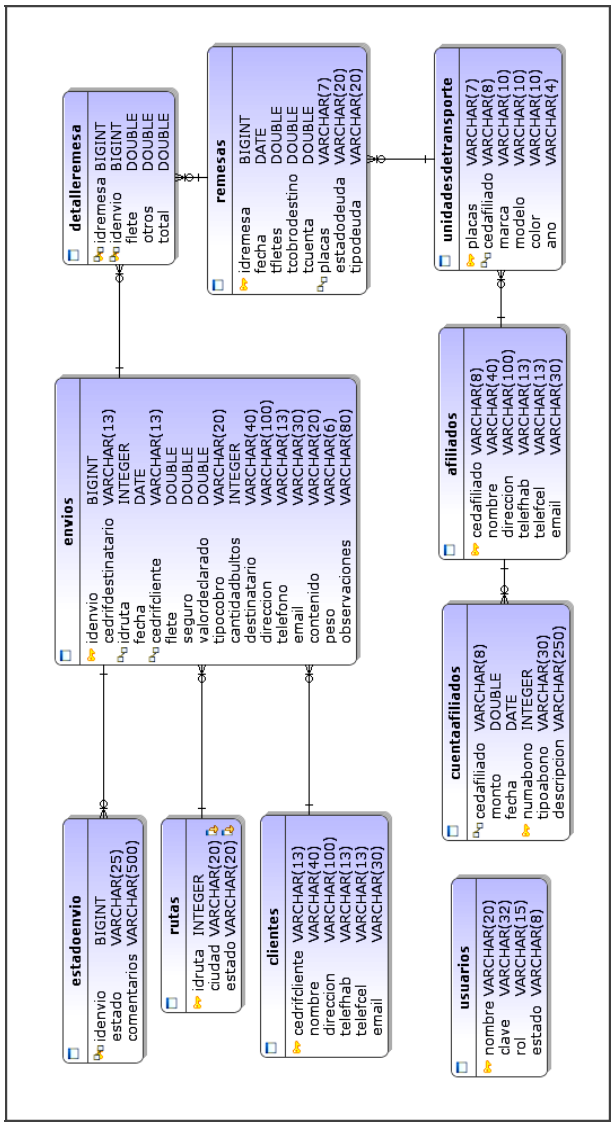

Figura 4.19. Relación de las Tablas de la Base de Datos

Fuente: Elaboración Propia

4.2.1.2 Estructura de la Base de Datos

Partiendo del modelo de datos y una vez normalizada las entidades, se implementó el diseño de la base de datos del sistema, a través del sistema gestor de base de datos PostgreSQL. El conjunto de tablas que conforman la base de datos serán mostradas y descritas a continuación.

- **Tabla Clientes**

Registra el detalle de los datos de los diferentes clientes (*ver* **Figura 4.20**).

Figura 4.20. Estructura de la Tabla Cliente

Fuente: Elaboración Propia

- **Tabla Estado Envío**

Registra la información acerca del estado de los diferentes envíos procesados (*ver* **Figura 4.21**).

Figura 4.21. Estructura de la Tabla Estado Envío

Fuente: Elaboración Propia

- **Tabla Envíos**

Registra la información de los diferentes envíos procesados (*ver* **Figura 4.22**).

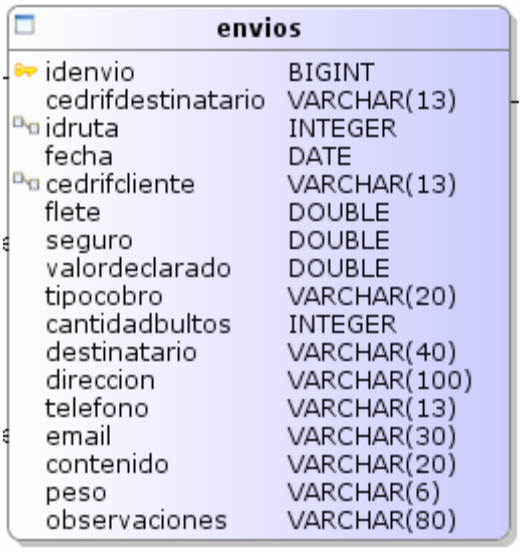

Figura 4.22. Estructura de la Tabla Envío

Fuente: Elaboración Propia

- **Tabla Afiliados**

Registra detalladamente toda la información de los diferentes afiliados (*ver* **Figura 4.23**).

Figura 4.23. Estructura de la Tabla Afiliado

Fuente: Elaboración Propia

- **Tabla Unidades de Transporte**

Registra los datos de las diferentes unidades de transporte que puede poseer cada afiliado de la empresa (*ver* **Figura 4.24**).

Figura 4.24. Estructura de la Tabla Unidad de Transporte

Fuente: Elaboración Propia

- **Tabla Rutas**

Registra el detalle de las diferentes rutas que pueden cubrir las unidades de transporte (*ver* **Figura 4.25**).

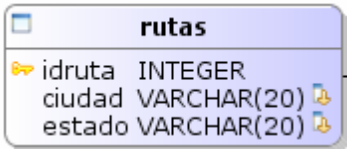

Figura 4.25. Estructura de la Tabla Rutas

Fuente: Elaboración Propia

- **Tabla Remesas**

Registra la información detallada de las diferentes remesas que posee cada unidad de transporte (*ver* **Figura 4.26**).

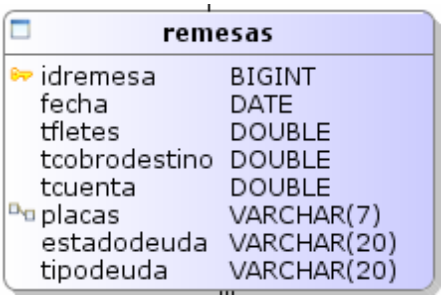

Figura 4.26. Estructura de la Tabla Remesas

Fuente: Elaboración Propia

- **Tabla Detalle Remesa**

Registra detalles de los diferentes paquetes que pertenecen a cada remesa (*ver* **Figura 4.27**).

Figura 4.27. Estructura de la Tabla Detalle Remesa

Fuente: Elaboración Propia

- **Tabla Usuarios**

Registra el detalle de los datos de los diferentes usuarios del sistema (*ver* **Figura 4.28**).

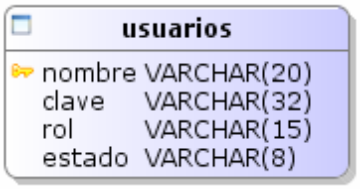

Figura 4.28. Estructura de la Tabla Usuarios

Fuente: Elaboración Propia

- **Tabla Cuenta Afiliados**

Registra los detalles de los abonos o pagos realizados a las cuentas de los afiliados (*ver* **Figura 4.29**).

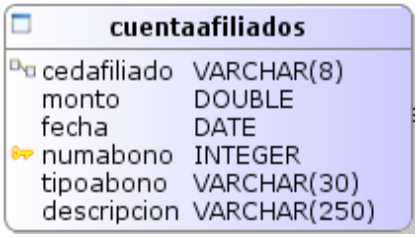

Figura 4.29. Estructura de la Tabla Abonos

Fuente: Elaboración Propia

4.2.2 Diseño de Interfaz

Uno de los aspectos más importantes del proceso de desarrollo de software es el diseño de la interfaz de usuario, su importancia radica en que, por medio de ésta se hace posible la interacción entre el usuario y el sistema.

Se deben considerar aspectos asociados a facilitar dicha interacción, tales como: realizar un diseño de pantallas que cumplan con las siguientes normas; deben ser claras, sencillas, no se deben recargar de información, las pantallas deben ser consistentes, se deben usar los colores adecuados y con una buena selección de menús de usuarios.

Las interfaces del sistema SCAAMP fueron desarrolladas usando como herramienta para el diseño la librería “Swing”, la cual puede ser manipulada directamente con el uso del lenguaje “Java” a través del IDE “NetBeans”, a continuación presentaremos las interfaces de mayor importancia dentro del entorno del sistema.

4.2.1.1 Ventana Principal

Al iniciar el sistema se muestra la ventana principal (ver **Figura 4.30**), en ella los usuarios tendrán que iniciar sesión para poder acceder a todas las funciones del sistema. De lo contrario sólo podrán cerrar la aplicación.

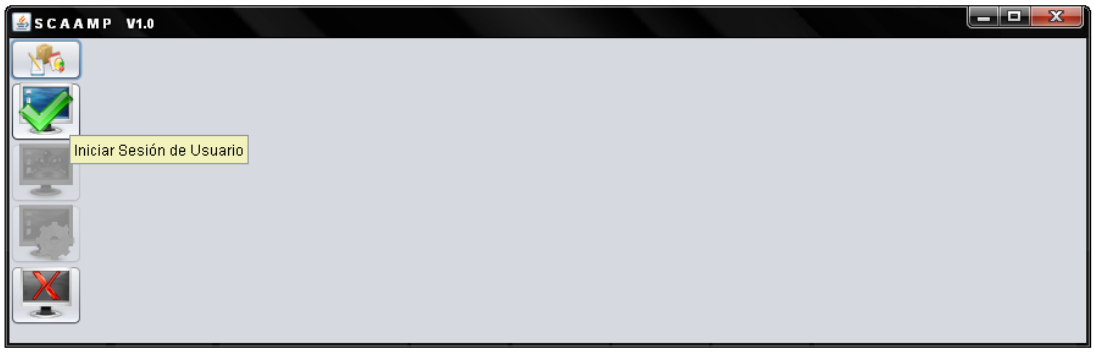

Figura 4.30. Interfaz Principal del Sistema SCAAMP
Fuente: Elaboración Propia

4.2.1.2 Acceso al sistema

Una vez que el usuario presiona en botón “Iniciar Sesión”, se despliega una ventana de validación de usuario (*ver* **Figura 4.31**) en la cual se le permite ingresar su correspondiente nombre de usuario y contraseña, si estos datos corresponden a un usuario existente en el sistema se permite su acceso, con las restricciones definidas para el mismo.

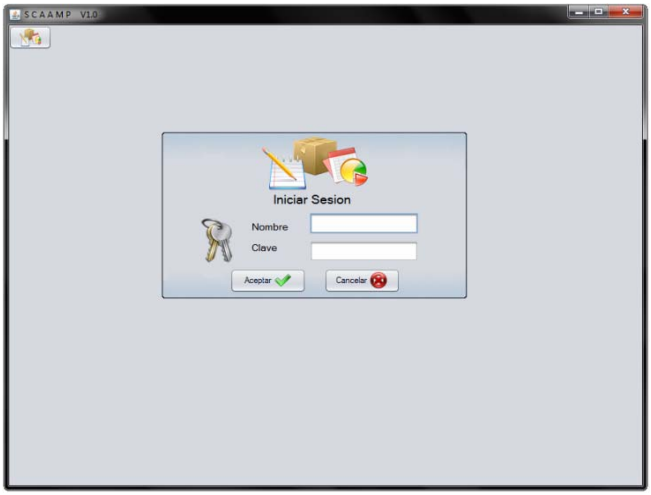

Figura 4.31. Ventana de Validación de Usuario

Fuente: Elaboración Propia

4.2.1.3 Menú principal

El menú principal (*ver* **Figura 4.32**) proporciona una interfaz de fácil acceso a cada una de las funcionalidades del sistema.

Figura 4.32. Menú Principal

Fuente: Elaboración Propia

- **Botón "Inicio"**

A través de él, el usuario podrá acceder a las funciones básicas del sistema, como lo son, Cerrar Sesión, Cambiar Usuario, Salir del Sistema (*ver* **Figura 4.33**).

Figura 4.33. Botón Inicio

Fuente: Elaboración Propia

- **Menú "Envíos"**

Este menú (*ver* **Figura 4.34**) provee acceso a las distintas operaciones referidas al procesamiento de envíos.

Figura 4.34. Menú Envíos

Fuente: Elaboración Propia

- **Ventana Nuevo Envío**

La presente ventana (*ver* **Figura 4.35**) permite a los usuarios gestionar un nuevo envío.

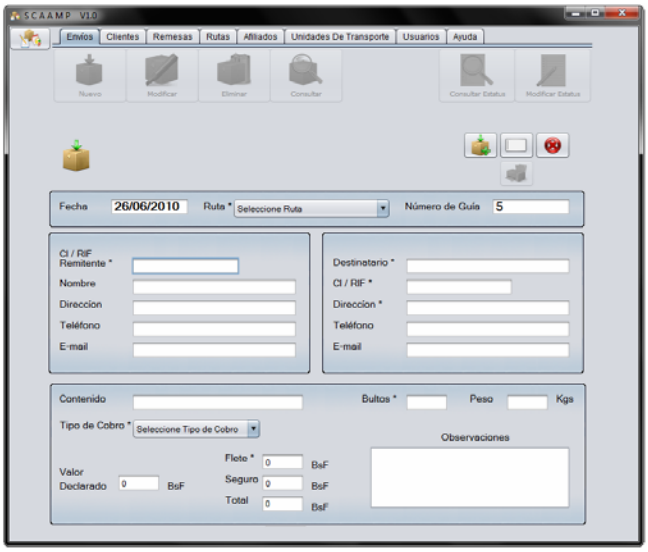

Figura 4.35. Ventana Nuevo Envío

Fuente: Elaboración Propia

- **Ventana Modificar Envío**

Por medio de la ventana "Modificar Envío" (*ver* **Figura 4.36**) los usuarios pueden modificar la información relacionada a un envío.

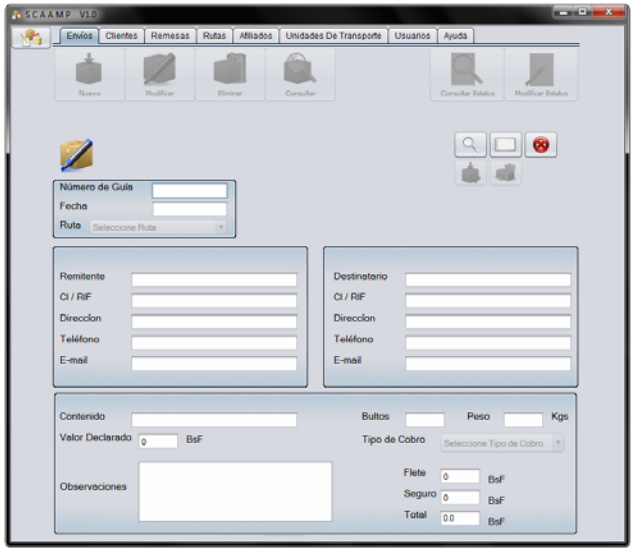

Figura 4.36. Ventana Modificar Envío

Fuente: Elaboración Propia

- **Ventana Eliminar Envío**

A través de la ventana “Eliminar Envío” (*ver* **Figura 4.37**) los usuarios del sistema pueden eliminar la información de cualquier envío que haya sido procesado.

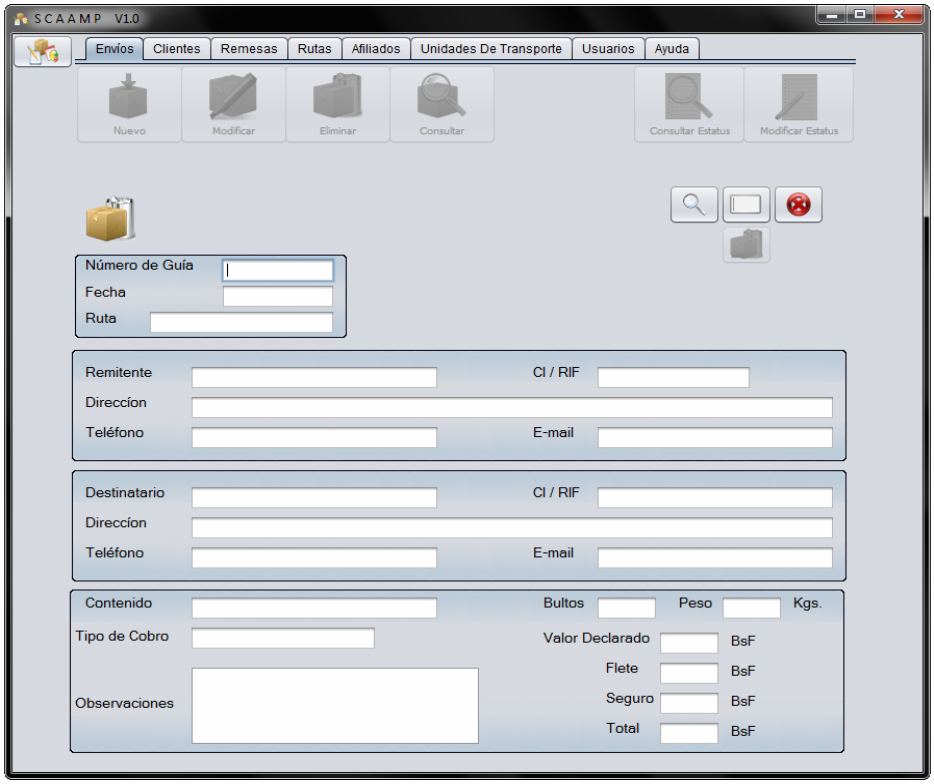

Figura 4.37. Ventana Eliminar Envío

Fuente: Elaboración Propia

- **Ventana Consultar Envío**

Por medio de esta ventana (*ver* **Figura 4.38**) los usuarios del sistema pueden visualizar la información de cualquier envío que haya sido procesado.

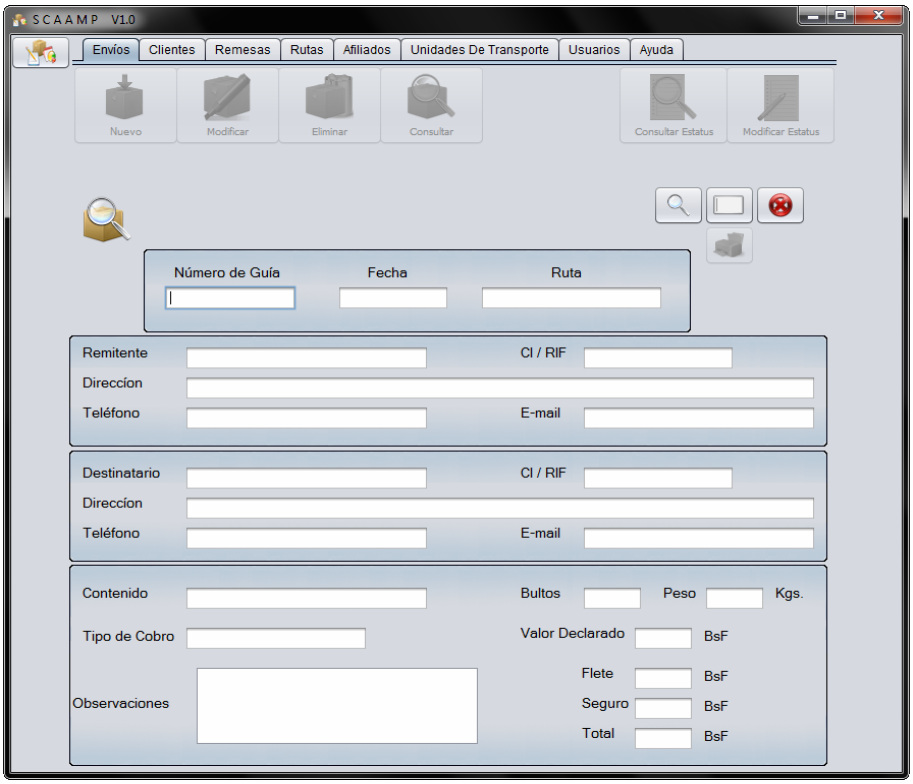

Figura 4.38. Ventana Consultar Envío

Fuente: Elaboración Propia

- **Menú "Clientes"**

A través de este menú (*ver* **Figura 4.39**) se le proporciona al usuario acceso a las cuatro operaciones básicas para el manejo de la información de clientes.

Figura 4.39. Menú Clientes

Fuente: Elaboración Propia

- **Menú “Remesas”**

Por medio de este menú (*ver* **Figura 4.40**) el usuario podrá Crear, Modificar, Eliminar o Consultar, información relacionada con las guías transportadas por cada unidad de transporte.

Figura 4.40. Menú Remesas

Fuente: Elaboración Propia

- **Ventana Crear Remesa**

Por medio de esta ventana (*ver* **Figura 4.41**) el usuario puede crear nuevas remesas con la información relacionada de las guías transportadas por cada unidad de transporte.

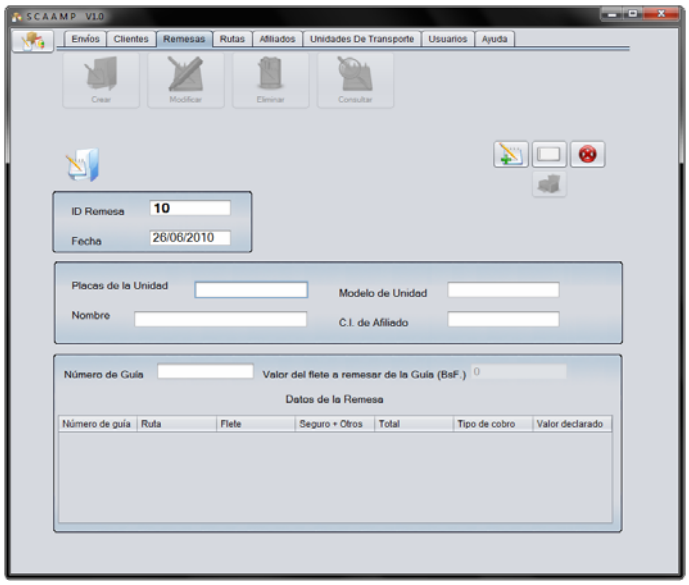

Figura 4.41. Ventana Crear Remesa

Fuente: Elaboración Propia

- **Ventana Modificar Remesa**

A través de la ventana “Modificar Remesa” (*ver* **Figura 4.42**) el usuario pueden modificar la información de las remesas que posee cada afiliado.

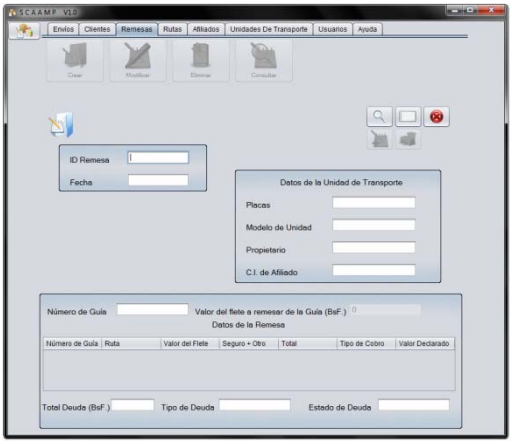

Figura 4.42. Ventana Modificar Remesa

Fuente: Elaboración Propia

- **Ventana Eliminar Remesa**

Por medio de esta ventana (*ver* **Figura 4.43**) el usuario puede eliminar las remesas asociadas a cada unidad de transporte.

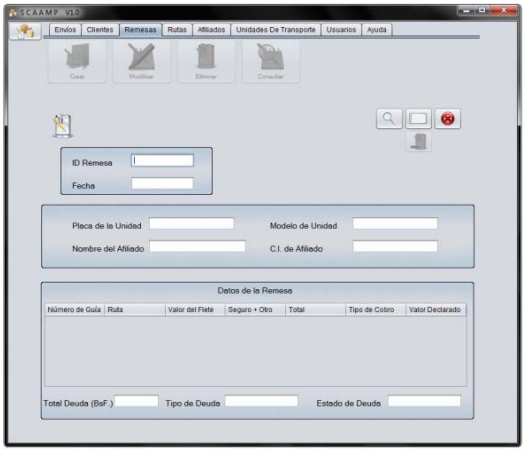

Figura 4.43. Ventana Eliminar Remesa

Fuente: Elaboración Propia

- **Ventana Consultar Remesa**

El usuario a través de la ventana “Consultar Remesa” (*ver* **Figura 4.44**) puede consultar la información de las remesas que poseen las unidades de transporte registradas en el sistema.

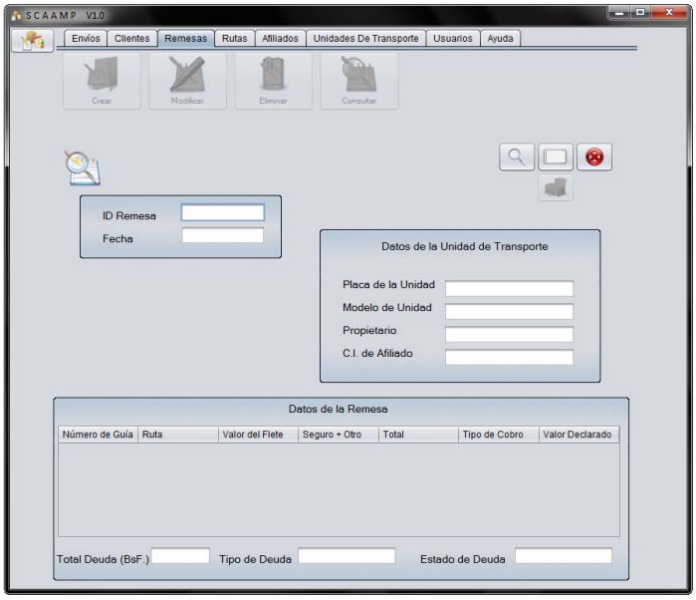

Figura 4.44. Ventana Consultar Remesa

Fuente: Elaboración Propia

- **Menú “Rutas”**

El menú “Rutas” (*ver* **Figura 4.45**) provee acceso a las distintas operaciones referidas al procesamiento de rutas.

Figura 4.45. Menú Rutas

Fuente: Elaboración Propia

- **Menú “Afiliados”**

El usuario podrá a través de este menú (*ver* **Figura 4.46**) ingresar a las distintas operaciones para gestionar a los afiliados.

Figura 4.46. Menú Afiliados

Fuente: Elaboración Propia

- **Ventana Realizar Pago**

Por medio de esta ventana los usuarios pueden realizar pagos en las cuentas de los afiliados (*ver* **Figura 4.47**).

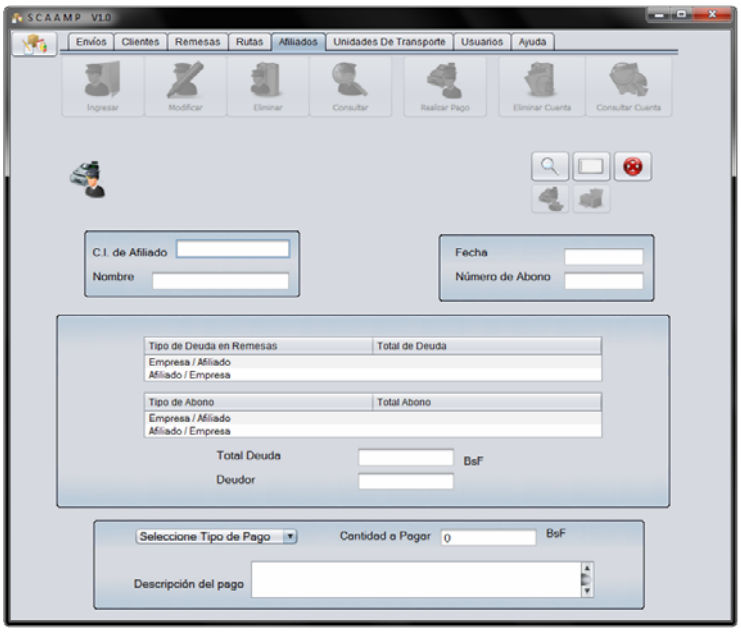

Figura 4.47. Ventana Realizar Pagos

Fuente: Elaboración Propia

- **Ventana Eliminar Cuenta**

Esta ventana permite a los usuarios visualizar la información de la cuenta del afiliado, y eliminar los datos que desee de la misma (*ver* **Figura 4.48**).

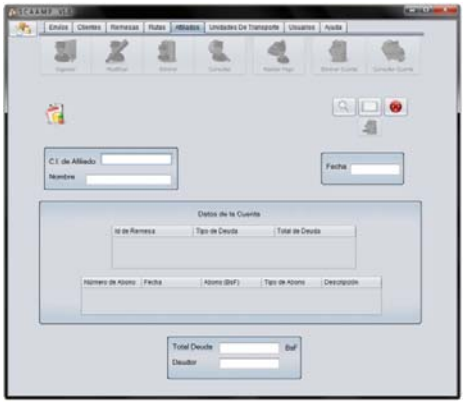

Figura 4.48. Ventana Eliminar Cuenta

Fuente: Elaboración Propia

- **Ventana Consultar Cuenta**

El usuario a través de la ventana “Consultar Cuenta” (*ver* **Figura 4.49**) puede consultar la información de las cuentas que poseen cada afiliado de la empresa.

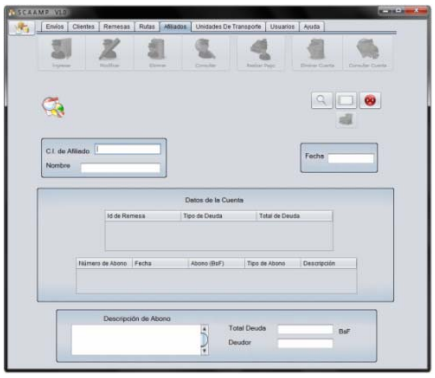

Figura 4.49. Ventana Consultar Cuenta

Fuente: Elaboración Propia

- **Menú “Unidades de Transporte”**

Por medio del menú “Unidades de Transporte” (*ver* **Figura 4.50**) el usuario tiene acceso a las rutinas para la manipulación de información de las unidades.

Figura 4.50. Menú Unidades de Transporte
Fuente: Elaboración Propia

- **Menú “Usuarios”**

Este menú provee al administrador las herramientas necesarias para el manejo de los registros de los usuarios del sistema (*ver* **Figura 4.51**).

Figura 4.51. Menú Usuarios
Fuente: Elaboración Propia

- **Menú “Ayuda”**

Por medio del menú “Ayuda” (*ver* **Figura 4.52**) el usuario tiene acceso al sistema de ayuda desarrollado para asistirlo durante el uso del sistema e información acerca de la aplicación.

Figura 4.52. Menú Ayuda

Fuente: Elaboración Propia

4.3 FLUJO DE TRABAJO DE IMPLEMENTACIÓN

En la implementación se comenzará con el uso de los resultados del diseño y se implementará el sistema en términos de componentes. El propósito principal de este flujo de trabajo es desarrollar la arquitectura y el sistema como un todo. Aquí se realizará el diagrama de componentes parcial de la aplicación y se implementará la interfaz de bienvenida a la aplicación y validación de usuarios, debido a que el resto se verán completamente implementadas en la fase de Construcción.

4.3.1 Diagrama de Componentes

Para la fase de elaboración se ha construido un adelanto de los componentes que conforman el sistema con sus relaciones. En la **Figura 4.53** se observan los componentes relacionados al ingreso de un nuevo cliente por parte del usuario general.

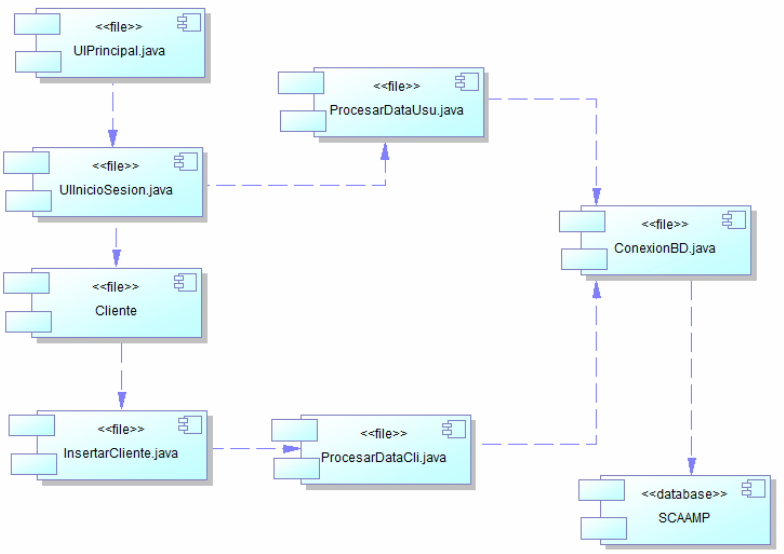

Figura 4.53. Diagrama parcial de Componentes (CU Ingresar Nuevo Cliente)

Fuente: Elaboración Propia

4.3.2 Implementación de los Componentes Asociados al Ingreso de un Nuevo Cliente

En la **Figura 4.54** se observa la interfaz para el inicio de sesión, con ésta estamos observando la ejecución de los componentes UIPrincipal, UIInicioSesion, ProcesarDataUsu, ConexioBD y SCAAMP, del diagrama de la **Figura 4.53**

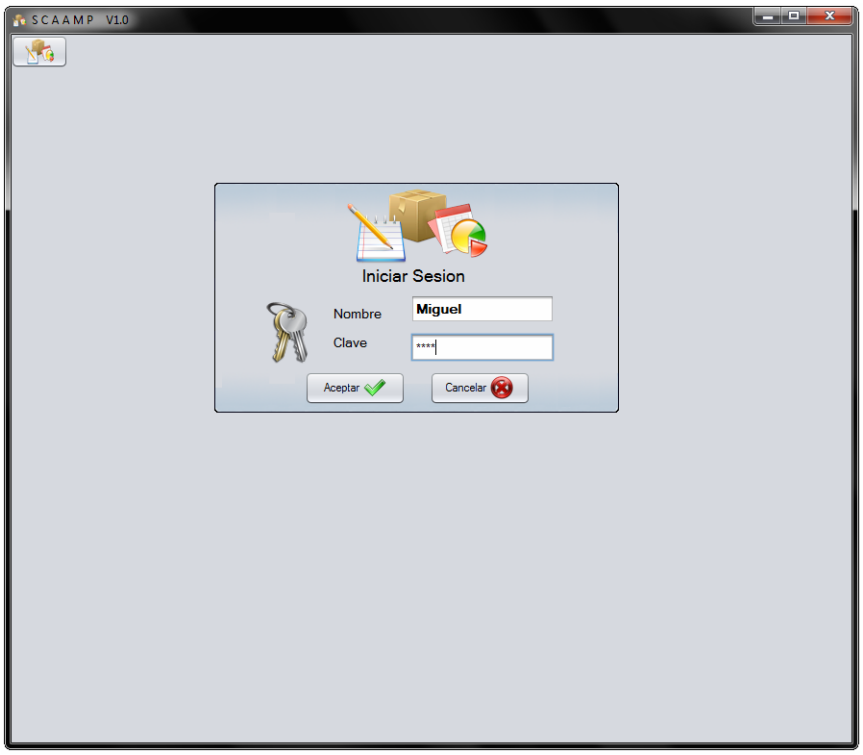

Figura 4.54. Interfaz de inicio de Sesión (autenticación de usuario)

Fuente: Elaboración Propia

```
package UIScaamp;

import Usuario.ProcesarDataUsu;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;


public class UIInicioSesion extends javax.swing.JPanel {

  /** Creates new form UIInicioSesion */
  private UIManager.LookAndFeelInfo apariencias[];

  public UIInicioSesion() {
    apariencias = UIManager.getInstalledLookAndFeels();

    try {
      UIManager.setLookAndFeel( apariencias[ 1 ].getClassName() );
      SwingUtilities.updateComponentTreeUI( this );
    }catch ( Exception excepcion ){
      excepcion.printStackTrace();
    }
```

```
    initComponents();
  }
  static public int usuario=0;

  private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel2 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jLabel4 = new javax.swing.JLabel();
    jTFLogin = new javax.swing.JTextField();
    BotonAceptarSesion = new javax.swing.JButton();
    BotonCancelarSecion = new javax.swing.JButton();
    jTFPassword = new javax.swing.JPasswordField();

    setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
    setPreferredSize(new java.awt.Dimension(340, 165));
    setLayout(null);

    jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/password.png"))); // NOI18N
    add(jLabel1);
    jLabel1.setBounds(10, 40, 64, 80);

    jLabel2.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 18));
    jLabel2.setText("Iniciar Sesion");
    add(jLabel2);
    jLabel2.setBounds(120, 10, 121, 21);

    jLabel3.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14)); // NOI18N
    jLabel3.setText("Nombre");
    add(jLabel3);
    jLabel3.setBounds(90, 50, 60, 20);

    jLabel4.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel4.setText("Clave");
    add(jLabel4);
    jLabel4.setBounds(90, 80, 70, 20);

    jTFLogin.setFont(new java.awt.Font("Microsoft Sans Serif", 1, 14));
    jTFLogin.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTFLoginActionPerformed(evt);
      }
    });
    add(jTFLogin);
    jTFLogin.setBounds(170, 40, 150, 30);

    BotonAceptarSesion.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonAceptarSesion.setIcon(new   javax.swing.ImageIcon(getClass().getResource("/Imagenes/apply.png")));   //
NOI18N
    BotonAceptarSesion.setText("Aceptar");
```

```
    BotonAceptarSesion.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
    BotonAceptarSesion.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonAceptarSesionActionPerformed(evt);
      }
    });
    add(BotonAceptarSesion);
    BotonAceptarSesion.setBounds(60, 120, 105, 35);

    BotonCancelarSecion.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonCancelarSecion.setIcon(new    javax.swing.ImageIcon(getClass().getResource("/Imagenes/cancel.png")));
// NOI18N
    BotonCancelarSecion.setText("Cancelar");
    BotonCancelarSecion.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
    BotonCancelarSecion.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonCancelarSecionActionPerformed(evt);
      }
    });
    add(BotonCancelarSecion);
    BotonCancelarSecion.setBounds(190, 120, 105, 35);

    jTFPassword.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTFPasswordActionPerformed(evt);
      }
    });
    add(jTFPassword);
    jTFPassword.setBounds(170, 80, 150, 30);
  }// </editor-fold>

  private void BotonCancelarSecionActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.setVisible(false);
  }

  private void BotonAceptarSesionActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Accesar();
  }

  private void jTFLoginActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    jTFPassword.requestFocus();
  }

  private void jTFPasswordActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Accesar();
  }
```

```
 public void Accesar(){
   usuario=procesar.VerificarAcceso();
   if(usuario==1){
     ActivarSesion();
     PanelMenu.ActivarAdministrador();
     PanelMenu.jTPPrincipal.setSelectedIndex(0);
   }
   else if(usuario==2){
     ActivarSesion();
     PanelMenu.ActivarUsuario();
     PanelMenu.jTPPrincipal.setSelectedIndex(0);
   }
 }

 public void ActivarSesion(){
   UIPrincipal.PanMen.setVisible(true);
   UIPrincipal.jBotonConsultas.setVisible(true);
   UIPrincipal.BotonInicioSesion.setEnabled(false);
   UIPrincipal.BotonCambiarUsuario.setEnabled(true);
   UIPrincipal.BotonCerrarSesion.setEnabled(true);
   this.setVisible(false);
 }

 // Variables declaration - do not modify
 private javax.swing.JButton BotonAceptarSesion;
 private javax.swing.JButton BotonCancelarSecion;
 private javax.swing.JLabel jLabel1;
 private javax.swing.JLabel jLabel2;
 private javax.swing.JLabel jLabel3;
 private javax.swing.JLabel jLabel4;
 public static javax.swing.JTextField jTFLogin;
 public static javax.swing.JPasswordField jTFPassword;
 // End of variables declaration

 ProcesarDataUsu procesar = new ProcesarDataUsu();

}
```

En las **Figuras 4.54** y **4.55** se observan las interfaces para el ingreso de un nuevo cliente, en las cuales se hace referencia a los componentes Cliente, ProcesarDataCli, ConexioBD y SCAAMP, del diagrama de la **Figura 4.53**

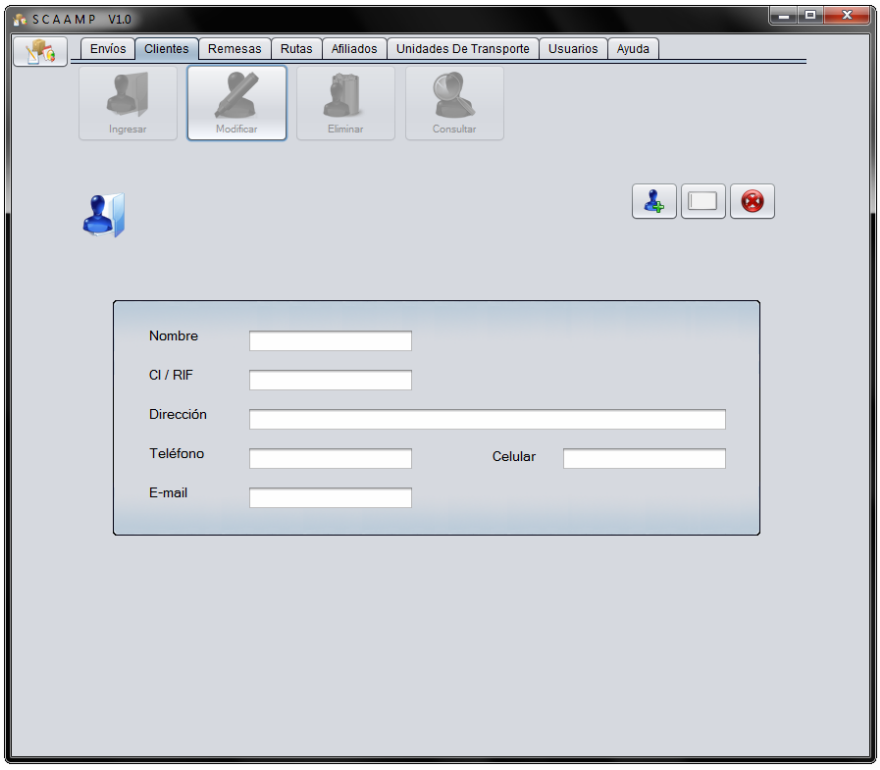

Figura 4.54. Interfaz Nuevo Cliente

Fuente: Elaboración Propia

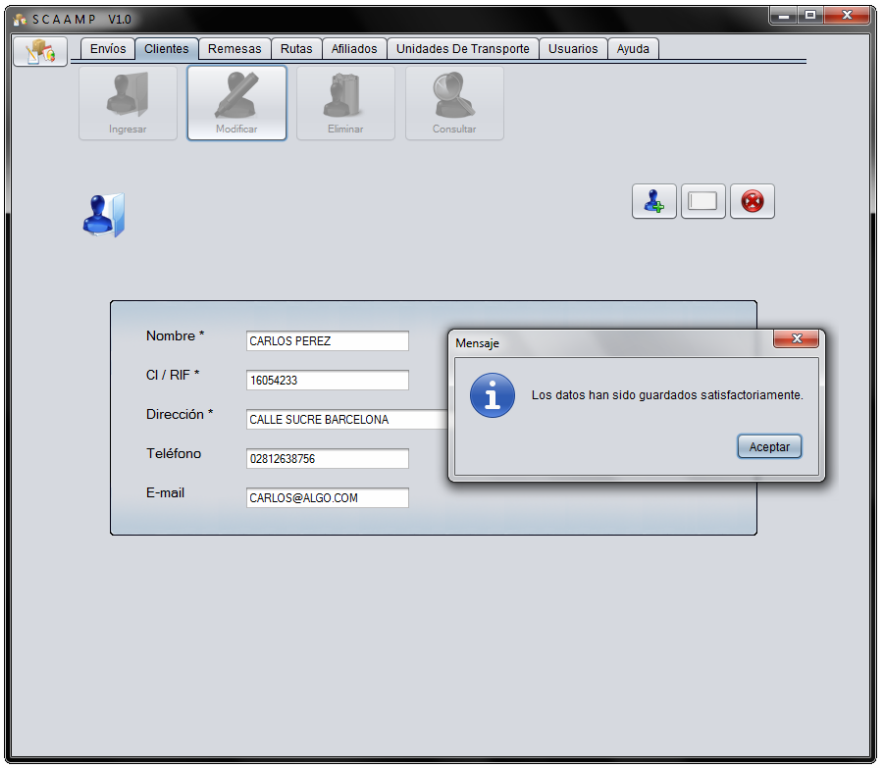

Figura 4.55. Interfaz Nuevo Cliente

Fuente: Elaboración Propia

```
package Cliente;

import UIScaamp.UIPrincipal;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;

public class UIInsertarCliente extends javax.swing.JPanel {

 /** Creates new form UIInsertarCliente */
 private UIManager.LookAndFeelInfo apariencias[];

 public UIInsertarCliente() {

  apariencias = UIManager.getInstalledLookAndFeels();

  try {
   UIManager.setLookAndFeel( apariencias[ 1 ].getClassName() );
   SwingUtilities.updateComponentTreeUI( this );
  }catch ( Exception excepcion ){
   excepcion.printStackTrace();
```

```
    }
    initComponents();
  }


  private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    BotonGuardarCliente = new javax.swing.JButton();
    jLabel3 = new javax.swing.JLabel();
    jTFNombreCliente = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jTFCedRifCliente = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();
    jTFDireccionCliente = new javax.swing.JTextField();
    jLabel6 = new javax.swing.JLabel();
    jTFTelefonoCliente = new javax.swing.JTextField();
    jTFCelularCliente = new javax.swing.JTextField();
    jLabel8 = new javax.swing.JLabel();
    jTFEmailCliente = new javax.swing.JTextField();
    jLabel7 = new javax.swing.JLabel();
    BotonCancelarCliente = new javax.swing.JButton();
    BotonLimpiarAfiliado = new javax.swing.JButton();

    setPreferredSize(new java.awt.Dimension(630, 340));
    setLayout(null);

    jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/newcliente.png"))); // NOI18N
    add(jLabel1);
    jLabel1.setBounds(30, 90, 40, 40);

    BotonGuardarCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonGuardarCliente.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Imagenes/add_user.png"))); // NOI18N
    BotonGuardarCliente.setToolTipText("Guardar Nuevo Cliente");
    BotonGuardarCliente.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonGuardarClienteActionPerformed(evt);
      }
    });
    add(BotonGuardarCliente);
    BotonGuardarCliente.setBounds(510, 30, 50, 40);

    jLabel3.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel3.setText("Nombre");
    add(jLabel3);
    jLabel3.setBounds(20, 140, 60, 15);

    jTFNombreCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    jTFNombreCliente.addFocusListener(new java.awt.event.FocusAdapter() {
      public void focusLost(java.awt.event.FocusEvent evt) {
```

```
    jTFNombreClienteFocusLost(evt);
  }
});
jTFNombreCliente.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    jTFNombreClienteKeyTyped(evt);
  }
});
add(jTFNombreCliente);
jTFNombreCliente.setBounds(120, 140, 170, 25);

jLabel4.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel4.setText("CI / RIF");
add(jLabel4);
jLabel4.setBounds(20, 180, 60, 15);

jTFCedRifCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
jTFCedRifCliente.addFocusListener(new java.awt.event.FocusAdapter() {
  public void focusLost(java.awt.event.FocusEvent evt) {
    jTFCedRifClienteFocusLost(evt);
  }
});
jTFCedRifCliente.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    jTFCedRifClienteKeyTyped(evt);
  }
});
add(jTFCedRifCliente);
jTFCedRifCliente.setBounds(120, 180, 170, 25);

jLabel5.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel5.setText("Dirección");
add(jLabel5);
jLabel5.setBounds(20, 220, 60, 15);

jTFDireccionCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
jTFDireccionCliente.addFocusListener(new java.awt.event.FocusAdapter() {
  public void focusLost(java.awt.event.FocusEvent evt) {
    jTFDireccionClienteFocusLost(evt);
  }
});
jTFDireccionCliente.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    jTFDireccionClienteKeyTyped(evt);
  }
});
add(jTFDireccionCliente);
jTFDireccionCliente.setBounds(120, 220, 490, 25);

jLabel6.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel6.setText("Teléfono");
```

```
    add(jLabel6);
    jLabel6.setBounds(20, 260, 60, 15);

    jTFTelefonoCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    jTFTelefonoCliente.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        jTFTelefonoClienteKeyTyped(evt);
      }
    });
    add(jTFTelefonoCliente);
    jTFTelefonoCliente.setBounds(120, 260, 170, 25);

    jTFCelularCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    jTFCelularCliente.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        jTFCelularClienteKeyTyped(evt);
      }
    });
    add(jTFCelularCliente);
    jTFCelularCliente.setBounds(440, 260, 170, 25);

    jLabel8.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel8.setText("Celular");
    add(jLabel8);
    jLabel8.setBounds(370, 260, 60, 20);

    jTFEmailCliente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    jTFEmailCliente.addFocusListener(new java.awt.event.FocusAdapter() {
      public void focusLost(java.awt.event.FocusEvent evt) {
        jTFEmailClienteFocusLost(evt);
      }
    });
    jTFEmailCliente.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        jTFEmailClienteKeyTyped(evt);
      }
    });
    add(jTFEmailCliente);
    jTFEmailCliente.setBounds(120, 300, 170, 25);

    jLabel7.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel7.setText("E-mail");
    add(jLabel7);
    jLabel7.setBounds(20, 300, 60, 15);

    BotonCancelarCliente.setIcon(new   javax.swing.ImageIcon(getClass().getResource("/Imagenes/cancel.png")));
// NOI18N
    BotonCancelarCliente.setToolTipText("Cancelar");
    BotonCancelarCliente.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonCancelarCliente.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        BotonCancelarClienteActionPerformed(evt);
      }
    });
    add(BotonCancelarCliente);
    BotonCancelarCliente.setBounds(560, 30, 50, 40);

    BotonLimpiarAfiliado.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonLimpiarAfiliado.setIcon(new  javax.swing.ImageIcon(getClass().getResource("/Imagenes/editclear.png")));
// NOI18N
    BotonLimpiarAfiliado.setToolTipText("Limpiar Formularios");
    BotonLimpiarAfiliado.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonLimpiarAfiliado.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonLimpiarAfiliadoActionPerformed(evt);
      }
    });
    add(BotonLimpiarAfiliado);
    BotonLimpiarAfiliado.setBounds(510, 70, 50, 40);
  }// </editor-fold>

  private void BotonCancelarClienteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Limpiar();
    this.setVisible(false);
    UIPrincipal.PanMen.ActivarAdministrador();
}

  private void BotonGuardarClienteActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

if(jTFNombreCliente.getText().equals("")||jTFCedRifCliente.getText().equals("")||jTFDireccionCliente.getText().equ
als("")){
      JOptionPane.showMessageDialog(null, " FALTAN DATOS POR COMPLETAR\nVerificar: Nombre, CI/RIF y
Dirección");
    }
    else{
    int salir = JOptionPane.showConfirmDialog(null,"Seguro Desea Guardar Los Datos?",
        "ADVERTENCIA",0);
    if(salir == 0){
      procesar.InsertarCliente();
      Limpiar();
    }
    }
  }

  private void BotonLimpiarAfiliadoActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Limpiar();
}

  private void jTFNombreClienteFocusLost(java.awt.event.FocusEvent evt) {
```

```
  // TODO add your handling code here:
  jTFNombreCliente.setText(jTFNombreCliente.getText().toUpperCase());
}

private void jTFCedRifClienteFocusLost(java.awt.event.FocusEvent evt) {
  // TODO add your handling code here:
  jTFCedRifCliente.setText(jTFCedRifCliente.getText().toUpperCase());
}

private void jTFDireccionClienteFocusLost(java.awt.event.FocusEvent evt) {
  // TODO add your handling code here:
  jTFDireccionCliente.setText(jTFDireccionCliente.getText().toUpperCase());
}

private void jTFEmailClienteFocusLost(java.awt.event.FocusEvent evt) {
  // TODO add your handling code here:
  jTFEmailCliente.setText(jTFEmailCliente.getText().toUpperCase());
}

private void jTFNombreClienteKeyTyped(java.awt.event.KeyEvent evt) {
  // TODO add your handling code here:
  if(jTFNombreCliente.getText().length()>=40)
    evt.consume();
}

private void jTFCedRifClienteKeyTyped(java.awt.event.KeyEvent evt) {
  // TODO add your handling code here:
  if(jTFCedRifCliente.getText().length()>=13)
    evt.consume();
  else{
    char caracter=evt.getKeyChar();
    if((caracter<'0'||caracter>'9')&&caracter!=evt.VK_BACK_SPACE&&caracter!='-'&&caracter!='J'&&caracter!='j')
      evt.consume();
  }
}

private void jTFDireccionClienteKeyTyped(java.awt.event.KeyEvent evt) {
  // TODO add your handling code here:
  if(jTFDireccionCliente.getText().length()>=100)
    evt.consume();
}

private void jTFTelefonoClienteKeyTyped(java.awt.event.KeyEvent evt) {
  // TODO add your handling code here:
  if(jTFTelefonoCliente.getText().length()>=13)
    evt.consume();
  else{
    char caracter=evt.getKeyChar();
    if((caracter<'0'||caracter>'9')&&caracter!=evt.VK_BACK_SPACE&&caracter!='-')
      evt.consume();
  }
```

```
    }

    private void jTFCelularClienteKeyTyped(java.awt.event.KeyEvent evt) {
      // TODO add your handling code here:
      if(jTFCelularCliente.getText().length()>=13)
        evt.consume();
      else{
        char caracter=evt.getKeyChar();
        if((caracter<'0'||caracter>'9')&&caracter!=evt.VK_BACK_SPACE&&caracter!='-')
          evt.consume();
      }
    }

    private void jTFEmailClienteKeyTyped(java.awt.event.KeyEvent evt) {
      // TODO add your handling code here:
      if(jTFEmailCliente.getText().length()>=30)
        evt.consume();
    }

    public void Limpiar(){
      jTFCedRifCliente.setText("");
      jTFNombreCliente.setText("");
      jTFDireccionCliente.setText("");
      jTFTelefonoCliente.setText("");
      jTFCelularCliente.setText("");
      jTFEmailCliente.setText("");
    }

    // Variables declaration - do not modify
    private javax.swing.JButton BotonCancelarCliente;
    private javax.swing.JButton BotonGuardarCliente;
    private javax.swing.JButton BotonLimpiarAfiliado;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JLabel jLabel8;
    public static javax.swing.JTextField jTFCedRifCliente;
    public static javax.swing.JTextField jTFCelularCliente;
    public static javax.swing.JTextField jTFDireccionCliente;
    public static javax.swing.JTextField jTFEmailCliente;
    public static javax.swing.JTextField jTFNombreCliente;
    public static javax.swing.JTextField jTFTelefonoCliente;
    // End of variables declaration

    ProcesarDataCli procesar = new ProcesarDataCli();
}
```

4.4 EVALUACIÓN DE LA FASE DE ELABORACIÓN

Al comienzo de la fase de elaboración, se recibió de la fase de inicio la base para su desarrollo, un modelo de casos de uso completo y una descripción de la arquitectura candidata. Durante el desarrollo de los flujos de trabajo se obtuvieron las diferentes vistas de la arquitectura del sistema, estableciendo la prioridad de los casos de uso, la línea base de la arquitectura soporta los casos de uso críticos del sistema, ya que los riesgos críticos fueron tratados a través de los casos de uso, siendo mitigados al implementar los casos de uso correspondientes. El objetivo principal de esta fase, fue llevar a cabo el diseño y estructura del sistema para dar soporte a todos los requisitos que se obtuvieron durante la fase de inicio.

CAPITULO V
FASE DE CONSTRUCCIÓN

INTRODUCCIÓN

El propósito de la fase de construcción es producir la primera versión operativa de calidad del sistema, llamada versión *beta*. Para este objetivo se tomó como base los resultados obtenidos en las fases anteriores, para construir un producto final donde los requerimientos de los usuarios sean plenamente satisfechos, además de realizar un conjunto de pruebas que garanticen el correcto funcionamiento de la aplicación.

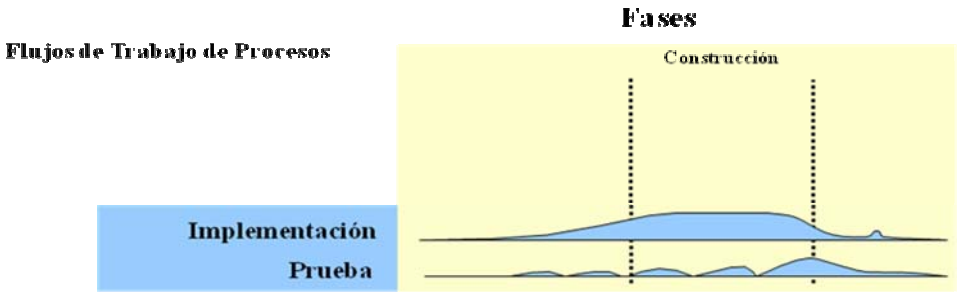

Figura 5.1. Flujos de trabajo de la fase de elaboración.
Fuente: Rumbaugh, 2004

En esta fase se hace hincapié en las iteraciones de implementación y prueba del flujo de trabajo normal (*ver* **Figura 5.1**), debido a que su meta es lograr el desarrollo del sistema con calidad de producción, mediante la implementación de toda la funcionalidad y realización de las pruebas.

5.1 FLUJO DE TRABAJO DE IMPLEMENTACIÓN

En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. La implementación trata al sistema en términos de desarrollo de componentes y codificación del software, su relación con la base de datos, integración de módulos y la explicación acerca de las funcionalidades del sistema y su correcto uso, delatando así toda la estructura en forma de código abierto e identificación de las actividades que éste puede realizar.

5.1.1 Escogencia del Lenguaje de Programación

Para el desarrollo del sistema SCAAMP, se debe contar con un lenguaje que soporte la programación orientada a objetos, ya que esta promueve una mejor comprensión de los requisitos, diseños más limpios y sistemas más fáciles de mantener.

Es por esto que se escogió como lenguaje de programación JAVA, debido a que es un lenguaje simple, seguro y multiplataforma que puede ser usado indistintamente bajo cualquier sistema operativo sin necesidad de hacer cambios en la programación.

Como entorno de desarrollo se escogió el IDE NetBeans. Algunas de las características que hacen interesante la elección de NetBeans como plataforma para nuestra aplicación son las siguientes:

- Los proyectos desarrollados no dejan de ser multiplataforma, y poseen largadores (launchers) para cada plataforma.

- Sistema de ventanas práctico para desarrollar las interfaces de usuario
- Sistema de fichero virtual en donde se van montando los diferentes módulos con el cual se van adaptando automáticamente los menús, barra de herramientas, menús contextuales, entre otros, de la aplicación
- Su licencia nos permite construir tanto aplicaciones *open source* como comerciales
- No es obligatorio que una aplicación deba tener interfaz de usuario gráfica (GUI), ya que la plataforma permite dejar de lado la misma y seguir disfrutando del resto de los beneficios, por ejemplo la actualización de módulos desde un repositorio remoto.
- Soporte completo para desarrollar desde NetBeans IDE, por lo que no necesitaremos otra herramienta adicional para el desarrollo

En la **Figura 5.2** observamos la interfaz del software NetBeans, a través del cual se realizó la programación de la aplicación en el lenguaje JAVA.

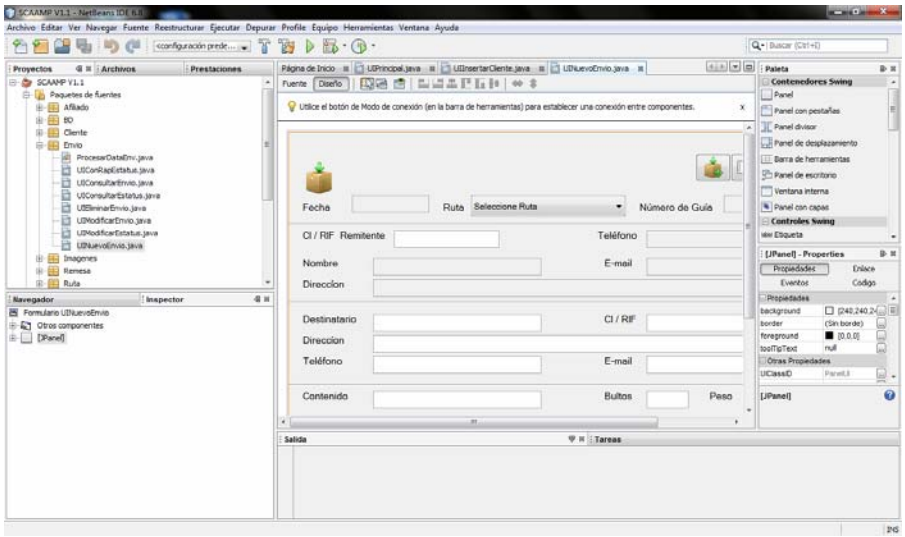

Figura 5.2. Interfaz del entorno de desarrollo JAVA

Fuente: Elaboración Propia

5.1.2 Escogencia del Gestor de Base de Datos

Para la construcción de la base de datos de SCAAMP, se escogió PostgreSQL, es un sistema de gestión de base de datos relacional, multihilo y multiusuario, además, es altamente confiable en cuanto a estabilidad se refiere. El cliente gráfico de PostgreSQL es el pgAdmin III en el cual podemos ver y trabajar con casi todos los objetos de la base de datos, examinar sus propiedades y realizar tareas administrativas.

En la **Figura 5.3** se observa el entorno por medio del cual se construyeron las tablas que constituyen la base de datos SCAAMP.

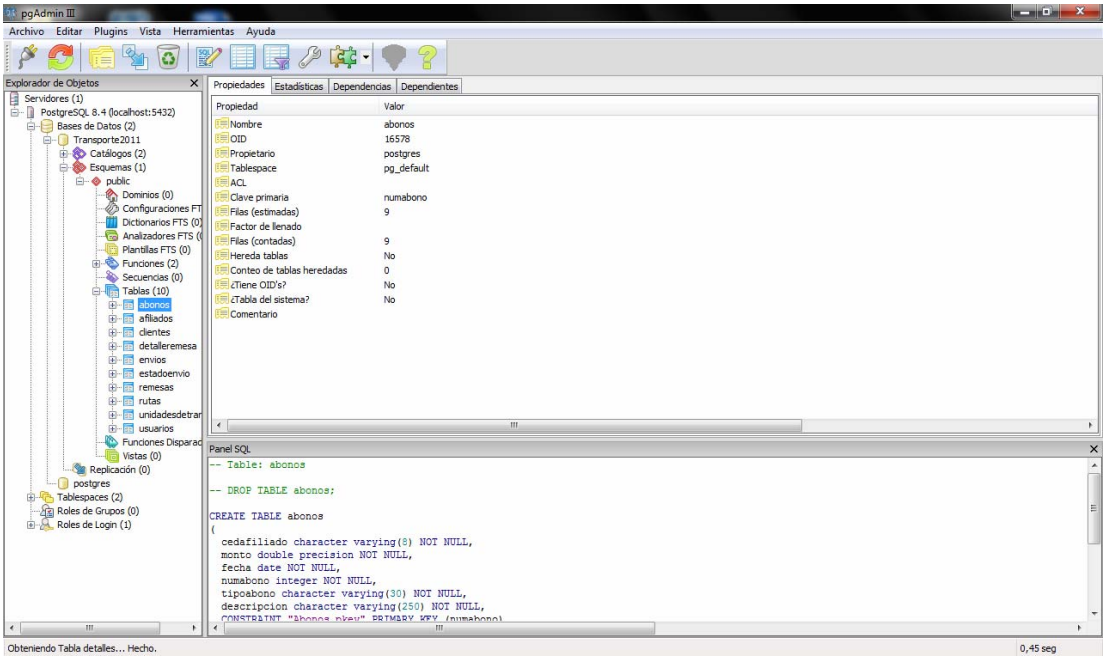

Figura 5.3. Interfaz del entorno gráfico de PostgreSQL

Fuente: Elaboración Propia

5.1.3 Diagrama de Componentes Total

Un componente es un módulo de código, así mismo el diagrama de componentes permite representar los componentes del sistema así como las relaciones que existen entre dichos componentes En esta fase se desarrolla el diagrama de componentes con la totalidad de los componentes del sistema, indicando las clases que necesitan cada uno.

En la **Figura 5.4** se puede observar el diagrama con la totalidad de los componentes.

5.1.4 Descripción de Componentes

A continuación se muestra la descripción de las principales áreas dentro de la aplicación luego de su implementación. Se llevará a cabo bajo el siguiente formato:

- Breve explicación del área (en qué consiste).
- Identificación de los datos a manejar por el área. (datos que debe introducir el usuario durante su visita).
- Imagen referencial de la pantalla del sistema.
- Código Fuente

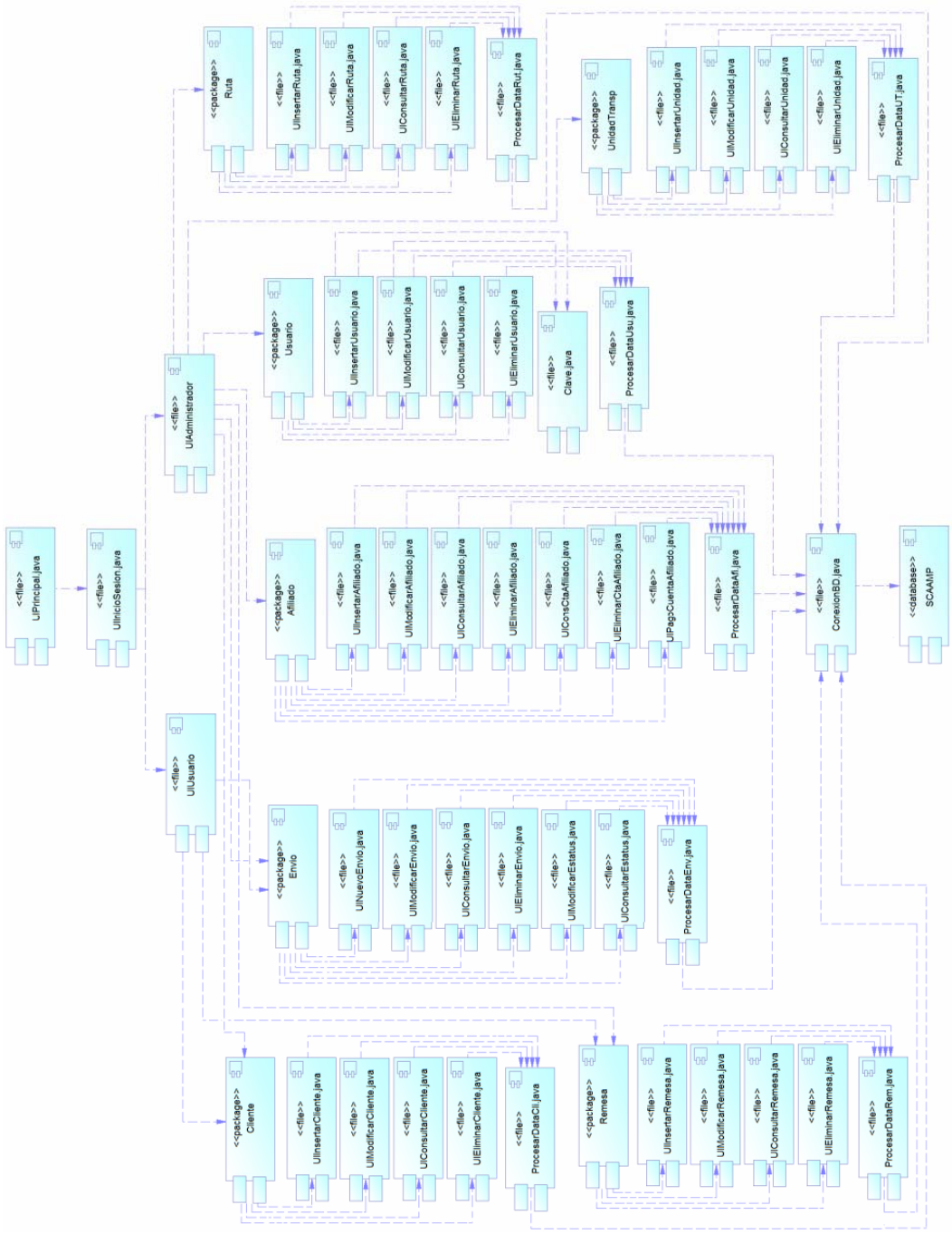

Figura 5.4. Diagrama de Componentes Total SCAAMP

Fuente: Elaboración Propia

5.1.4.1 Área Envíos

Durante el modelado de la aplicación se definió el área de envíos, que contempla las interfaces ingresar, modificar, eliminar y consultar, así como también consultar y modificar estatus. Esta vista está disponible para todos los usuarios del sistema.

En la **Figura 5.5** mostrada a continuación, se observa la interfaz consultar envío, en la cual el usuario debe ingresar el número de guía para poder visualizar toda la información relacionada ese envío.

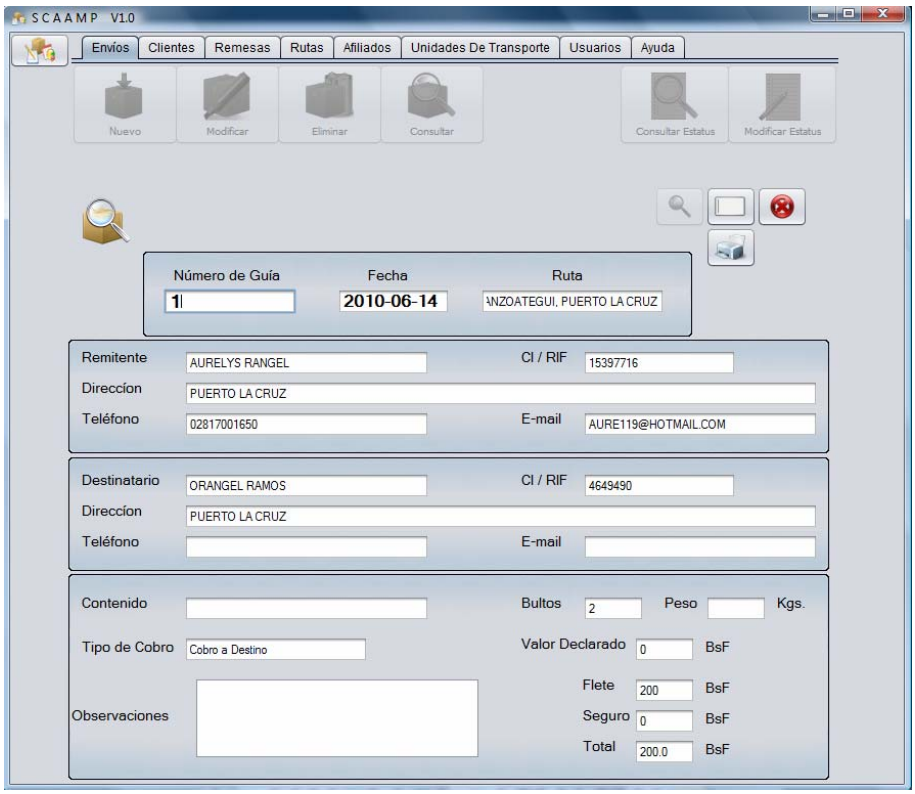

Figura 5.5. Pantalla Consultar Envío: Área Envíos

Fuente: Elaboración Propia

Código Fuente Consultar Envío

```
package Envio;

import UIScaamp.UIPrincipal;
import javax.swing.JOptionPane;

public class UIConsultarEnvio extends javax.swing.JPanel {

  /** Creates new form UIConsultarEnvio */
  public UIConsultarEnvio() {
    initComponents();
  }

  private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jTFNumGuia = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jTFNombreRemitente = new javax.swing.JTextField();
    jLabel6 = new javax.swing.JLabel();
    jTFTelefonoRemitente = new javax.swing.JTextField();
    jTFCedRifRemitente = new javax.swing.JTextField();
    jLabel7 = new javax.swing.JLabel();
    jTFFecha = new javax.swing.JTextField();
    jTFDireccionDestinatario = new javax.swing.JTextField();
    jLabel9 = new javax.swing.JLabel();
    jTFCedRifDestinatario = new javax.swing.JTextField();
    jLabel10 = new javax.swing.JLabel();
    jTFNombreDestinatario = new javax.swing.JTextField();
    jLabel11 = new javax.swing.JLabel();
    jTFTelefonoDestinatario = new javax.swing.JTextField();
    jLabel12 = new javax.swing.JLabel();
    jTFDireccionRemitente = new javax.swing.JTextField();
    jLabel13 = new javax.swing.JLabel();
    jTFTipoCobro = new javax.swing.JTextField();
    jLabel29 = new javax.swing.JLabel();
    jLabel8 = new javax.swing.JLabel();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTAObservaciones = new javax.swing.JTextArea();
    BotonConsultarEnvio = new javax.swing.JButton();
    BotonCancelar = new javax.swing.JButton();
    jLabel14 = new javax.swing.JLabel();
    jTFRuta = new javax.swing.JTextField();
    jLabel15 = new javax.swing.JLabel();
    jTFContenido = new javax.swing.JTextField();
    jLabel16 = new javax.swing.JLabel();
    jTFBultos = new javax.swing.JTextField();
```

```
    jLabel17 = new javax.swing.JLabel();
    jTFPeso = new javax.swing.JTextField();
    jLabel18 = new javax.swing.JLabel();
    jLabel19 = new javax.swing.JLabel();
    jTFFlete = new javax.swing.JTextField();
    jLabel20 = new javax.swing.JLabel();
    jTFSeguro = new javax.swing.JTextField();
    jLabel21 = new javax.swing.JLabel();
    jTFTotal = new javax.swing.JTextField();
    jLabel25 = new javax.swing.JLabel();
    jLabel26 = new javax.swing.JLabel();
    jLabel27 = new javax.swing.JLabel();
    jTFValorDeclarado = new javax.swing.JTextField();
    jLabel23 = new javax.swing.JLabel();
    jLabel24 = new javax.swing.JLabel();
    jLabel28 = new javax.swing.JLabel();
    jLabel22 = new javax.swing.JLabel();
    jTFEmailRemitente = new javax.swing.JTextField();
    BotonLimpia = new javax.swing.JButton();
    jSeparator1 = new javax.swing.JSeparator();
    jSeparator2 = new javax.swing.JSeparator();
    jSeparator3 = new javax.swing.JSeparator();
    jLabel30 = new javax.swing.JLabel();
    jTFEmailDestinatario = new javax.swing.JTextField();

    setLayout(null);

    jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/buscar_caja.png"))); //
NOI18N
    add(jLabel1);
    jLabel1.setBounds(20, 40, 50, 50);

    jLabel3.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel3.setText("CI / RIF ");
    add(jLabel3);
    jLabel3.setBounds(450, 180, 50, 15);

    jTFNumGuia.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 18)); // NOI18N
    jTFNumGuia.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTFNumGuiaActionPerformed(evt);
      }
    });
    jTFNumGuia.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        jTFNumGuiaKeyTyped(evt);
      }
    });
    add(jTFNumGuia);
    jTFNumGuia.setBounds(140, 100, 130, 25);
```

```
jLabel4.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel4.setText("Fecha");
add(jLabel4);
jLabel4.setBounds(560, 130, 50, 15);

jLabel5.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel5.setText("Remitente");
add(jLabel5);
jLabel5.setBounds(20, 180, 80, 15);

jTFNombreRemitente.setEditable(false);
jTFNombreRemitente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFNombreRemitente);
jTFNombreRemitente.setBounds(120, 180, 240, 25);

jLabel6.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel6.setText("TelÃ©fono");
add(jLabel6);
jLabel6.setBounds(20, 240, 60, 15);

jTFTelefonoRemitente.setEditable(false);
jTFTelefonoRemitente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFTelefonoRemitente);
jTFTelefonoRemitente.setBounds(120, 240, 240, 25);

jTFCedRifRemitente.setEditable(false);
jTFCedRifRemitente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFCedRifRemitente);
jTFCedRifRemitente.setBounds(510, 180, 150, 25);

jLabel7.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel7.setText("NÃºmero de GuÃa");
add(jLabel7);
jLabel7.setBounds(20, 100, 110, 15);

jTFFecha.setEditable(false);
jTFFecha.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFFecha);
jTFFecha.setBounds(630, 130, 110, 25);

jTFDireccionDestinatario.setEditable(false);
jTFDireccionDestinatario.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFDireccionDestinatario);
jTFDireccionDestinatario.setBounds(120, 320, 620, 25);

jLabel9.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel9.setText("CI / RIF");
add(jLabel9);
jLabel9.setBounds(450, 290, 60, 15);

jTFCedRifDestinatario.setEditable(false);
```

```
jTFCedRifDestinatario.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFCedRifDestinatario);
jTFCedRifDestinatario.setBounds(510, 290, 230, 25);

jLabel10.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel10.setText("Destinatario");
add(jLabel10);
jLabel10.setBounds(20, 290, 80, 15);

jTFNombreDestinatario.setEditable(false);
jTFNombreDestinatario.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFNombreDestinatario);
jTFNombreDestinatario.setBounds(120, 290, 240, 25);

jLabel11.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel11.setText("TelÃ©fono");
add(jLabel11);
jLabel11.setBounds(20, 350, 80, 15);

jTFTelefonoDestinatario.setEditable(false);
jTFTelefonoDestinatario.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFTelefonoDestinatario);
jTFTelefonoDestinatario.setBounds(120, 350, 240, 25);

jLabel12.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel12.setText("DireccÃon");
add(jLabel12);
jLabel12.setBounds(20, 210, 80, 15);

jTFDireccionRemitente.setEditable(false);
jTFDireccionRemitente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFDireccionRemitente);
jTFDireccionRemitente.setBounds(120, 210, 620, 25);

jLabel13.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel13.setText("E-mail");
add(jLabel13);
jLabel13.setBounds(450, 350, 60, 15);

jTFTipoCobro.setEditable(false);
jTFTipoCobro.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFTipoCobro);
jTFTipoCobro.setBounds(560, 430, 180, 25);

jLabel29.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel29.setText("DireccÃon");
add(jLabel29);
jLabel29.setBounds(20, 320, 80, 15);

jLabel8.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel8.setText("Observaciones");
```

```
    add(jLabel8);
    jLabel8.setBounds(10, 510, 100, 15);

    jTAObservaciones.setColumns(20);
    jTAObservaciones.setEditable(false);
    jTAObservaciones.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 12));
    jTAObservaciones.setRows(3);
    jTAObservaciones.setTabSize(5);
    jScrollPane2.setViewportView(jTAObservaciones);

    add(jScrollPane2);
    jScrollPane2.setBounds(130, 480, 280, 80);

    BotonConsultarEnvio.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonConsultarEnvio.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/search.png")));
// NOI18N
    BotonConsultarEnvio.setToolTipText("Consultar EnvÃo");
    BotonConsultarEnvio.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonConsultarEnvio.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonConsultarEnvioActionPerformed(evt);
      }
    });
    add(BotonConsultarEnvio);
    BotonConsultarEnvio.setBounds(580, 30, 50, 40);

    BotonCancelar.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/cancel.png"))); //
NOI18N
    BotonCancelar.setToolTipText("Cancelar");
BotonCancelar.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonCancelar.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonCancelarActionPerformed(evt);
      }
    });
    add(BotonCancelar);
    BotonCancelar.setBounds(680, 30, 50, 40);

    jLabel14.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel14.setText("Ruta");
    add(jLabel14);
    jLabel14.setBounds(20, 130, 50, 15);

    jTFRuta.setEditable(false);
    jTFRuta.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFRuta);
    jTFRuta.setBounds(140, 130, 180, 25);

    jLabel15.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel15.setText("Contenido");
    add(jLabel15);
```

```
jLabel15.setBounds(20, 400, 80, 15);

jTFContenido.setEditable(false);
jTFContenido.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFContenido);
jTFContenido.setBounds(120, 400, 240, 25);

jLabel16.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel16.setText("Bultos");
add(jLabel16);
jLabel16.setBounds(460, 400, 50, 15);

jTFBultos.setEditable(false);
jTFBultos.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFBultos);
jTFBultos.setBounds(510, 400, 60, 25);

jLabel17.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel17.setText("Kgs.");
add(jLabel17);
jLabel17.setBounds(690, 400, 40, 15);

jTFPeso.setEditable(false);
jTFPeso.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFPeso);
jTFPeso.setBounds(630, 400, 60, 25);

jLabel18.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel18.setText("Peso");
add(jLabel18);
jLabel18.setBounds(590, 400, 40, 15);

jLabel19.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel19.setText("Flete");
add(jLabel19);
jLabel19.setBounds(510, 480, 50, 15);

jTFFlete.setEditable(false);
jTFFlete.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFFlete);
jTFFlete.setBounds(560, 480, 60, 25);

jLabel20.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel20.setText("Seguro");
add(jLabel20);
jLabel20.setBounds(510, 510, 50, 15);

jTFSeguro.setEditable(false);
jTFSeguro.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFSeguro);
jTFSeguro.setBounds(560, 510, 60, 25);
```

```
jLabel21.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel21.setText("Total");
add(jLabel21);
jLabel21.setBounds(510, 540, 50, 15);

jTFTotal.setEditable(false);
jTFTotal.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFTotal);
jTFTotal.setBounds(560, 540, 60, 25);

jLabel25.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel25.setText("BsF");
add(jLabel25);
jLabel25.setBounds(630, 540, 30, 20);

jLabel26.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel26.setText("BsF");
add(jLabel26);
jLabel26.setBounds(630, 510, 30, 20);

jLabel27.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel27.setText("BsF");
add(jLabel27);
jLabel27.setBounds(200, 430, 30, 20);

jTFValorDeclarado.setEditable(false);
jTFValorDeclarado.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFValorDeclarado);
jTFValorDeclarado.setBounds(130, 430, 60, 25);

jLabel23.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel23.setText("Valor Declarado");
add(jLabel23);
jLabel23.setBounds(20, 430, 110, 15);

jLabel24.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel24.setText("BsF");
add(jLabel24);
jLabel24.setBounds(780, 490, 30, 15);

jLabel28.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel28.setText("BsF");
add(jLabel28);
jLabel28.setBounds(630, 480, 30, 20);

jLabel22.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel22.setText("E-mail");
add(jLabel22);
jLabel22.setBounds(450, 240, 60, 15);
```

```
    jTFEmailRemitente.setEditable(false);
    jTFEmailRemitente.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFEmailRemitente);
    jTFEmailRemitente.setBounds(510, 240, 230, 25);

    BotonLimpia.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonLimpia.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/editclear.png"))); //
NOI18N
    BotonLimpia.setToolTipText("Limpiar Formularios");
    BotonLimpia.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonLimpia.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonLimpiaActionPerformed(evt);
      }
    });
    add(BotonLimpia);
    BotonLimpia.setBounds(580, 70, 50, 40);
    add(jSeparator1);
    jSeparator1.setBounds(0, 160, 760, 10);
    add(jSeparator2);
    jSeparator2.setBounds(0, 270, 760, 10);
    add(jSeparator3);
    jSeparator3.setBounds(0, 380, 760, 10);

    jLabel30.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel30.setText("Tipo de Cobro");
    add(jLabel30);
    jLabel30.setBounds(450, 430, 100, 20);

    jTFEmailDestinatario.setEditable(false);
    jTFEmailDestinatario.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFEmailDestinatario);
    jTFEmailDestinatario.setBounds(510, 350, 230, 25);
  }// </editor-fold>//GEN-END:initComponents

  private void BotonConsultarEnvioActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonConsultarEnvioActionPerformed
    // TODO add your handling code here:
    BuscarEnvio();
  }//GEN-LAST:event_BotonConsultarEnvioActionPerformed

  @SuppressWarnings("static-access")
  private void BotonCancelarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonCancelarActionPerformed
    // TODO add your handling code here:
    Limpiar();
    BotonConsultarEnvio.setEnabled(true);
    this.setVisible(false);
    UIPrincipal.PanMen.ActivarAdministrador();
  }//GEN-LAST:event_BotonCancelarActionPerformed
```

```
  private void BotonLimpiaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonLimpiaActionPerformed
    // TODO add your handling code here:
    Limpiar();
    BotonConsultarEnvio.setEnabled(true);
}//GEN-LAST:event_BotonLimpiaActionPerformed

  private void jTFNumGuiaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTFNumGuiaActionPerformed
    // TODO add your handling code here:
    BuscarEnvio();
  }//GEN-LAST:event_jTFNumGuiaActionPerformed

  @SuppressWarnings("static-access")
  private void jTFNumGuiaKeyTyped(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_jTFNumGuiaKeyTyped
    // TODO add your handling code here:
    char caracter = evt.getKeyChar();
    if((caracter<'0'||caracter>'9')&&caracter!=evt.VK_BACK_SPACE)
      evt.consume();
  }//GEN-LAST:event_jTFNumGuiaKeyTyped

  public void BuscarEnvio(){
    if(jTFNumGuia.getText().equals(""))
      JOptionPane.showMessageDialog(null,"Ingrese NÃºmero de GuÃa");
    else{
      int b=procesar.ConsultarGuia(1);
      if (b==1){
      BotonConsultarEnvio.setEnabled(false);
      jTFNumGuia.setEditable(false);
      }
    }
  }

  public void Limpiar(){
    jTFNumGuia.setEditable(true);
    jTFNumGuia.setText("");
    jTFRuta.setText("");
    jTFCedRifRemitente.setText("");
    jTFNombreRemitente.setText("");
    jTFDireccionRemitente.setText("");
    jTFTelefonoRemitente.setText("");
    jTFEmailRemitente.setText("");
    jTFCedRifDestinatario.setText("");
    jTFNombreDestinatario.setText("");
    jTFDireccionDestinatario.setText("");
    jTFTelefonoDestinatario.setText("");
    jTFTipoCobro.setText("");
    jTFContenido.setText("");
    jTFValorDeclarado.setText("");
    jTFBultos.setText("");
    jTFPeso.setText("");
```

```
  jTFFlete.setText("");
  jTFSeguro.setText("");
  jTFTotal.setText("");
  jTAObservaciones.setText("");
  jTFFecha.setText("");
  jTFNumGuia.requestFocus();
}

// Variables declaration - do not modify//GEN-BEGIN:variables
private javax.swing.JButton BotonCancelar;
private javax.swing.JButton BotonConsultarEnvio;
private javax.swing.JButton BotonLimpia;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel10;
private javax.swing.JLabel jLabel11;
private javax.swing.JLabel jLabel12;
private javax.swing.JLabel jLabel13;
private javax.swing.JLabel jLabel14;
private javax.swing.JLabel jLabel15;
private javax.swing.JLabel jLabel16;
private javax.swing.JLabel jLabel17;
private javax.swing.JLabel jLabel18;
private javax.swing.JLabel jLabel19;
private javax.swing.JLabel jLabel20;
private javax.swing.JLabel jLabel21;
private javax.swing.JLabel jLabel22;
private javax.swing.JLabel jLabel23;
private javax.swing.JLabel jLabel24;
private javax.swing.JLabel jLabel25;
private javax.swing.JLabel jLabel26;
private javax.swing.JLabel jLabel27;
private javax.swing.JLabel jLabel28;
private javax.swing.JLabel jLabel29;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel30;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JLabel jLabel8;
private javax.swing.JLabel jLabel9;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JSeparator jSeparator1;
private javax.swing.JSeparator jSeparator2;
private javax.swing.JSeparator jSeparator3;
public static javax.swing.JTextArea jTAObservaciones;
public static javax.swing.JTextField jTFBultos;
public static javax.swing.JTextField jTFCedRifDestinatario;
public static javax.swing.JTextField jTFCedRifRemitente;
public static javax.swing.JTextField jTFContenido;
public static javax.swing.JTextField jTFDireccionDestinatario;
```

```
  public static javax.swing.JTextField jTFDireccionRemitente;
  public static javax.swing.JTextField jTFEmailDestinatario;
  public static javax.swing.JTextField jTFEmailRemitente;
  public static javax.swing.JTextField jTFFecha;
  public static javax.swing.JTextField jTFFlete;
  public static javax.swing.JTextField jTFNombreDestinatario;
  public static javax.swing.JTextField jTFNombreRemitente;
  public static javax.swing.JTextField jTFNumGuia;
  public static javax.swing.JTextField jTFPeso;
  public static javax.swing.JTextField jTFRuta;
  public static javax.swing.JTextField jTFSeguro;
  public static javax.swing.JTextField jTFTelefonoDestinatario;
  public static javax.swing.JTextField jTFTelefonoRemitente;
  public static javax.swing.JTextField jTFTipoCobro;
  public static javax.swing.JTextField jTFTotal;
  public static javax.swing.JTextField jTFValorDeclarado;
  // End of variables declaration//GEN-END:variables

  ProcesarDataEnv procesar = new ProcesarDataEnv();
}
```

5.1.4.2 Área Remesas

Durante el modelado de la aplicación se definió el área de remesas, que contempla las interfaces crear, modificar, eliminar y consultar. Esta vista está disponible para todos los usuarios del sistema.

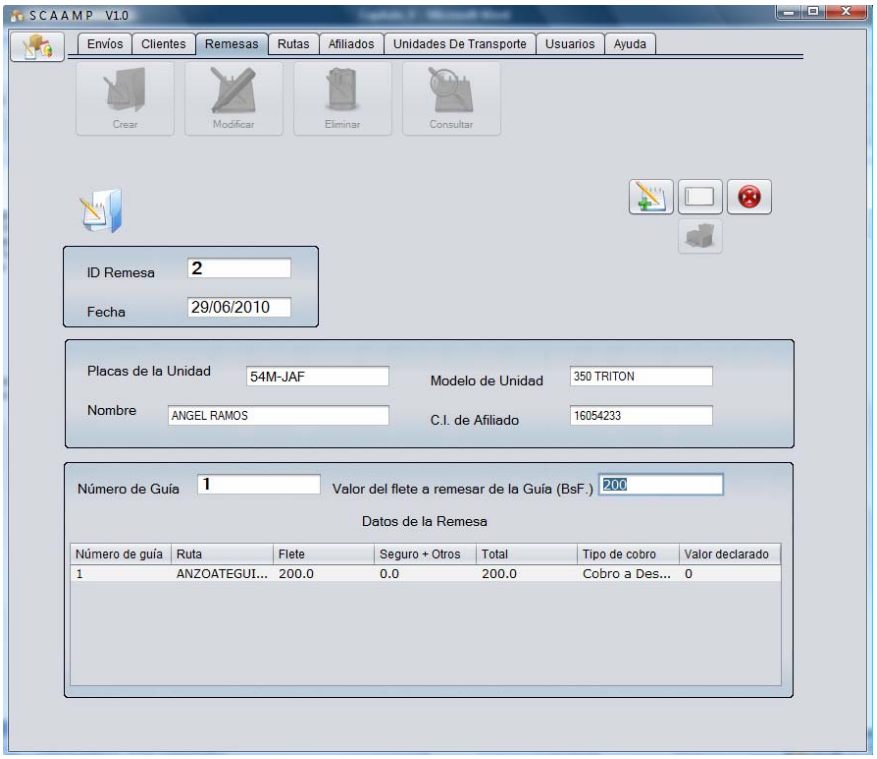

Figura 5.6. Pantalla Crear Remesa: Área Remesas

Fuente: Elaboración Propia

En la **Figura 5.6** se puede observa la interfaz crear remesa, en la cual el usuario debe ingresar la placa de la unidad para visualizar los datos del propietario de dicha unidad, posteriormente deberá ingresar los números de guía que desea asociar a dicha remesa.

Código Fuente Crear Remesa

```
package Remesa;

import UIScaamp.UIPrincipal;
import java.text.NumberFormat;
import java.util.Locale;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
import javax.swing.table.DefaultTableModel;

public class UIInsertarRemesa extends javax.swing.JPanel {
```

```
  /** Creates new form UIInsertarRemesa */
  public UIInsertarRemesa() {
    initComponents();

  }

  NumberFormat disp= NumberFormat.getCurrencyInstance();
  NumberFormat edit=NumberFormat.getNumberInstance(Locale.ENGLISH);

private void initComponents() {

    jLabel1 = new javax.swing.JLabel();
    jLabel3 = new javax.swing.JLabel();
    jTFIdRemesa = new javax.swing.JTextField();
    jLabel4 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jTFNombre = new javax.swing.JTextField();
    jLabel6 = new javax.swing.JLabel();
    jTFCedRif = new javax.swing.JTextField();
    jLabel7 = new javax.swing.JLabel();
    jTFNumGuia = new javax.swing.JTextField();
    jTFPlacas = new javax.swing.JTextField();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTDescripcion = new javax.swing.JTable();
    jLabel8 = new javax.swing.JLabel();
    jTFFecha = new javax.swing.JTextField();
    BotonGuardarNuevaRemesa = new javax.swing.JButton();
    BotonCancelarRemesa = new javax.swing.JButton();
    BotonLimpiarEnvio = new javax.swing.JButton();
    jLabel9 = new javax.swing.JLabel();
    jLabel10 = new javax.swing.JLabel();
    edit.setGroupingUsed(false);
    jTFFlete = new javax.swing.JFormattedTextField();

    setMinimumSize(new java.awt.Dimension(720, 480));
    setPreferredSize(new java.awt.Dimension(730, 480));
    setLayout(null);

    jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/crear_remesa.png"))); //
NOI18N
    add(jLabel1);
    jLabel1.setBounds(30, 80, 60, 50);

    jLabel3.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel3.setText("Placas de la Unidad");
    add(jLabel3);
    jLabel3.setBounds(20, 180, 130, 15);

    jTFIdRemesa.setEditable(false);
    jTFIdRemesa.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFIdRemesa);
```

```
jTFIdRemesa.setBounds(120, 130, 110, 25);

jLabel4.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel4.setText("ID Remesa");
add(jLabel4);
jLabel4.setBounds(20, 140, 70, 15);

jLabel5.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel5.setText("Nombre");
add(jLabel5);
jLabel5.setBounds(20, 220, 60, 15);

jTFNombre.setEditable(false);
jTFNombre.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFNombre);
jTFNombre.setBounds(90, 210, 230, 25);

jLabel6.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel6.setText("CI / RIF");
add(jLabel6);
jLabel6.setBounds(360, 220, 60, 15);

jTFCedRif.setEditable(false);
jTFCedRif.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
add(jTFCedRif);
jTFCedRif.setBounds(420, 210, 230, 25);

jLabel7.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel7.setText("NÃºmero de GuÃa");
add(jLabel7);
jLabel7.setBounds(20, 260, 110, 15);

jTFNumGuia.setEditable(false);
jTFNumGuia.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
jTFNumGuia.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    jTFNumGuiaActionPerformed(evt);
  }
});
jTFNumGuia.addKeyListener(new java.awt.event.KeyAdapter() {
  public void keyTyped(java.awt.event.KeyEvent evt) {
    jTFNumGuiaKeyTyped(evt);
  }
});
add(jTFNumGuia);
jTFNumGuia.setBounds(140, 250, 130, 25);

jTFPlacas.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
jTFPlacas.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(java.awt.event.ActionEvent evt) {
    jTFPlacasActionPerformed(evt);
```

```
      }
    });
    jTFPlacas.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        jTFPlacasKeyTyped(evt);
      }
    });
    add(jTFPlacas);
    jTFPlacas.setBounds(170, 170, 150, 25);

    jTDescripcion.setFont(new java.awt.Font("MS Reference Sans Serif", 0, 12));
    jTDescripcion.setModel(new javax.swing.table.DefaultTableModel(
      new Object [][] {

      },
      new String [] {
        "NÃºmero de guÃa", "Ruta", "Flete", "Seguro + Otros", "Total", "Tipo de cobro", "Valor declarado"
      }
    ) {
      boolean[] canEdit = new boolean [] {
        false, false, false, false, false, false, false
      };

      public boolean isCellEditable(int rowIndex, int columnIndex) {
        return canEdit [columnIndex];
      }
    });
    jTDescripcion.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyPressed(java.awt.event.KeyEvent evt) {
        jTDescripcionKeyPressed(evt);
      }
    });
    jScrollPane1.setViewportView(jTDescripcion);
    jTDescripcion.getTableHeader().setReorderingAllowed(false);

    add(jScrollPane1);
    jScrollPane1.setBounds(10, 320, 730, 150);

    jLabel8.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel8.setText("Fecha");
    add(jLabel8);
    jLabel8.setBounds(480, 140, 50, 15);

    jTFFecha.setEditable(false);
    jTFFecha.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFFecha);
    jTFFecha.setBounds(550, 130, 110, 25);

    BotonGuardarNuevaRemesa.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonGuardarNuevaRemesa.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Imagenes/add_remesa.png"))); // NOI18N
```

```
    BotonGuardarNuevaRemesa.setToolTipText("Guardar Nueva Remesa");
    BotonGuardarNuevaRemesa.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonGuardarNuevaRemesa.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonGuardarNuevaRemesaActionPerformed(evt);
      }
    });
    add(BotonGuardarNuevaRemesa);
    BotonGuardarNuevaRemesa.setBounds(580, 30, 50, 40);

    BotonCancelarRemesa.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/Imagenes/cancel.png"))); // NOI18N
    BotonCancelarRemesa.setToolTipText("Cancelar");
    BotonCancelarRemesa.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonCancelarRemesa.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonCancelarRemesaActionPerformed(evt);
      }
    });
    add(BotonCancelarRemesa);
    BotonCancelarRemesa.setBounds(680, 30, 50, 40);

    BotonLimpiarEnvio.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonLimpiarEnvio.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/editclear.png"))); //
NOI18N
    BotonLimpiarEnvio.setToolTipText("Limpiar Formularios");
    BotonLimpiarEnvio.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonLimpiarEnvio.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonLimpiarEnvioActionPerformed(evt);
      }
    });
    add(BotonLimpiarEnvio);
    BotonLimpiarEnvio.setBounds(630, 30, 50, 40);

    jLabel9.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel9.setText("Valor del flete a remesar de la GuÃa (BsF.)");
    add(jLabel9);
    jLabel9.setBounds(280, 260, 270, 15);

    jLabel10.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel10.setText("Datos de la Remesa");
    add(jLabel10);
    jLabel10.setBounds(310, 290, 130, 20);

    jTFFlete.setFormatterFactory(new javax.swing.text.DefaultFormatterFactory(new
javax.swing.text.NumberFormatter(disp),new javax.swing.text.NumberFormatter(edit),new
javax.swing.text.NumberFormatter(edit)));
    jTFFlete.setText("0");
    jTFFlete.setEnabled(false);
    jTFFlete.setValue(0);
```

```
    jTFFlete.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTFFleteActionPerformed(evt);
      }
    });
    jTFFlete.addFocusListener(new java.awt.event.FocusAdapter() {
      public void focusGained(java.awt.event.FocusEvent evt) {
        jTFFleteFocusGained(evt);
      }
    });
    jTFFlete.addKeyListener(new java.awt.event.KeyAdapter() {
      public void keyTyped(java.awt.event.KeyEvent evt) {
        jTFFleteKeyTyped(evt);
      }
    });
    add(jTFFlete);
    jTFFlete.setBounds(550, 250, 130, 25);
  }// </editor-fold>//GEN-END:initComponents

  private void BotonGuardarNuevaRemesaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonGuardarNuevaRemesaActionPerformed
    // TODO add your handling code here:
    if(jTFPlacas.getText().equals("")||jTDescripcion.getRowCount()==0){
      JOptionPane.showMessageDialog(null, "FALTAN DATOS POR COMPLETAR");
    }
    else{
    int salir = JOptionPane.showConfirmDialog(null,"Seguro Desea Guardar Los Datos?",
        "ADVERTENCIA",0);
    if(salir == 0){
      procesar.InsertarRemesa();
      LimpiarActualizar();
      jTFIdRemesa.setText(Integer.toString(procesar.NumRemesa()));
      jTFPlacas.requestFocus();
    }
    }
}//GEN-LAST:event_BotonGuardarNuevaRemesaActionPerformed

  private void BotonCancelarRemesaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonCancelarRemesaActionPerformed
    // TODO add your handling code here:
    LimpiarActualizar();
    this.setVisible(false);
    UIPrincipal.PanMen.ActivarAdministrador();
}//GEN-LAST:event_BotonCancelarRemesaActionPerformed
  private void BotonLimpiarEnvioActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonLimpiarEnvioActionPerformed
    // TODO add your handling code here:
    LimpiarActualizar();
}//GEN-LAST:event_BotonLimpiarEnvioActionPerformed
```

```
  private void jTFPlacasActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTFPlacasActionPerformed
    // TODO add your handling code here:
    jTFPlacas.setText(jTFPlacas.getText().toUpperCase());
    if(jTFPlacas.getText().equals(""))
      JOptionPane.showMessageDialog(null, "Ingrese placa del vehÃculo");
    else {
      int b=procesar.BuscarUTAfi();
      if (b==1) {
        jTFPlacas.setEditable(false);
        jTFNumGuia.setEditable(true);
        jTFNumGuia.requestFocus();
      }
    }
  }//GEN-LAST:event_jTFPlacasActionPerformed

  private void jTFNumGuiaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTFNumGuiaActionPerformed
    // TODO add your handling code here:
    if(jTFNumGuia.getText().equals(""))
      JOptionPane.showMessageDialog(null, "Debe ingresar nÃºmero de guÃa");
    else{
      int bb=0;
      for(int j=0;j<jTDescripcion.getRowCount();j++)
        if(jTFNumGuia.getText().equals(jTDescripcion.getValueAt(j, 0))){
          bb=1;
          JOptionPane.showMessageDialog(null, "GuÃa ya cargada en remesa");
          return;
        }
      if(bb==0){
      int b=procesar.BuscarGuia(1);
      if(b==1){
        jTFNumGuia.setEditable(false);
        jTFFlete.setEnabled(true);
        jTFFlete.requestFocus();
      }
      }
    }

  }//GEN-LAST:event_jTFNumGuiaActionPerformed

  private void jTDescripcionKeyPressed(java.awt.event.KeyEvent evt) {//GEN-
FIRST:event_jTDescripcionKeyPressed
    // TODO add your handling code here:
    if(evt.getKeyCode()==127){
      int fila=jTDescripcion.getSelectedRow();
      DefaultTableModel temp = (DefaultTableModel) jTDescripcion.getModel();
      temp.removeRow(fila);
      jTFFlete.setText("");
      jTFFlete.setEnabled(false);
      jTFNumGuia.setText("");
```

```
      jTFNumGuia.setEditable(true);
      jTFNumGuia.requestFocus();
    }
  }//GEN-LAST:event_jTDescripcionKeyPressed

  private void jTFNumGuiaKeyTyped(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_jTFNumGuiaKeyTyped
    // TODO add your handling code here:
    char caracter = evt.getKeyChar();
    if((caracter<'0'||caracter>'9')&&caracter!=evt.VK_BACK_SPACE)
      evt.consume();
  }//GEN-LAST:event_jTFNumGuiaKeyTyped

  private void jTFPlacasKeyTyped(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_jTFPlacasKeyTyped
    // TODO add your handling code here:
    if(jTFPlacas.getText().length()>=7)
      evt.consume();
  }//GEN-LAST:event_jTFPlacasKeyTyped

  private void jTFFleteActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTFFleteActionPerformed
    // TODO add your handling code here:
    int fila=jTDescripcion.getRowCount()-1;
    if(Double.valueOf(jTDescripcion.getValueAt(fila, 2).toString()).doubleValue()<
        Double.valueOf(jTFFlete.getText()).doubleValue()){
      JOptionPane.showMessageDialog(null, " El valor del flete a remesar no puede ser\n" +
          "mayor al flete de la guÃa. VERIFICAR MONTO.");
      return;
    }
    jTDescripcion.setValueAt(Double.valueOf(jTFFlete.getText()).doubleValue(), fila, 2);
    jTFNumGuia.setText("");
    jTFFlete.setText("");
    jTFFlete.setEnabled(false);
    jTFNumGuia.setEditable(true);
    jTFNumGuia.requestFocus();
    double seguro=Double.valueOf(jTDescripcion.getValueAt(fila, 4).toString()).doubleValue()
        -Double.valueOf(jTDescripcion.getValueAt(fila, 2).toString()).doubleValue();
    jTDescripcion.setValueAt(seguro, fila, 3);
}//GEN-LAST:event_jTFFleteActionPerformed

  private void jTFFleteFocusGained(java.awt.event.FocusEvent evt) {//GEN-FIRST:event_jTFFleteFocusGained
    // TODO add your handling code here:
    SwingUtilities.invokeLater(new Runnable(){
      public void run(){
        jTFFlete.selectAll();
      }
    });
}//GEN-LAST:event_jTFFleteFocusGained

  private void jTFFleteKeyTyped(java.awt.event.KeyEvent evt) {//GEN-FIRST:event_jTFFleteKeyTyped
    // TODO add your handling code here:
    char caracter = evt.getKeyChar();
```

```
    if((caracter<'0'||caracter>'9')&&caracter!=evt.VK_BACK_SPACE&&caracter!='.')
      evt.consume();
}//GEN-LAST:event_jTFFleteKeyTyped


  public void LimpiarActualizar(){
    jTFPlacas.setEditable(true);
    jTFNumGuia.setEditable(false);
    jTFPlacas.setText("");
    jTFNombre.setText("");
    jTFCedRif.setText("");
    jTFNumGuia.setText("");
    jTFFlete.setText("0");
    jTFFlete.requestFocus();
    jTFFlete.setEnabled(false);
    int filas=jTDescripcion.getRowCount();
    for(int j=0;j<filas;j++){
      DefaultTableModel temp = (DefaultTableModel) jTDescripcion.getModel();
      temp.removeRow(temp.getRowCount()-1);
    }
    jTFIdRemesa.setText(Integer.toString(procesar.NumRemesa()));
    jTFPlacas.requestFocus();
  }
  // Variables declaration - do not modify//GEN-BEGIN:variables
  private javax.swing.JButton BotonCancelarRemesa;
  private javax.swing.JButton BotonGuardarNuevaRemesa;
  private javax.swing.JButton BotonLimpiarEnvio;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel10;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JLabel jLabel4;
  private javax.swing.JLabel jLabel5;
  private javax.swing.JLabel jLabel6;
  private javax.swing.JLabel jLabel7;
  private javax.swing.JLabel jLabel8;
  private javax.swing.JLabel jLabel9;
  private javax.swing.JScrollPane jScrollPane1;
  public static javax.swing.JTable jTDescripcion;
  public static javax.swing.JTextField jTFCedRif;
  public static javax.swing.JTextField jTFFecha;
  public static javax.swing.JFormattedTextField jTFFlete;
  public static javax.swing.JTextField jTFIdRemesa;
  public static javax.swing.JTextField jTFNombre;
  public static javax.swing.JTextField jTFNumGuia;
  public static javax.swing.JTextField jTFPlacas;
  // End of variables declaration//GEN-END:variables

  ProcesarDataRem procesar = new ProcesarDataRem();
}
```

5.1.4.2 Área Afiliados

Durante el modelado de la aplicación se definió el área de afiliados, que contempla las interfaces ingresar, modificar, eliminar y consultar, así como también consultar, eliminar y realizar pagos a las cuentas asociadas a cada afiliado. Esta vista está disponible para el administrador del sistema.

En la **Figura 5.7** se puede observar la interfaz consultar cuenta, en la cual el administrador debe ingresar la CI del afiliado para visualizar los datos de su cuenta, y de esta manera visualizar también el total y tipo de deuda de la misma.

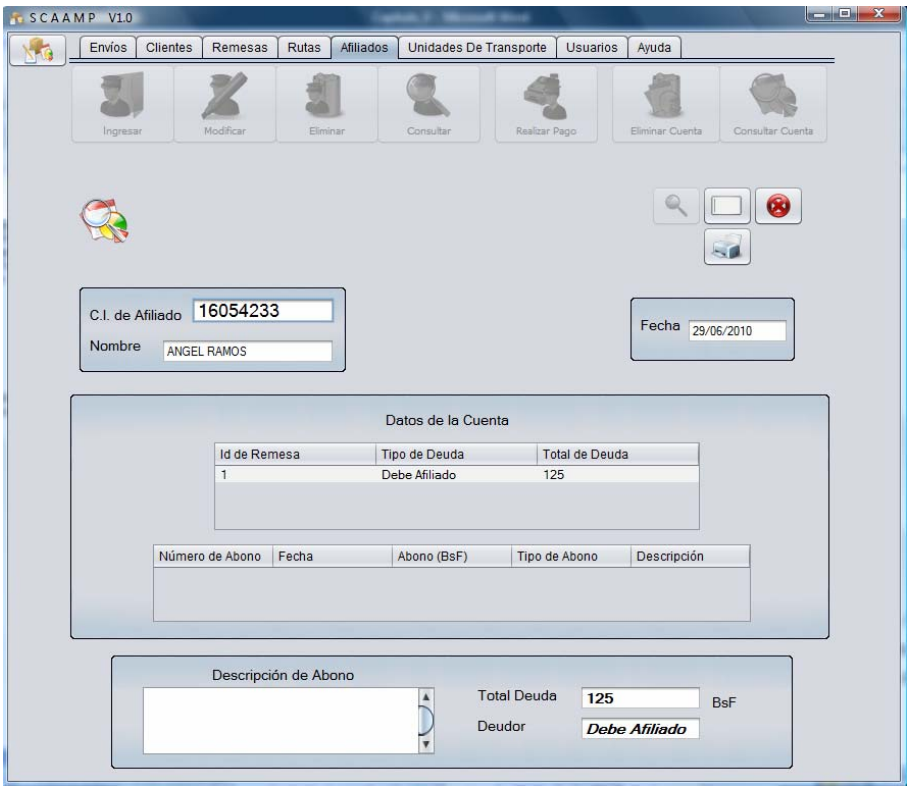

Figura 5.7. Pantalla Consultar Cuenta: Área Afiliados

Fuente: Elaboración Propia

Código Fuente Consultar Cuenta

```
package Afiliado;

import UIScaamp.UIPrincipal;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;

public class UIConsCtaAfiliado extends javax.swing.JPanel {

  /** Creates new form UIConsCtaAfiliado */
  public UIConsCtaAfiliado() {
    initComponents();
  }
private void initComponents() {
    jLabel1 = new javax.swing.JLabel();
    jTFNombreAfiliado = new javax.swing.JTextField();
    BotonCancelar = new javax.swing.JButton();
    jLabel4 = new javax.swing.JLabel();
    jTFCedRifAfiliado = new javax.swing.JTextField();
    jLabel8 = new javax.swing.JLabel();
    jLabel5 = new javax.swing.JLabel();
    jTFFecha = new javax.swing.JTextField();
    BotonConsultarCuenta = new javax.swing.JButton();
    jLabel6 = new javax.swing.JLabel();
    jScrollPane1 = new javax.swing.JScrollPane();
    jTDetallesAbono = new javax.swing.JTable();
    jScrollPane2 = new javax.swing.JScrollPane();
    jTDetallesCuenta = new javax.swing.JTable();
    jTFTotal = new javax.swing.JTextField();
    jTFDeudor = new javax.swing.JTextField();
    jLabel12 = new javax.swing.JLabel();
    jLabel13 = new javax.swing.JLabel();
    jLabel14 = new javax.swing.JLabel();
    BotonLimpiar = new javax.swing.JButton();

    setLayout(null);

    jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/buscar_cuenta.png"))); //
NOI18N
    add(jLabel1);
    jLabel1.setBounds(20, 70, 50, 50);

    jTFNombreAfiliado.setEditable(false);
    jTFNombreAfiliado.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFNombreAfiliado);
    jTFNombreAfiliado.setBounds(90, 170, 170, 25);
```

```
    BotonCancelar.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/cancel.png"))); //
NOI18N
    BotonCancelar.setToolTipText("Cancelar");
    BotonCancelar.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonCancelar.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonCancelarActionPerformed(evt);
      }
    });
    add(BotonCancelar);
    BotonCancelar.setBounds(560, 30, 50, 40);

    jLabel4.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel4.setText("Nombre");
    add(jLabel4);
    jLabel4.setBounds(20, 175, 60, 15);

    jTFCedRifAfiliado.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 18));
    jTFCedRifAfiliado.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        jTFCedRifAfiliadoActionPerformed(evt);
      }
    });
    add(jTFCedRifAfiliado);
    jTFCedRifAfiliado.setBounds(90, 135, 170, 25);

    jLabel8.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel8.setText("CI / RIF");
    add(jLabel8);
    jLabel8.setBounds(20, 140, 60, 15);

    jLabel5.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel5.setText("Fecha");
    add(jLabel5);
    jLabel5.setBounds(460, 140, 50, 15);

    jTFFecha.setEditable(false);
    jTFFecha.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    add(jTFFecha);
    jTFFecha.setBounds(520, 135, 80, 25);

    BotonConsultarCuenta.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonConsultarCuenta.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/search.png")));
// NOI18N
    BotonConsultarCuenta.setToolTipText("Consultar Cuenta");
    BotonConsultarCuenta.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonConsultarCuenta.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonConsultarCuentaActionPerformed(evt);
      }
```

```
});
add(BotonConsultarCuenta);
BotonConsultarCuenta.setBounds(460, 30, 50, 40);
jLabel6.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
jLabel6.setText("Datos de la Cuenta");
add(jLabel6);
jLabel6.setBounds(250, 210, 130, 20);

jTDetallesAbono.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 12)); // NOI18N
jTDetallesAbono.setModel(new javax.swing.table.DefaultTableModel(
  new Object [][] {

  },
  new String [] {
    "NÃºmero de Abono", "Fecha", "Abono (BsF)", "Tipo de Abono", "DescripciÃ³n"
  }
) {
  boolean[] canEdit = new boolean [] {
    false, false, false, false, false
  };

  public boolean isCellEditable(int rowIndex, int columnIndex) {
    return canEdit [columnIndex];
  }
});
jTDetallesAbono.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
jScrollPane1.setViewportView(jTDetallesAbono);
jTDetallesAbono.getTableHeader().setReorderingAllowed(false);

add(jScrollPane1);
jScrollPane1.setBounds(20, 340, 590, 80);

jTDetallesCuenta.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 12)); // NOI18N
jTDetallesCuenta.setModel(new javax.swing.table.DefaultTableModel(
  new Object [][] {

  },
  new String [] {
    "Id de Remesa", "Tipo de Deuda", "Total de Deuda"
  }
) {
  boolean[] canEdit = new boolean [] {
    false, false, false
  };

  public boolean isCellEditable(int rowIndex, int columnIndex) {
    return canEdit [columnIndex];
  }
});
jScrollPane2.setViewportView(jTDetallesCuenta);
jTDetallesCuenta.getTableHeader().setReorderingAllowed(false);
```

```
    add(jScrollPane2);
    jScrollPane2.setBounds(80, 240, 480, 90);

    jTFTotal.setEditable(false);
    jTFTotal.setFont(new java.awt.Font("Microsoft Sans Serif", 1, 14));
    add(jTFTotal);
    jTFTotal.setBounds(310, 440, 120, 25);

    jTFDeudor.setEditable(false);
    jTFDeudor.setFont(new java.awt.Font("Microsoft Sans Serif", 3, 14));
    add(jTFDeudor);
    jTFDeudor.setBounds(310, 470, 120, 25);

    jLabel12.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel12.setText("Deudor");
    add(jLabel12);
    jLabel12.setBounds(210, 470, 90, 20);

    jLabel13.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel13.setText("BsF");
    add(jLabel13);
    jLabel13.setBounds(440, 450, 30, 15);

    jLabel14.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 14));
    jLabel14.setText("Total Deuda");
    add(jLabel14);
    jLabel14.setBounds(210, 440, 90, 20);

    BotonLimpiar.setFont(new java.awt.Font("Microsoft Sans Serif", 0, 11));
    BotonLimpiar.setIcon(new javax.swing.ImageIcon(getClass().getResource("/Imagenes/editclear.png"))); //
NOI18N
    BotonLimpiar.setToolTipText("Limpiar Formularios");
    BotonLimpiar.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
    BotonLimpiar.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
        BotonLimpiarActionPerformed(evt);
      }
    });
    add(BotonLimpiar);
    BotonLimpiar.setBounds(510, 30, 50, 40);
  }// </editor-fold>//GEN-END:initComponents

  private void BotonCancelarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonCancelarActionPerformed
    // TODO add your handling code here:
    Limpiar();
    UIPrincipal.PanMen.ActivarAdministrador();
    this.setVisible(false);

}//GEN-LAST:event_BotonCancelarActionPerformed
```

```
  private void BotonConsultarCuentaActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonConsultarCuentaActionPerformed
    // TODO add your handling code here:
    RealizarConsulta();
}//GEN-LAST:event_BotonConsultarCuentaActionPerformed

  private void BotonLimpiarActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_BotonLimpiarActionPerformed
    // TODO add your handling code here:
    Limpiar();
}//GEN-LAST:event_BotonLimpiarActionPerformed

  private void jTFCedRifAfiliadoActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_jTFCedRifAfiliadoActionPerformed
    // TODO add your handling code here:
    RealizarConsulta();
  }//GEN-LAST:event_jTFCedRifAfiliadoActionPerformed

  public void RealizarConsulta(){
    jTFCedRifAfiliado.setText(jTFCedRifAfiliado.getText().toUpperCase());
    if(jTFCedRifAfiliado.getText().equals(""))
      JOptionPane.showMessageDialog(null, "Ingrese Cedula o RIF del Afiliado");
    else{
      int b=procesar.ConsultarCuenta(1);
      if (b==1){
        DateFormat FormFecha= new SimpleDateFormat("dd/MM/yyyy");
        Date Fecha= new Date();
        jTFFecha.setText(FormFecha.format(Fecha));
        BotonConsultarCuenta.setEnabled(false);
        jTFCedRifAfiliado.setEditable(false);
        procesar.TotalizarCuenta(1);
      }
    }
  }

  public void Limpiar(){
    jTFCedRifAfiliado.setText("");
    jTFNombreAfiliado.setText("");
    jTFFecha.setText("");
    jTFTotal.setText("");
    jTFDeudor.setText("");
    DefaultTableModel tempCuenta = (DefaultTableModel) jTDetallesCuenta.getModel();
    DefaultTableModel tempAbono = (DefaultTableModel) jTDetallesAbono.getModel();
    int filas=jTDetallesCuenta.getRowCount();
    for(int i=0;i<filas;i++)
      tempCuenta.removeRow(jTDetallesCuenta.getRowCount()-1);
    filas=jTDetallesAbono.getRowCount();
    for(int i=0;i<filas;i++)
      tempAbono.removeRow(jTDetallesAbono.getRowCount()-1);
    BotonConsultarCuenta.setEnabled(true);
    jTFCedRifAfiliado.setEditable(true);
```

```
  jTFCedRifAfiliado.requestFocus();
 }

 // Variables declaration - do not modify//GEN-BEGIN:variables
 private javax.swing.JButton BotonCancelar;
 private javax.swing.JButton BotonConsultarCuenta;
 private javax.swing.JButton BotonLimpiar;
 private javax.swing.JLabel jLabel1;
 private javax.swing.JLabel jLabel12;
 private javax.swing.JLabel jLabel13;
 private javax.swing.JLabel jLabel14;
 private javax.swing.JLabel jLabel4;
 private javax.swing.JLabel jLabel5;
 private javax.swing.JLabel jLabel6;
 private javax.swing.JLabel jLabel8;
 private javax.swing.JScrollPane jScrollPane1;
 private javax.swing.JScrollPane jScrollPane2;
 public static javax.swing.JTable jTDetallesAbono;
 public static javax.swing.JTable jTDetallesCuenta;
 public static javax.swing.JTextField jTFCedRifAfiliado;
 public static javax.swing.JTextField jTFDeudor;
 public static javax.swing.JTextField jTFFecha;
 public static javax.swing.JTextField jTFNombreAfiliado;
 public static javax.swing.JTextField jTFTotal;
 // End of variables declaration//GEN-END:variables

 ProcesarDataAfi procesar = new ProcesarDataAfi();
}
```

5.2 FLUJO DE TRABAJO DE PRUEBAS

El flujo de trabajo Pruebas, es una de las fases más importantes del ciclo de vida de desarrollo de software. Un producto de software que se desarrolla se debe entregar al cliente libre de defectos o de errores. La prueba es el proceso de ejecutar la aplicación con la intención de descubrir defectos en el programa.

En el ciclo de vida de desarrollo de la aplicación, la fase de prueba ocurre después de la fase de programación. En tal sentido se plantea la revisión del funcionamiento de los distintos módulos de la aplicación, sometiendo cada uno a una entrada, proceso y salida esperada, a fin de

verificar la confiabilidad, seguridad e integridad del sistema. Las pruebas garantizan que el producto sea de calidad.

5.2.1 Casos de Pruebas

Un caso de prueba representa una forma de probar el sistema, incluyendo la entrada resultado con la que se ha de probar y las condiciones bajo las cuales ha de probarse. Existen dos casos de prueba comunes, prueba de “caja negra” y prueba de “caja blanca”.

Un caso de prueba específica como probar un caso de uso o un escenario específico de un caso de uso. Un caso de prueba de este tipo incluye la verificación del resultado de la interacción entre los actores y el sistema, conocido como prueba de “caja negra”.

5.2.2 Modelo de pruebas

Un modelo de prueba describe como se evalúan los componentes ejecutables del sistema y se realiza con la finalidad de encontrar errores a tiempo para su corrección. Para ello se implementarán un tipo de prueba muy común, la cual representa un caso de prueba que detalla cómo probar un caso de uso o un escenario específico y que incluye la verificación del resultado de la interacción entre actores y el sistema, representa el caso de prueba del sistema conocido como “caja negra”. Es importante especificar que esta prueba posee dos tipos: caja negra en lo pequeño y caja negra en lo grande.

5.2.3 Pruebas por Unidad

Las pruebas por unidad se emplearon mediante la aplicación de la prueba de la caja negra sobre los diversos componentes del sistema. Para realizar este tipo de pruebas, se identifican un conjunto de valores que pueden ser introducidos por un actor, y se expresan como clases de equivalencia para poder abarcar la totalidad de las ocurrencias de un evento de inserción de datos.

A continuación en la **Tabla 5.1**, se representan las clases de equivalencia del componente UIInsertarCliente.java, el mismo se encarga de procesar los datos para registrar un nuevo cliente en el sistema. (Ver **Figura 5.9**).

Tabla 5.1. Clase de equivalencia para el componente UIInsertarCliente.java (1/2).

Nº	Dato	Clase de Equivalencia	Válido	Inválido
1	Nombre	Carácter numérico		X
2	Nombre	Carácter alfanumérico	X	
3	Nombre	Longitud carácter <= 40	X	
4	Nombre	Dato nulo		X
5	Dirección	Carácter numérico		X
6	Dirección	Carácter alfanumérico	X	
7	Dirección	Longitud carácter <= 100	X	
8	Dirección	Dato nulo		X
9	CI / RIF	Carácter numérico	X	
10	CI / RIF	Carácter alfanumérico		X
11	CI / RIF	Longitud carácter < = 13	X	
12	CI / RIF	Dato nulo		X
13	Teléfono Habitación	Carácter numérico	X	
14	Teléfono Habitación	Carácter alfanumérico		X
15	Teléfono Habitación	Longitud carácter = 13	X	

Tabla 5.1. Clase de equivalencia para el componente UIInsertarCliente.java (2/2).

Nº	Dato	Clase de Equivalencia	Válido	Inválido
16	Teléfono Habitación	Dato nulo	X	

17	Teléfono Celular	Carácter numérico	X	
18	Teléfono Celular	Carácter alfanumérico		X
19	Teléfono Celular	Longitud carácter = 13	X	
20	Teléfono Celular	Dato nulo	X	
21	E-mail	Carácter numérico		X
22	E-mail	Carácter alfanumérico	X	
23	E-mail	Longitud carácter <= 30	X	
24	E-mail	Dato nulo	X	

Fuente: Elaboración Propia

En la **Tabla 5.2** se presenta algunos casos de prueba de caja negra, donde se incluirán datos que serán cotejados por las clases de equivalencia referenciadas anteriormente. Si se cumplen todas las clases involucra con dicho dato, entonces la salida será validada (*ver* **Figura 5.10**).

Tabla 5.2. Casos de prueba de caja negra para el componente UIInsertarCliente.java (1/2).

Dato	Caso de Prueba	Salida	Clase Cubierta
Nombre	15612	Inválido	1
Nombre	Inversiones 1234	Válido	2,3
Nombre	Dato nulo	Inválido	4
Dirección	123123	Inválido	5
Dirección	Av. Alterna # 145 Barcelona	Válido	6,7
Dirección	Dato nulo	Inválido	8
CI / RIF	15432456	Válido	9,11
CI / RIF	Vfe123123	Inválido	10
CI / RIF	Dato nulo	Inválido	12
Teléfono Habitación	02812634512	Válido	13,15
Teléfono Habitación	Rewe1231	Inválido	14
Teléfono Habitación	Dato nulo	Válido	16
Teléfono Celular	04149876543	Válido	17,19
Teléfono Celular	Asda1212	Inválido	18

Tabla 5.2. Casos de prueba de caja negra para el componente UIInsertarCliente.java (2/2).

Dato	Caso de Prueba	Salida	Clase Cubierta
Teléfono Celular	Dato nulo	Válido	20

E-mail	12313	Inválido	21
E-mail	migroa18@hotmail.com	Válido	22,23
E-mail	Dato nulo	Válido	24

Fuente: Elaboración Propia

5.2.4 Pruebas de Integración

El objetivo general de las pruebas de integración es, detectar las fallas de interacción entre las distintas clases que componen al sistema. Debido a que cada clase probada por separado se inserta de manera progresiva dentro de la estructura, las pruebas de integración son realmente un mecanismo para comprobar el correcto ensamblaje del sistema completo. Al efectuar la integración de los módulos, se concentra el esfuerzo en la búsqueda de fallas que puedan provocar excepciones arrojadas por los métodos; el empleo de operaciones equivocada, e invocación inadecuada de los métodos.

Luego de verificar la calidad de los componentes, se procede a comprobar la eficiencia de un conjunto de componentes integrados por fase.

A continuación en la **Figura 5.8** se pueden observar resaltadas todas las fases, en donde se despliega el diagrama de componentes por fase de integración del sistema.

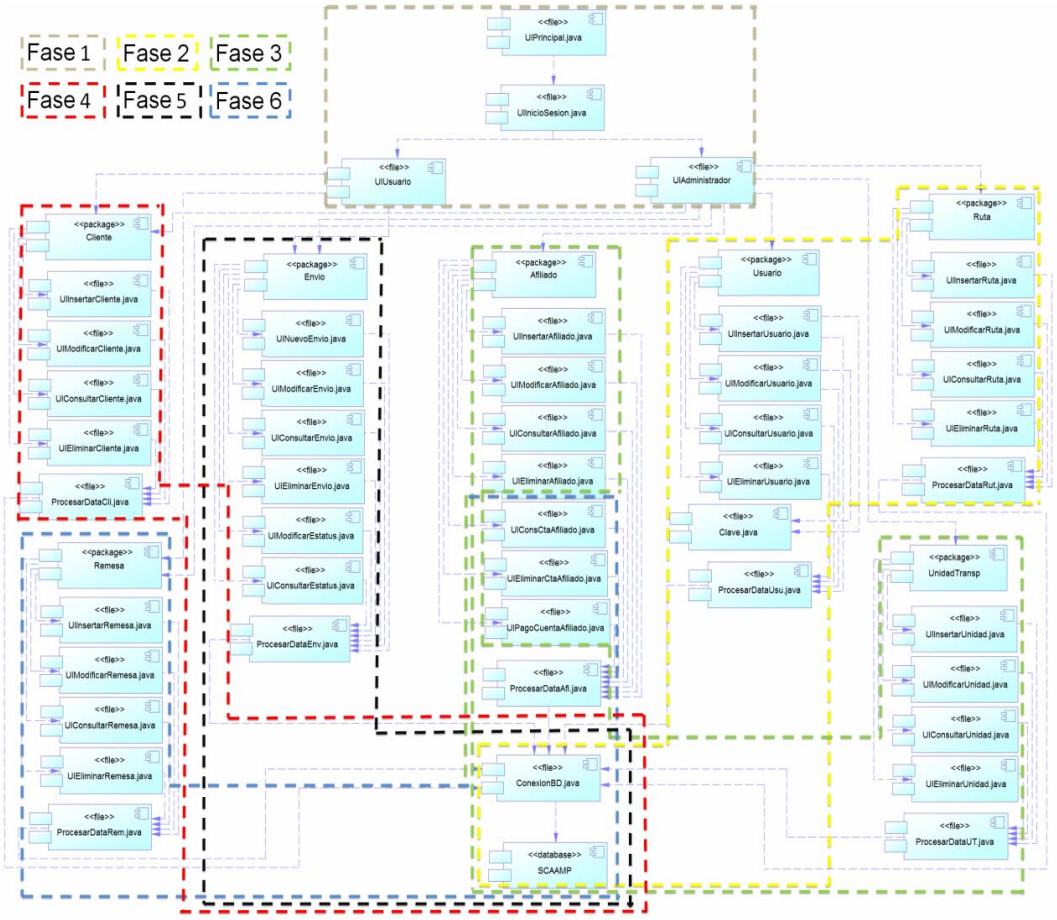

Figura 5.8 Diagrama de Componentes Total por Fase de Integración de SCAAMP

Fuente: Elaboración Propia

5.2.4.1 Prueba de Integración de Insertar Cliente

Este caso de prueba verifica la funcionalidad de la implementación de la fase resaltada con el color rojo de la **Figura 5.8.**

Entrada:

Tabla 5.3. Datos de Entrada para Probar la Integración de Insertar Cliente (1/2).

Datos			
Nombre	Daniel Hernández	CI / Rif	16987123

Tabla 5.3. Datos de Entrada para Probar la Integración de Insertar Cliente (2/2).

Datos			
Dirección	Conj. Resd. Bosquemar	Teléfono	02813272488
E-mail	danielhernz@hotmail.com	Celular	04146542341

Fuente: Elaboración Propia

Resultado:

Obtención de un mensaje con la confirmación del ingreso satisfactorio de los datos del nuevo cliente.

Procedimiento de Prueba:

- Activar la interfaz de usuario administrador.
- Acceder a la interfaz de Clientes.
- Hacer click en Ingresar, para acceder al formulario destinado al ingreso de nuevos clientes.
- Introducir los datos de la **Tabla 5.3** y hacer click en el botón Guardar Nuevo Cliente, si los datos son correctos hacer click en Si, en el mensaje “Desea Guardar los Datos”. (*ver* **Figura 5.9**).
- A continuación se despliega un mensaje para informar que los datos han sido almacenados correctamente (*ver* **Figura 5.10**).
- Se realiza la comunicación con la base de datos a través del componente conexiónBD y se observa que el cliente ha sido ingresado de manera correctamente. (*ver* **Figura 5.11**).
- Luego se presiona el botón de cancelar para cerrar la interfaz de ingresar cliente y de esta manera poder ingresar en otra.
- A través del botón Consultar se accede a la ventana en la cual se pueden observar los datos que se han almacenado, esto para modificar en caso de ser necesario (*ver* **Figura 5.12**).

- Se activa el gestor de procesar cliente el cual se comunica con la base de datos a través del componente conexiónBD para realizar la consulta. (*ver* **Figura 5.13**)

A continuación se presenta una secuencia de imágenes que muestran el procedimiento de integración:

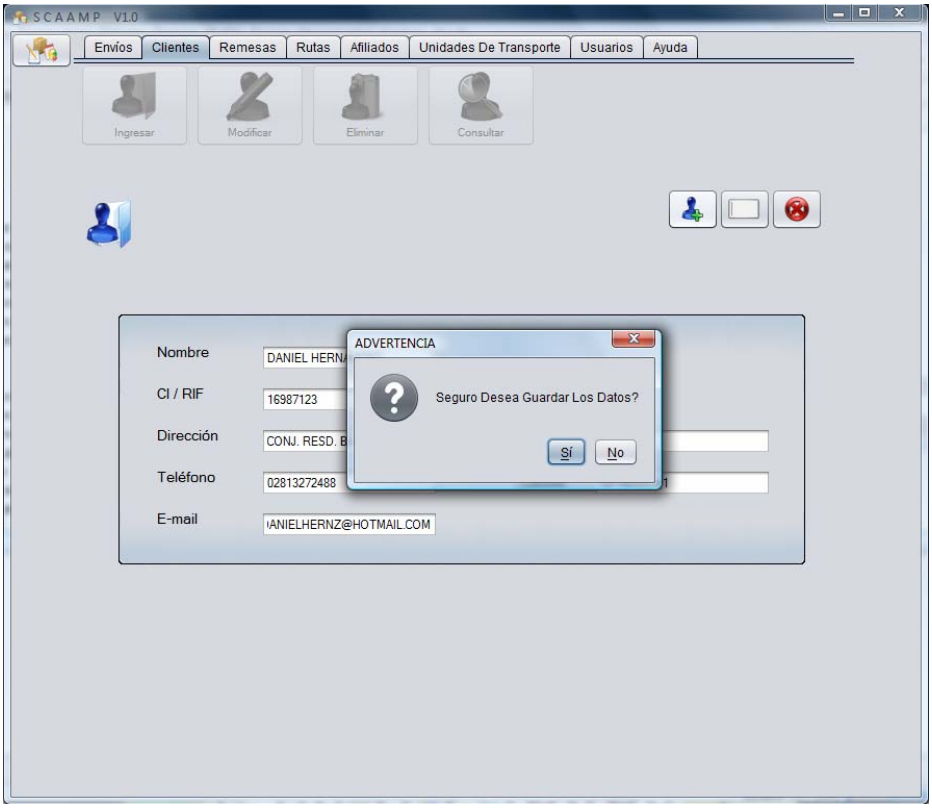

Figura 5.9. Introducción de los Datos en la Ventana Ingresar Cliente

Fuente: Elaboración Propia

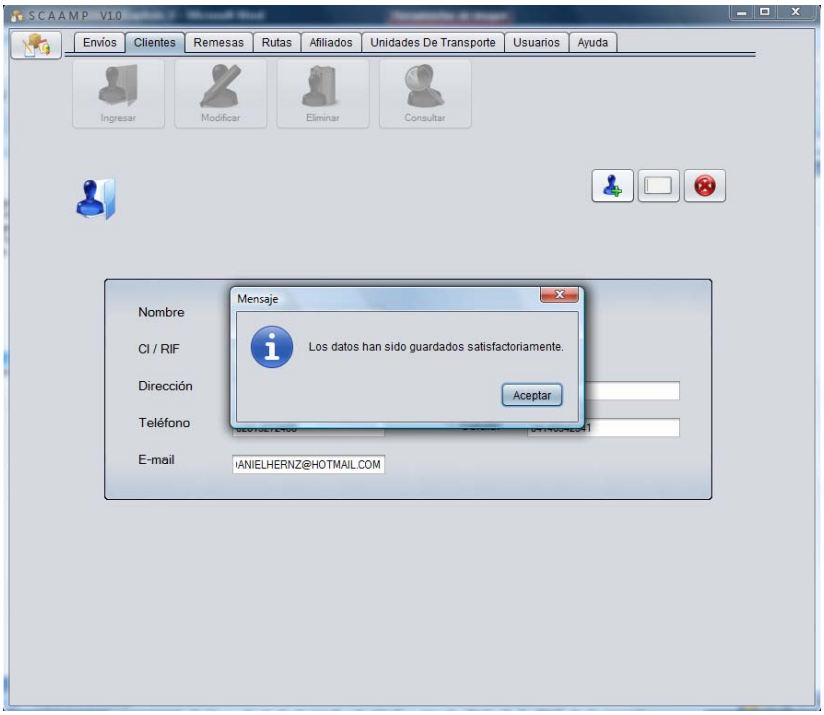

Figura 5.10. Confirmación del Almacenamiento Satisfactorio de los Datos del Cliente

Fuente: Elaboración Propia

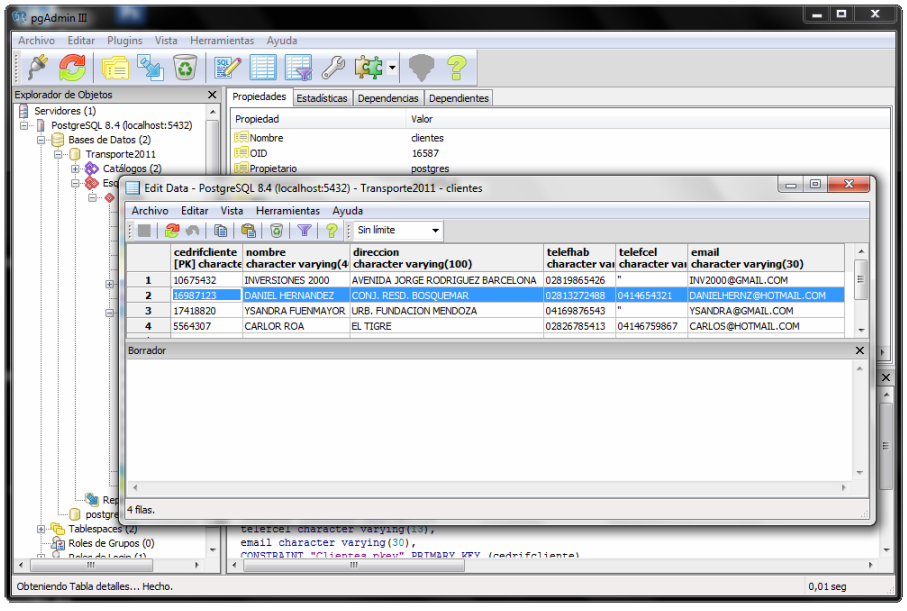

Figura 5.11. Detalles en la Base de Datos del Cliente Ingresado Correctamente.

Fuente: Elaboración Propia

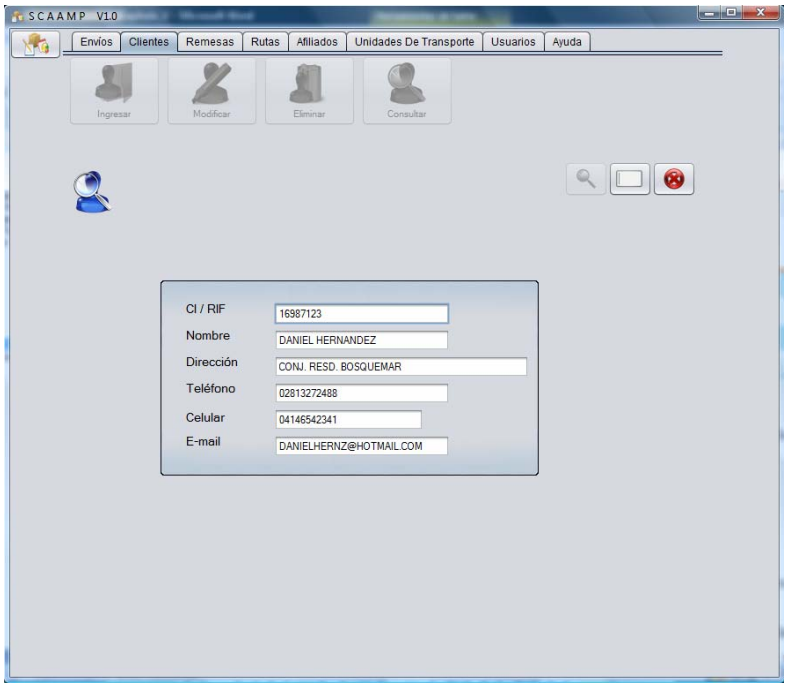

Figura 5.12. Interfaz para Consultar Clientes.

Fuente: Elaboración Propia

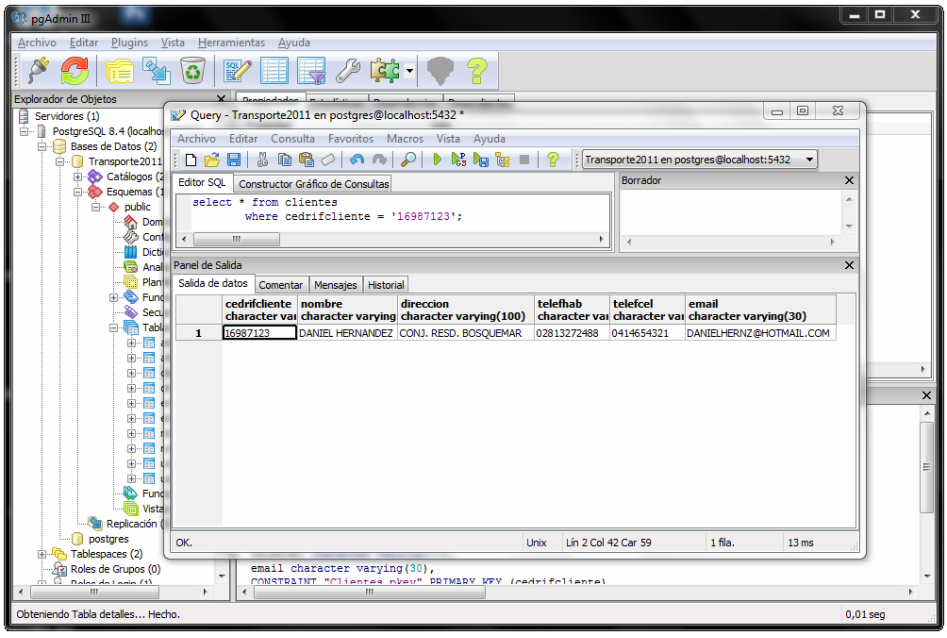

Figura 5.13. Detalles en la Base de Datos del Cliente Consultado.

Fuente: Elaboración Propia

5.3 EVALUACIÓN DE LA FASE DE CONSTRUCCIÓN

Durante la fase de construcción se complementó el diseño de la arquitectura base de SCAAMP, se codificaron e integraron todos los componentes del sistema.

Las pruebas de unidad se aplicaron mediante el método de caja negra, adicionalmente se aplicaron pruebas de integración lo cual permitió detectar y corregir fallas en el funcionamiento de los componentes y en la forma de acoplarse unos a otros.

Durante la construcción, se amortizaron los riesgos correspondientes a esta fase, logrando el propósito de tener lista la primera versión operativa del sistema.

CONCLUSIONES

- Mediante este proyecto de investigación se creó una aplicación para el control administrativo de socios y monitoreo de servicios, a través de una interfaz gráfica sencilla que facilitará la introducción, búsqueda y actualización de información, lo cual optimizará el tiempo de respuesta por parte del personal de una empresa de encomiendas.

- El RUP, (*Rational Unified Process*), como metodología elegida, aunado al uso de UML como herramienta principal para la documentación y guía, permitieron la organización y desarrollo del sistema, cumpliendo con todas las etapas de este proceso y ayudó a la prevención de fallas que se pudieron presentar, evitando así un replanteamiento del proyecto, canalizando los diferentes flujos de trabajo necesarios por cada una de las fases de desarrollo.

- La fase de inicio del proyecto jugó un papel fundamental, puesto que permitió obtener una noción clara del contexto del sistema, a través del Modelo de Dominio y el diagrama de Casos de Uso, logrando con ello la obtención de los requisitos necesarios para llevar a cabo el diseño. Los casos de uso también fueron de gran importancia para representar las condiciones y requerimientos del sistema, logrando esto a través de una buena documentación y análisis.

- La fase de elaboración permitió eliminar la mayor cantidad de riesgos posibles empezando por los más altos o de más relevancia. Los requisitos funcionales y no funcionales además de toda información obtenida en las etapas anteriores, fueron refinados en esta fase, y adecuados a las posibilidades de la herramienta de desarrollo que se

uso. También se definieron los flujos de trabajo en diseño y organización del sistema, representados por los diagramas correspondientes.

- El diseño previo de la arquitectura del sistema facilitó la visualización del entorno con el cual se relaciona el sistema, además proporcionó una herramienta con la cual se lograba apreciar de mejor forma los posibles errores o inconvenientes a solucionar, esto contribuye a una etapa de construcción más limpia y rápida.

- La base de datos del sistema se diseñó considerando el criterio de escalabilidad, a fin de facilitar la implementación de posibles mejoras a la aplicación, en versiones posteriores.

- La escogencia de Java como lenguaje de programación permitió hacer una limpia codificación del sistema ya que en las etapas anteriores se planteo una sólida estructura de clases en conformidad con el paradigma orientado a objetos y Java es un lenguaje de programación orientado a objetos de última generación y figura como uno de los lenguajes con un mayor crecimiento y amplitud. Por otra parte Java es netamente orientado a aplicaciones gráficas y posee potentes herramientas para el desarrollo de GUI’s, lo que facilitó la construcción de Interfaces Gráficas organizadas, comunicativas y muy amigables.

- Se realizó de manera exitosa la integración y pruebas del sistema, se evaluó y depuró el sistema en varias iteraciones, obteniendo un software altamente funcional, al que puede realizarse mantenimiento y actualizaciones.

RECOMENDACIONES

- Efectuar un mantenimiento periódico a la base de datos, realizando respaldos, para así garantizar el correcto funcionamiento del sistema, optimizar los tiempos de respuesta y mantener la información actualizada.

- Se recomienda que, en caso de realizarse la actualización de SCAAMP (una versión 2.0), inspeccionar el código de los componentes y, de ser posible, optimizar los mismos.

- Mantener actualizados los documentos de diseño, con respecto a cualquier modificación o actualización que se realice al sistema y llevar un registro de éstos.

- Incentivar el desarrollo e implementación de sistemas, para la automatización de las demás dependencias, que permitan incrementar la eficiencia de sus operaciones y minimizar los tiempos de respuestas en sus actividades diarias.

BIBLIOGRAFÍA

- Arcila, A. y Zacarías, R. **(2006). "Desarrollo de un Software como Soporte para la Automatización de las Actividades Administrativas que se llevan a cabo en una Institución Educativa".** Trabajo de Grado de Ingeniería de Computación, Universidad de Oriente. Barcelona, Venezuela.

- Arranz, L. **(2001)**. **Guía de Encomienda.** http://www.sappiens.com/castellano/glosario.nsf/Comercio_Exterior/Gu%C3%ADa_de_encomienda/79858565334E7393002569E50045144C!opendocument. España.

- Howard, D. y Freeman, M. **(2009). Medios de Transporte, Definición e Historia.** http://www.medios.us/transporte/social/medios-de-transporte-definicion-e-historia/. USA.

- Larman, C. **(2003). "UML Y PATRONES. Una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado".** Segunda edición, Pearson Education, S.A. Madrid, España.

- Lugo, E. y Pino, M. **(2009). "Desarrollo de un Software Gerencial para los Procesos Administrativos del Área de Postgrado del Núcleo de Anzoátegui de la Universidad de Oriente".** Trabajo de Grado de Ingeniería de Computación, Universidad de Oriente. Barcelona, Venezuela.

- Marín, J. **(2009)**.**Transporte.** http://www.xuletas.es/ficha/clasificacio-n-del-transporte/. Cartagena, Colombia.

- Myers, B. **(1996)**. "**User Interface Software Technology".** ACM New York, USA.

- Pacheco, R. **(2008). "Desarrollo de una aplicación para automatizar los módulos de compra y venta de un sistema administrativo para una empresa bajo la licencia de software libre".** Trabajo de Grado de Ingeniería de Computación, Universidad de Oriente. Barcelona, Venezuela.

- Pérez, L. **(2003). "Desarrollo de un Software para Automatizar las Actividades de Control Interno de una Empresa de Transporte de Alimentos".** Trabajo de Grado de Ingeniería de Computación, Universidad de Oriente. Barcelona, Venezuela.

- Pressman, R. **(1993)**. **"Ingeniería de Software un Enfoque Práctico".** Tercera Edición, Editorial Mc Graw Hill/Interamericana de España. Madrid, España.

- Salazar, F. **(2007). "Desarrollo de un Software para la Automatización del Control de Entrada y Salida de Bienes, Equipos y Materiales en una Empresa Cervecera".** Trabajo de Grado de Ingeniería de Computación, Universidad de Oriente. Barcelona, Venezuela.

- Sánchez, J. **(2009)**. **Definición Automatización**. http://www.mitecnologico.com/Main/DefinicionAutomatizacion. México.

- Tabare, M. **(2006). "Desarrollo de un Software que Permite la Adquisición y Procesamiento de Datos Asociados a la Producción de Barras de Acero en una Empresa Siderúrgica".** Trabajo de Grado de Ingeniería de Computación, Universidad de Oriente. Barcelona, Venezuela.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

TÍTULO	**DESARROLLO DE UNA APLICACIÓN PARA LA AUTOMATIZACIÓN DEL CONTROL ADMINISTRATIVO DE AFILIADOS Y MONITOREO DE SERVICIOS PARA UNA EMPRESA DE ENCOMIENDAS PRIVADA.**
SUBTÍTULO	

AUTOR (ES):

APELLIDOS Y NOMBRES	CÓDIGO CULAC / E MAIL
Ramos A., Angel L.	**CVLAC: V-16.054.233** **E MAIL: alramos@cantv.net**
Roa Q., Miguel A.	**CVLAC: V-16.019.387** **E MAIL: migroa18@hotmail.com**
	CVLAC: **E MAIL:**
	CVLAC: **E MAIL:**

PALÁBRAS O FRASES CLAVES:

Automatización, Aplicación, Control Administrativo, Transporte, Encomiendas, UML.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ÀREA	SUBÀREA
Computación	Desarrollo de Software
	Base de datos

RESUMEN (ABSTRACT):

Las empresas de encomiendas se dedican a prestar los servicios de encomiendas y transporte de paquetería. En un mercado tan competitivo como el del correo privado, donde coexisten tantas firmas dedicadas a la misma actividad, tienen que saber ganarse el gusto y preferencia de los clientes gracias a la filosofía de calidad, eficiencia y rapidez, que envuelve cada uno de los procesos operativos de la marca, la garantía de entrega y el equipo humano. Atendiendo a esta problemática se propuso el desarrollo de un software para la automatización del control administrativo de los afiliados y el monitoreo de paquetes. En este informe se presenta el diseño de un Sistema de Control Administrativo de Afiliados y Monitoreo de Paquetes (*SCAAMP*) que apoyará los procesos descritos anteriormente. Este es un proyecto basado en el Proceso Unificado de desarrollo de software, bajo el Lenguaje Unificado de Modelado (*UML*), para ser implantado bajo cualquier plataforma, programado con el lenguaje de programación Java y cuyos datos son almacenados en una base de datos PostgreSQL.

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

CONTRIBUIDORES:

APELLIDOS Y NOMBRES	ROL / CÓDIGO CVLAC / E_MAIL				
Veracierta, Gabriela	ROL	CA	AS	TU X	JU
	CVLAC:	V-14.616.683			
	E_MAIL	gveracierta@hotmail.com			
	E_MAIL				
Dorta, Pedro	ROL	CA	AS	TU	JU X
	CVLAC:	V-12.914.617			
	E_MAIL	dortap@hotmail.com			
	E_MAIL				
Rodríguez, Rhonald	ROL	CA	AS	TU	JU X
	CVLAC:	V-14.077.185			
	E_MAIL	rhoen2003@hotmail.com			
	E_MAIL				
	ROL	CA	AS	TU	JU
	CVLAC:				
	E_MAIL				
	E_MAIL				

FECHA DE DISCUSIÓN Y APROBACIÓN:

2010	07	30
AÑO	MES	DÍA

LENGUAJE. SPA

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

ARCHIVO (S): TESIS

NOMBRE DE ARCHIVO	TIPO MIME
TESIS.Monitoreodeservicios.doc	APLICATION/MSWORD

CARACTERES EN LOS NOMBRES DE LOS ARCHIVOS: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. a b c d e f g h i j k l m n o p q r s t u v w x y z. 0 1 2 3 4 5 6 7 8 9.

ALCANCE

ESPACIAL: _____ (OPCIONAL)

TEMPORAL: _____ (OPCIONAL)

TÍTULO O GRADO ASOCIADO CON EL TRABAJO:

_____Ingeniero en Computación_____

NIVEL ASOCIADO CON EL TRABAJO:

_____Pre-Grado_____

ÁREA DE ESTUDIO:

_____Computación y Sistema_____

INSTITUCIÓN:

_____Universidad de Oriente – Núcleo de Anzoátegui_____

METADATOS PARA TRABAJOS DE GRADO, TESIS Y ASCENSO:

DERECHOS

_____De acuerdo con el artículo 41 del reglamento de trabajo de grado: _____

"Los trabajos de grado son de exclusiva propiedad de la universidad de oriente, y sólo podrán ser utilizados para otros fines con el consentimiento del consejo de núcleo respectivo, quién deberá participarlo previamente al consejo universitario, para su autorización".

Ramos A., Angel L.	**Roa Q., Miguel A.**
AUTOR	**AUTOR**

Veracierta, Gabriela	**Dorta, Pedro**	**Rodríguez, Rhonald**
TUTOR	**JURADO**	**JURADO**

POR LA SUBCOMISIÓN DE TESIS