ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

TESIS PREVIO A LA OBTENCIÓN DE TÍTULO DE:

INGENIERA EN SISTEMAS E INFORMÁTICA

TEMA: "ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INFORMÁTICO PARA LA COOPERATIVA DE AHORRO Y CRÉDITO PRIMMA LTDA."

AUTOR: JEHTCY AMZIRA VALLADARES CALLE

DIRECTOR: ING. VILLACÍS, CÉSAR

CODIRECTOR: ING. GRANIZO, EVELIO

Sangolquí

2015

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

CERTIFICADO

Ing. César Villacís

Ing. Evelio Granizo

CERTIFICAN

Que el presente trabajo denominado "ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INFORMÁTICO PARA LA COOPERATIVA DE AHORRO Y CREDITO PRIMMA LTDA." fue realizado en su totalidad por la Srta. Jehtcy Amzira Valladares Calle, como requerimiento parcial a la obtención del título de INGENIERA EN SISTEMAS E INFORMÁTICA.

Sangolqui, abril del 2015

Ing. César Villacís
DIRECTOR

ING. EVELIO GRANIZO
CODIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

AUTORÍA DE RESPONSABILIDAD

Yo, Jehtcy Amzira Valladares Calle, declaro que el presente trabajo es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación personal y que hemos consultado las referencias bibliográficas que se incluyen en el documento.

La Universidad de las Fuerzas Armadas – ESPE puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la ley de Propiedad Intelectual por su reglamento y por la normativa institucional vigente.

Sangolquí, abril del 2015

Jehtcy Amzira Valladares Calle

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo:

Jehtcy Amzira Valladares Calle

Autorizo a la Universidad de las Fuerzas Armadas – ESPE la publicación en el repositorio digital de la Biblioteca Alejandro Segovia el presente proyecto de tesis "ANÁLISIS, DISEÑO Y CONSTRUCCIÓN DE UN SISTEMA INFORMÁTICO PARA LA COOPERATIVA DE AHORRO Y CREDITO PRIMMA LTDA." así como también los materiales y documentos relacionados a la misma.

Sangolquí, abril del 2015

Jehtcy Amzira Valladares Calle

DEDICATORIA

A mi hija Milenita, por haber sacrificado el tiempo que debimos pasar juntas, por haber colmado mi vida de felicidad infinita y por ser el motor que me mueve cada día; a mi madre por haberme acompañado en el trajinar de la vida y por haberme permitido estudiar en las instituciones educativas de mi preferencia, sin importar los sacrificios que tuviera que hacer. A mi padre, porque de él aprendí a ser una persona fuerte y perseverante; y sobre todo por el inmenso amor que siempre me brindó.

Jehtcy Amzira Valladares Calle

AGRADECIMIENTO

En primer lugar quiero agradecerle a Dios, porque a pesar del tiempo transcurrido, me ha permitido alcanzar mi meta de convertirme en una profesional, y sobre todo por la dosis de fortaleza que me da cada día. Al Ing. Evelio Granizo que me animó constantemente para materializar este logro y por su apoyo en cada paso. A mis hermanos Milton y Gioconda por su cariño incondicional y a la familia Andrade Castañeda que constantemente me estuvieron animando para salir adelante.

Jehtcy Amzira Valladares Calle

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

La cooperativa de ahorro y crédito PRIMMA LTDA., nace en el seno de una compañía de seguro con la infraestructura instalada, tanto en hardware como en software, por lo que no es necesario seleccionar las herramientas a ser utilizadas para el desarrollo del sistema informático, excepto la metodología usada para el análisis, diseño y construcción del sistema; la cual se desarrollara bajo los lineamientos dados en la metodología OMT. Por lo tanto, la cooperativa toma la decisión de desarrollar un sistema informático propio, a la medida y con una visión a futuro; que le permita ir creciendo paulatinamente. El sistema informático está enfocado en la implementación de los siguientes módulos: Generales: Dirigido a cubrir la gestión de usuarios, roles de usuarios y parámetro del sistema; Socios**:** Está enfocado a realizar la gestión de socios, con la visión de mantener una base de datos amplia, que permita proveer la información necesaria para la generación de reportes legales; Cajas**:** Se encarga de la recepción, entrega y registro de dinero entre la cooperativa y los socios; y Captaciones a la Vista: Gestiona el manejo de las cuentas que las cooperativas están autorizadas a mantener, por parte de las entidades de control, como son: certificados de aportación, encajes para préstamos y cuentas de ahorro. Este proyecto logra el objetivo de reducir costos con respecto a un sistema informático comercial, pues los sistemas financieros tienen costos muy elevados, y por otro lado, se cubre las necesidades funcionales de la cooperativa.

ABSTRACT

The credit union PRIMMA LTDA., born to within an insurance company with the installed infrastructure, both hardware and software, so it is not necessary to select the tools to be used in the development of a computer system, except the methodology used for the analysis, design and construction of the system; which will be developed under the guidelines given in the OMT methodology. Therefore, the cooperative makes the decision to develop its own computer system, on measure and with a vision for the future; that allows to grow gradually. The computer system is focused on the implementation of the following modules: Generals. Addressed to cover the management of users, roles of users and system parameter; Partners: It is focused on making partner management, with a view to maintaining a comprehensive database, which allows to provide the information needed to generate legal reports; Tellers: It is responsible for the receipt, delivery and registration of money between the cooperative and partners; and Deposits: Manages handling accounts that cooperatives are authorized to maintain by control entities, such as: contribution certificates, lace for loans and savings accounts. This project achieved the goal of reducing costs with respect to a commercial computer system, because financial systems have very high costs, on the other hand, the functional needs of the cooperative is covered.

CAPÍTULO 1

1. DEFINICIÓN DEL PROYECTO

1.1 Planteamiento del problema

Una cooperativa de ahorro y crédito es una institución que se conforma por la asociación de un grupo de personas, que se unen con el objetivo común de formar un capital económico, con la finalidad de ofrecer créditos con tasas y montos más convenientes y beneficiosos para el grupo de personas asociadas, de lo que lo se puede encontrar en el sector bancario.

Debido a la naturaleza de las cooperativas de ahorro y crédito todos los socios participan en igualdad de condiciones, y el beneficio o ganancia obtenida es distribuido entre los asociados de acuerdo a sus participaciones.

Las cooperativas adicionalmente a la concesión de préstamos, están habilitadas por la ley para realizar captaciones de fondos, ya sea en forma de ahorros a la vista conocidos como cuentas de ahorro (Valores que pueden ser retirados por el socio en cualquier momento) o como ahorros a plazo llamados también certificados a plazo, donde se pacta con el socio un tiempo en el cual el dinero no se podrá retirar y por tango teniendo una tasa de interés mayor.

1.2 Justificación

La empresa Primma S.A. actualmente se encuentra operando en el ramo de los seguros y reaseguros. Vista las necesidades de sus socios y con la visión de ampliar sus servicios, ha determinado que puede aportar al engrandecimiento de sus socios y por tanto del país, apoyando con la concesión de microcréditos. Debido a la naturaleza del negocio de seguros, la compañía no está habilitada a recibir dinero en forma de inversiones o como cuentas a la vista, ni tampoco a prestar su dinero, por lo que los accionistas junto con un grupo de asegurados decidieron crear una cooperativa de ahorro y crédito de tipo cerrado, que operará

únicamente con los socios de la aseguradora, con la finalidad de brindar estos servicio a sus actuales socios.

Para obtener la autorización necesaria para abrir la cooperativa y permiso operación, se presentó la respectiva documentación a la Dirección Nacional de Cooperativas que se encuentra bajo la dirección del Ministerio de Bienestar Social. Este organismo mediante Acuerdo Ministerial No. 0000091, con fecha 1ro de noviembre del 2006 dio su aprobación para la formación y funcionamiento de la Cooperativa de Ahorro y Crédito Primma Ltda.

Una vez obtenida la respectiva aprobación de funcionamiento, la cooperativa necesita de un sistema informático financiero para que le ayude a registrar a sus asociados y a realizar sus transacciones financieras, de modo que pueda mantener el control de estas transacciones y entregar informes tanto a sus asociados como a las instancias de control de la cooperativa.

Por lo antes mencionado se ha visto la necesidad de realizar un sistema informático financiero cooperativista que apoye en el crecimiento institucional, registro, control, análisis y emisión de informes para las entidades de control.

1.3 Objetivos

1.3.1 Objetivo General

Analizar, diseñar e implementar un sistema informático para la Cooperativa de Ahorro y Crédito Primma Ltda., que cubra los requerimientos básicos de un sistema informático y que a su vez sirva como una plataforma para que en el futuro cubran todos sus requerimientos financieros.

1.3.2 Objetivos Específicos

- Establecer los requerimientos básicos iniciales desde el punto de vista informático, considerando las necesidades operativas de una cooperativa de ahorro y crédito en la creación de una plataforma adecuada, utilizando la ingeniería de software

- Crear un sistema parametrizable que permita el crecimiento de la cooperativa a través de la creación de productos financieros a la medida de la institución sin necesidad de modificar al sistema informático.

- Crear una base de datos de socios lo suficientemente completa, con la finalidad de que en el futuro se tengan las herramientas necesarias para calificar al socio, generar reportes con la información requerida por la Superintendencia de Economía Popular o a cualquier otra entidad de control dispuesta por la legislación ecuatoriana sin afectar a la estructura del sistema.

- Aplicar la metodología OMT para el análisis, diseño e implementación del sistema que a desarrollar.

1.4 Organización y estructura de la Empresa

La organización y estructura de la Cooperativa Primma Ltda., se encuentra representada en la *Figura 1*:

1.4.1 Misión

Convertirse en una Cooperativa de Ahorro y Crédito, con cobertura nacional, que ofrece productos y servicios financieros de calidad a sus socios y socios, con transparencia, seguridad y responsabilidad social.

1.4.2 Visión de la Cooperativa

Ser la cooperativa líder del país en intermediación financiera y atención al socio, con personal amable, competente y comprometida; con enfoque en la responsabilidad social y proveyendo un excelente servicio a través de procesos ágiles, eficientes y nuevas tecnologías.

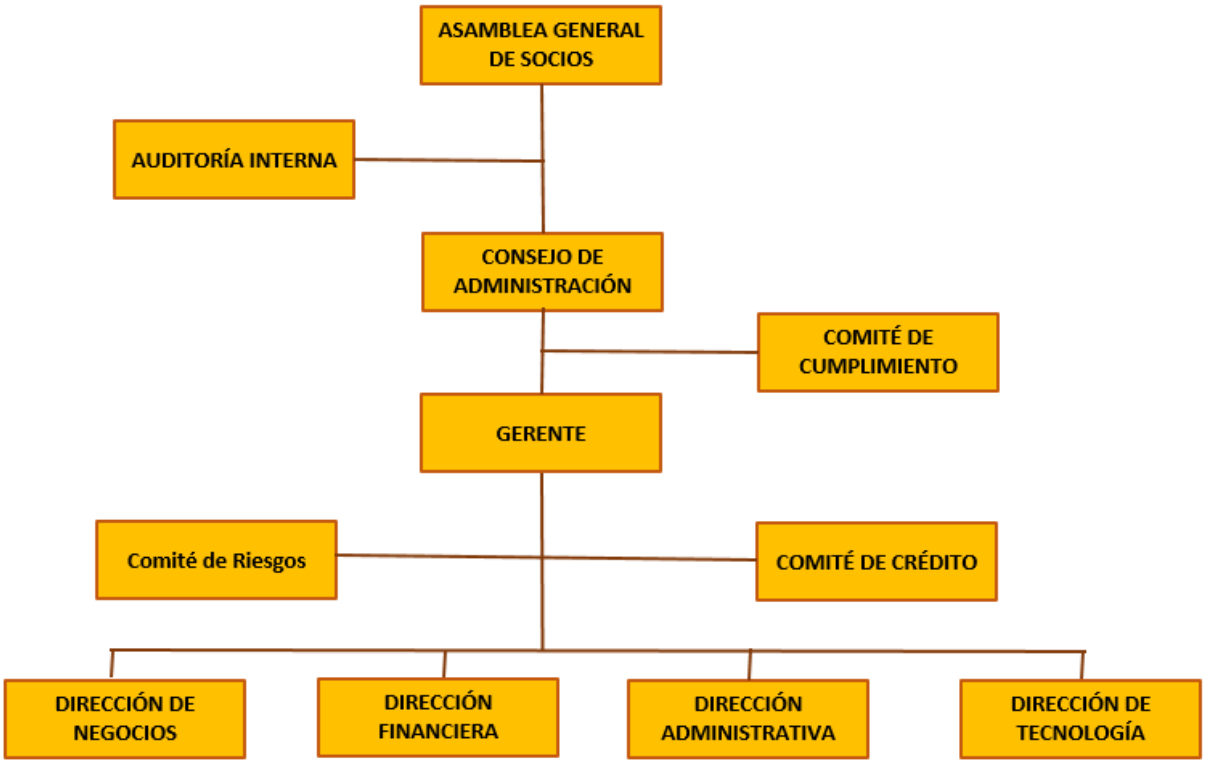

Figura 1: Organigrama de la Cooperativa de Ahorro y Crédito Primma Ltda.

1.5 Alcance

En vista de la magnitud tan amplia que abarca un sistema informático financiero cooperativista, se va a realizar un sistema para cubrir las necesidades y requerimientos de la Cooperativa en los siguientes módulos: Generales, Parámetros del Sistema, Socios, Cajas y Captaciones a la Vista. Por lo tanto los módulos restantes no serán desarrollados dentro de este proyecto de grado.

A continuación se hará una breve descripción de lo que contendrá cada módulo:

1.5.1 Generales y Parámetros del Sistema

Este módulo constituirá el cimiento sobre el cual se va a construir el sistema informático, pues estará compuesto de información de tipo general a ser utilizada en todo el sistema, como son: definiciones de catálogos para la formación de listas de valores, registro de empleados, perfiles de usuarios, administración de usuarios.

Por otro lado, contendrá parámetros, que serán los que darán los lineamientos que definan el comportamiento del sistema. Por lo tanto, se manejará la definición de módulos que conforman el sistema, definición de productos que conforman cada módulo, definición de monedas y mantenimiento de tasa pasivas.

1.5.2 Socios

Este módulo contemplará las necesidades de información para mantener un registro completo de sus asociados, tales como información general del socio, información específica para socios naturales, información específica para socios jurídicos, registro de varios tipos de direcciones por socio, registro de teléfonos, información sobre familiares, cuentas bancarias, tarjetas de crédito, referencias comerciales y referencias personales.

1.5.3 Cajas

Para permitir el control del dinero que se maneja en las cajas y bóveda, se considera importante crear un módulo de Cajas de modo que permita realizar la

apertura de las cajas, incrementos de capital, disminuciones del disponible y cierres de cajas.

1.5.4 Captaciones a la Vista

Este módulo sirve para administrar el registro y mantenimiento de cuentas a la vista, que son aquellos dineros que el socio deja en manos de la cooperativa pero que son de libre disponibilidad, es decir que pueden retirar sus dineros en cualquier momento sin necesidad de solicitar al banco con previo aviso o a una fecha pactada. El módulo debe permitir la creación de productos que sean parametrizables de acuerdo al carácter de cada uno, permitiendo de este modo el manejo de los certificados de aportación, cuentas de ahorro y encajes para préstamos. Además, se contempla el manejo de cálculo diario de la provisión y procesos de capitalización de intereses.

1.6 Visión a largo plazo

Con la implementación de los módulos indicados se pretende sentar las bases del sistema informático lo suficientemente sólidos para que la Cooperativa Prima continúe exitosamente el proceso de automatización de los módulos que no serán implementados en este proyecto, garantizando así un crecimiento sustentable, que le permita satisfacer sus necesidades de información, análisis, control y generación de reportes tanto internos como reportes legales indispensables para un normal desenvolvimiento de sus actividades y de esta forma tener la oportunidad de responder a los desafíos de hoy y atender las oportunidades del mañana.

CAPÍTULO 2

2. MARCO TEÓRICO

2.1. Introducción

Han existido numerosos métodos, notaciones y modelos para modelar distintos tipos de sistemas: organizaciones del mundo real, sistemas de software, sistemas de hardware, etc. Pero los desarrolladores de software deben formalizarse con una metodología y un lenguaje unificado, que cada una de las partes que comprende el desarrollo de todo software de diseño orientado a objetos, se visualice, especifique y documente con lenguaje común. Además, que sea un lenguaje gráfico, a fin de especificar y documentar un sistema de software, de un modo estándar incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema (GONZÁLEZ, 2008).

En esencia un modelo es una simplificación de la realidad, que proporciona "los planos" de un sistema y puede ser más o menos detallado, en función de los elementos que sean relevantes en cada momento, por lo que es fundamental en la construcción de software, porque el modelo ha de capturar "lo esencial" de todo sistema, y puede describirse desde distintos puntos de vista los modelos estructurales (organización del sistema) y los modelos de comportamiento (dinámica del sistema). Lo que permite mejorar en los siguientes aspectos el modelamiento de un sistema: (BOOCH, RUMBAUGH, & JACOBSON, 2015).

- Comunicar la estructura de un sistema complejo
- Especificar el comportamiento deseado del sistema
- Comprender mejor lo que se está construyendo
- Descubrir oportunidades de simplificación y reutilización

Existen muchos acercamientos de desarrollo de software que utilizan modelos orientado a objetos, pero que no tienen todos los soportes para desarrollo de aplicaciones de base de datos. Algunos acercamientos carecen de

suficientes abstracciones y tienen una baja descripción para detalles de implementación.

OMT (Object Modeling Technique) es una metodología de análisis y diseño orientados a objetos, que pone énfasis en la importancia y uso del modelo para lograr una abstracción, en el cual el análisis está enfocado en el mundo real para un nivel de diseño, también pone detalles particulares para modelado de recursos de la computadora. Esta Tecnología puede ser aplicada en varios aspectos de implementación incluyendo archivos, base de datos relacionales, base de datos orientados a objetos. (CHÁVEZ & OLIVARES, 2015)

OMT está construido alrededor de descripciones de estructura de datos, constantes, sistemas para procesos de transacciones, y es muy fácil de aprender porque para aproximadamente el 90% de todos los proyectos se ocupan casi todo el mismo subconjunto de notaciones, además, debido a su sencillez se ha extendido a casi todo los niveles de ingeniería, pero esta simplicidad del método hace posible que en algunos casos (sobre todo complejos) no se puedan modelar con este sistema, pero esta metodología se recomienda por la importancia y las facilidades que brinda el análisis y diseño orientado a objetos. (CHÁVEZ & OLIVARES, 2015)

Además, OMT es compatible con UML que es un lenguaje unificado, el cual cuenta con una notación estándar y semánticas esenciales, para el modelado de un sistema orientado a objetos. El UML unido a una gestión de calidad, evita malos entendidos y entrega ciertas precauciones en la evolución y mantención de programas. Especialmente en lo referente a los requerimientos asociados al levantamiento y diseño funcional de un sistema. Por lo que con el lenguaje UML, los desarrolladores de software sólo tienen que aprender una única notación que vale para los diferentes aspectos del diseño y construcción de los sistemas de software. (GONZÁLEZ, 2008).

2.2. Metodología OMT

La metodología OMT (Object Modeling Technique) es una de las metodologías de análisis y diseño orientadas a objetos, más madura y eficiente que existe en la actualidad; que fue creada por James Rumbaugh y Michael Blaha en 1991. “La gran virtud que aporta esta metodología es su carácter de abierta (no propietaria), que le permite ser de dominio público y, en consecuencia, sobrevivir con enorme vitalidad. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software”. (CHÁVEZ & OLIVARES, 2015)

“Las técnicas orientadas a objetos se basan en organizar el software como una colección de objetos discretos que incorporan tanto estructuras de datos como comportamiento. Esto contrasta con la programación convencional, en la que las estructuras de datos y el comportamiento estaban escasamente relacionadas.” (Mendoza, 2015)

2.2.1. Fases que conforman la metodología OMT

Las fases que conforman a la metodología OMT son: (CHÁVEZ & OLIVARES, 2015)

1. **Conceptualización.** El desarrollo empieza con el análisis de la empresa o negocio, o de cómo los usuarios conciben el sistema y formulan sus requerimientos. La conceptualización es una observación crítica de los procesos de la empresa y su impacto económico. En esta fase se debe tener en cuenta las siguientes preguntas:
   - ¿Cuál es la aplicación?
   - ¿Qué problemas tendrán que ser resueltos?
   - ¿Dónde será usado el sistema?
   - ¿Cuándo será requerido el sistema?
   - ¿Para qué es necesario el sistema?

2. **Análisis.** El analista construye un modelo del dominio del problema, mostrando sus propiedades más importantes. El modelo de análisis es una abstracción resumida y precisa de lo que debe de hacer el sistema deseado y no de la forma en que se hará. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos. Un buen modelo debe ser entendido y criticado por expertos en el dominio del problema que no tengan conocimientos informáticos.
3. **Diseño de Sistema.** El diseñador del sistema toma decisiones de alto nivel sobre la arquitectura de dicho sistema, y se organiza en subsistemas basándose tanto en la estructura del análisis como en la arquitectura propuesta.
4. **Diseño de Objetos**. En esta fase se realiza el diseño de objetos basándose en el modelo de análisis, pero incorporando detalles de implementación. El diseño de objetos se centra en las estructuras de datos y algoritmos que son necesarios para implementar cada clase. OMT describe la forma en que el diseño se implementa en distintos lenguajes (orientados y no orientados a objetos, bases de datos, etc.).
5. **Implementación.** Las clases de objetos y relaciones desarrolladas durante el análisis de objetos se traducen finalmente a una implementación concreta. Durante la fase de implementación es importante tener en cuenta los principios de la ingeniería del software de forma que la correspondencia con el diseño sea directa y el sistema implementado sea flexible y extensible.

2.2.2. Modelos de la metodología OMT

La metodología OMT emplea tres tipos de modelos para describir el sistema:

1. Modelo de objetos
2. Modelo dinámico
3. Modelo funcional

En los siguientes subtemas se describe cada uno de los tres tipos de modelos.

2.2.2.1. Modelo de objetos

Describe la estructura estática de los objetos del sistema (identidad, relaciones con otros objetos, atributos y operaciones), cuyo objetivo es capturar aquellos conceptos del mundo real que sean importantes para la aplicación, y se representa mediante diagramas de objetos. El modelo de objetos proporciona el entorno esencial en el cual se pueden situar el modelo dinámico y el modelo funcional. OMT considera este modelo el más importante de los tres. (CHÁVEZ & OLIVARES, 2015)

"El modelo de objetos se representa gráficamente con diagramas de objetos y diagramas de instancias, que contienen clases de objetos e instancias, respectivamente. Las clases se disponen en jerarquías que comparten una estructura de datos y un comportamiento comunes, y se relacionan con otras clases. Cada clase define los atributos que contiene cada uno de los objetos o instancias y las operaciones que realizan o sufren estos objetos." (Mendoza, 2015)

a. Elementos del modelo de objetos

El modelo de objetos puede contener los siguientes elementos:

1. Objetos o instancias
2. Clases
3. Atributos
4. Operaciones
5. Enlaces
6. Asociaciones
7. Multiplicidad
8. Atributos de una asociación
9. Calificación

10. Roles o papeles
11. Restricciones

Los primeros cuatro elementos se refieren en sí a la clase y objeto, los cuales se verán con más detalle a continuación: (Mendoza, 2015)

1. Objetos o instancias

Un objeto es un concepto o una abstracción con unos límites definidos y que es relevante para el problema en cuestión. Una característica de los objetos es que tienen identidad y son distinguibles, aunque dos objetos tengan los mismos valores para todos sus atributos son diferentes. Los objetos son las instancias de una clase con sus atributos y operaciones. El símbolo gráfico para representar instancias es un rectángulo de esquinas redondeadas, dentro del rectángulo figura la clase a la que pertenece la instancia (entre paréntesis) y los valores de sus atributos, como se muestra en la *Figura 2*2.

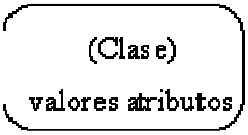

Figura 2: Objeto
(Mendoza, 2015)

2. Clases

Una clase de objetos es una abstracción que describe un grupo de instancias con propiedades (atributos) comunes, comportamiento (operaciones) común, relaciones comunes con otros objetos y lo que es más importante una semántica común. La diferencia entre instancia y clase está en el grado de abstracción. Un objeto es una abstracción de un objeto del mundo real, pero una clase es una abstracción de un grupo de objetos del mundo real. El símbolo gráfico para representar clases es un rectángulo, en el que figura el nombre de la clase, donde las clases se representan en los

diagramas de clases, que son plantillas que describen un conjunto de posibles diagramas de instancias, como se muestra en la *Figura 3*.

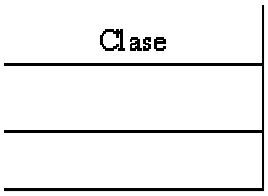

Figura 3: Clase

(Mendoza, 2015)

3. Atributos

Un atributo es un dato contenido en todas las instancias de una clase. Cada atributo tiene un valor para cada una de las instancias, y varias clases pueden tener atributos comunes pero cada atributo debe ser único dentro de una clase, además, los atributos tienen que ser datos, no objetos. Los atributos se representan en el segundo área de los símbolos de clase e instancia. En las clases, figurará el nombre del atributo, el tipo y el valor por defecto, como se muestra en la *Figura 10*. Se tiene la clase Persona con los atributos nombre y fnac.

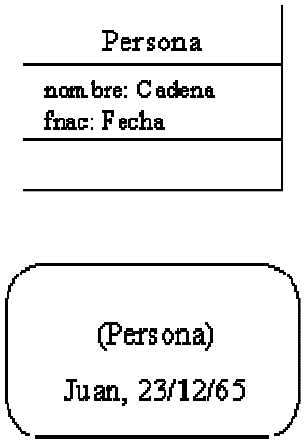

Figura 4: Atributos

(Mendoza, 2015)

4. Operaciones

Una operación o método es una función o transformación, que lleva implícito un objeto destino, sobre el que se va a realizar la operación, y el comportamiento de la operación depende de la clase del objeto destino. Todos los objetos de una clase comparten las mismas operaciones o métodos. Las operaciones figuran en la tercera área del símbolo de las clases, opcionalmente figuran también la lista de argumentos y el tipo de resultado de la operación (si es que la operación devuelve algún resultado), pero en los símbolos de instancia no figuran las operaciones, como se muestra en la *Figura 5*5.

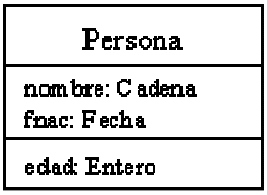

Figura 5: Operaciones

(Mendoza, 2015)

b. Relaciones de composición o agregación

La composición permite representar las relaciones que existen entre los objetos del modelo. Se puede utilizar asociaciones normales para representar la composición pero la relación de composición tiene unas características especiales, por lo que se ha incluido un tipo de asociación especial para representarla; además, estas características son que la composición es transitiva, mientras que las relaciones normales no lo son. La relación de composición se expresa gráficamente con un rombo al extremo de la asociación. Para modelar un objeto que se compone de objetos de varias clases distintas se puede agrupar las relaciones de composición en una única asociación ramificada, indicando la multiplicidad en cada extremo. Por ejemplo, los *Documentos* se componen de *Párrafos*, que a su vez se componen de *Caracteres*, se expresaría gráficamente en la *Figura 6*.

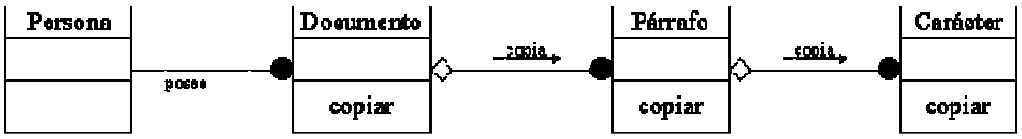

Figura 6: Relación de composición

(Mendoza, 2015)

Además, la agregación no es más que un tipo frecuente de asociación, que tiene unas características determinadas y que se representa en OMT mediante una notación gráfica especial. La diferencia entre asociación y agregación es fundamentalmente semántica, y hay que tener esto presente a la hora de decidir modelar una relación entre clases como asociación o como agregación.

c. Generalización y Herencia

El último tipo de relación puede presentarse entre las clases es la relación de generalización o especialización, en donde la generalización es una forma de abstracción que permite modelar que varias clases comparten una serie de características comunes, a la vez que tienen características propias. En el paradigma OO la generalización se relaciona con el mecanismo de herencia, mediante el que los objetos heredan las características comunes (atributos y operaciones) y es la fuente principal de la reutilización de código.

La generalización o especialización es la relación que se establece entre una clase y una o más versiones refinadas de esta clase, en donde la clase general se denomina superclase y las clases especializadas subclases; como se muestra en la *Figura 7*7.

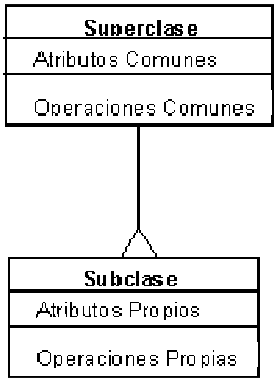

Figura 7: Generalización de Clases

(Mendoza, 2015)

En la superclase se define los atributos y operaciones comunes a todas las subclases, y en cada subclase se define los atributos y métodos específicos para esa subclase. El mecanismo de herencia garantiza que todas las subclases heredan los atributos y operaciones definidos en la superclase. Las subclases no sólo heredan los atributos y las operaciones de la superclase sino que también se pueden redefinirlas, esto se hace cuando un método general definido en una superclase no puede aplicarse tal como está a algunas de las subclases o puede realizarse una implementación más eficiente del método.

Además, se tiene una “clase abstracta” que no tiene instancias directas pero cuyas clases descendientes (clases concretas) sí pueden tener instancias directas. Una clase concreta puede tener subclases abstractas, pero éstas han de tener subclases concretas (las hojas de una jerarquía de clases han de ser clases concretas). Las clases abstractas organizan características comunes a varias clases, y a veces es útil crear una superclase abstracta que encapsule aquellas características (atributos, operaciones y asociaciones) comunes a un conjunto de clases.

Por último, la “herencia múltiple” permite a una clase tener más de una superclase, y así heredar las características de todos sus padres, esto

complica las jerarquías de herencia, que dejan de ser árboles para convertirse en grafos. La gran ventaja de la herencia múltiple es el incremento en las posibilidades de reutilización y el inconveniente es la pérdida de simplicidad conceptual y de implementación. Ver ejemplo de herencia múltiple en la Figura *8*8.

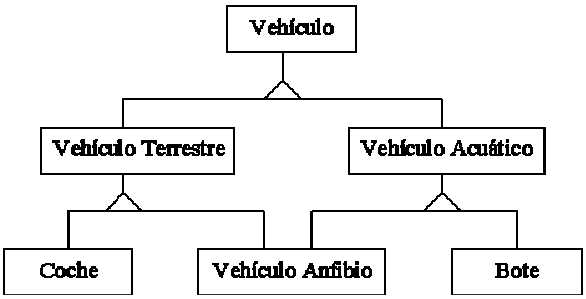

Figura 8: Herencia Múltiple

(Mendoza, 2015)

La herencia múltiple presenta dos problemas: conflictos y herencia repetida; que a continuación se explica cada uno:

- ***Conflicto.*** Se da cuando una clase hereda simultáneamente sus propiedades de varias clases, pueden existir en estas propiedades representadas por el mismo identificador con significados diferentes.
- ***Herencia repetida***. Se da cuando los conflictos que provoca la herencia múltiple se agravan en presencia de situaciones en las que una clase es heredada por otra por medio de caminos distintos.

d. Construcción de un modelo de objetos

Los pasos para la construcción de un modelo de objetos se indican a continuación: (CHÁVEZ & OLIVARES, 2015)

- Identificar las clases de objetos.
- Iniciar un diccionario de datos que contenga descripciones de clases, atributos y asociaciones.
- Agregar asociaciones entre clases.

- Agregar atributos a objetos y ligas.
- Organizar y simplificar las clases de objetos usando herencia.
- Probar las rutas de acceso usando escenarios e iterar los pasos anteriores según sea necesario.
- Agrupar las clases en módulos, basándose en "acoplamiento cercano" y función relacionada.

2.2.2.2. Modelo dinámico

"Describe los aspectos de un sistema que tratan de la temporización y secuencia de operaciones (sucesos que marcan los cambios, secuencias de sucesos, estados que definen el contexto para los sucesos) y la organización de sucesos y estados. Captura el control, aquel aspecto de un sistema que describe las secuencias de operaciones que se producen sin tener en cuenta lo que hagan las operaciones, aquello a lo que afecten o la forma en que están implementadas. Se representa gráficamente mediante diagramas de estado". (CHÁVEZ & OLIVARES, 2015)

a. Diagramas de estados

El modelo dinámico del método OMT se corresponde con el modelo de control o modelo de comportamiento de las técnicas de análisis estructurado, en ambos casos, el modelo se representa mediante diagramas de estados, pero en el caso de OMT estos diagramas de estados se utilizan para modelar el comportamiento de cada clase de objetos, es decir, para modelar el comportamiento común a todas las instancias de una clase. (CHÁVEZ & OLIVARES, 2015)

Los elementos que figuran en un diagrama de estados ya son conocidos, a continuación se va a referirse a las diferencias con respecto al análisis estructurado: (Mendoza, 2015)

1. Estados y eventos

Por estado de un objeto se entiende los valores de los atributos y los enlaces que mantiene un objeto en un momento determinado. Los objetos interactúan unos con otros y como consecuencia de esas interacciones cambian de estado (es decir, cambian el valor de sus atributos o sus enlaces con otros objetos). Los estados que figuran en los diagramas de estados son abstracciones de los valores de los atributos y enlaces de un objeto, es decir, engloban conjuntos de valores de los atributos, de forma que muestran situaciones en las que el objeto presenta un determinado comportamiento. En la *Figura 99* se muestra un ejemplo de un diagrama de estado, con la información: en el objeto *Cuenta Corriente*, el estado *En rojos* engloba todos los estados del objeto en los que el saldo es negativo.

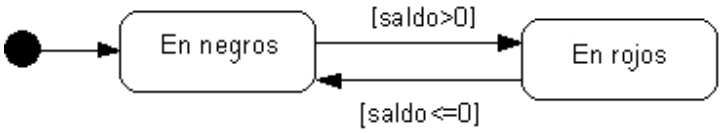

Figura 9: Diagrama de Estados

(Mendoza, 2015)

La interacción entre objetos se realiza mediante "*eventos"*, como consecuencia de la recepción de un evento, un objeto puede realizar una serie de acciones y/o enviar eventos a otros objetos y/o cambiar de estado. Los eventos son el método de intercambiar información entre los objetos, donde esta información consistirá en la mayoría de los casos en una señal lógica, pero en otros puede tratarse de información más compleja, y el evento tendrá una serie de atributos, por ejemplo los eventos *Cancelar* y *retirar():*

- *Cancelar, evento lógico*
- *retirar(número de cuenta, 5000),* evento con atributos

La respuesta de un objeto en un estado determinado ante un evento determinado puede depender de los valores exactos de los atributos, pero

sólo cuantitativamente la respuesta será la misma para todos los objetos que se encuentren en ese estado.

2. Guardas

La encapsulación propia del paradigma orientado a objetos impide con carácter general el acceso directo a valores de los atributos de otros objetos, por este motivo una guarda es una condición que se define sobre los "*atributos propios"* de un objeto. Al recibir un evento, la transición etiquetada con dicho evento se dispara si y sólo si se satisface la guarda.

3. Actividades y acciones

Las actividades y las acciones son las respuestas que tiene que dar un objeto ante los eventos que se producen en el exterior o las guardas (condiciones de datos) que se satisfacen internamente. Típicamente, las acciones y las actividades van a ser operaciones definidas en la clase a la que pertenece el objeto, aunque en ocasiones las acciones pueden consistir en enviar un mensaje a otro objeto. A continuación se define actividad y acción:

- **Actividad.** Es una operación que necesita un tiempo para completarse, y se corresponde siempre con la ejecución de un determinado método o con el objeto inactivo, esperando a que se cumplan determinadas condiciones. Cuando un objeto entra en un estado, ejecuta la actividad correspondiente hasta que ésta finalice o bien hasta que el disparo de una transición haga abandonar dicho estado.

- **Acción.** Es una operación instantánea, que idealmente no necesita tiempo para realizarse y que se asocia, por tanto, al disparo de una determinada transición. Las acciones generalmente consisten en asignar valores a los atributos del objeto o en generar eventos para que sean tratados por otro proceso (con la notación *enviar: evento*), por este motivo, las acciones no suelen corresponderse con

operaciones del objeto, sino posiblemente con parte del código de ciertas operaciones.

4. Trazas de eventos

Los eventos son el medio que utilizan los objetos para comunicarse entre sí, y una traza de eventos representa la secuencia de eventos que se ha producido entre los objetos de un determinado sistema. Esta traza es un diagrama que muestra la secuencia de envíos y recepciones de eventos entre varios objetos. Las trazas de eventos son útiles para mostrar cómo un sistema ha llegado a un escenario determinado, y se mostrarían entonces junto con el diagrama de instancias que represente ese escenario, pero también son útiles para construir el modelo dinámico del sistema, pues muestran una secuencia lógica de eventos que se puede utilizar para realizar los diagramas de estados de cada clase. Entonces, un diagrama de estados permite determinar la secuencia de estados por los que pasa un objeto a partir de una traza de eventos. Si el evento recibido figura en alguna de las transiciones que salen del estado, entonces la transición se dispara y el objeto pasa al estado de llegada de la transición.

b. Composición en los Diagramas de estados

El diagrama de estados de un objeto compuesto estará formado por la unión de los diagramas de estados de cada uno de los objetos que lo componen. Cada uno de los objetos estará en un estado determinado, por lo que el estado del objeto compuesto estará representado mediante una tupla que contiene cada uno de los estados de los componentes. Esta posibilidad permite describir la concurrencia, no sólo en los objetos compuestos, sino en cualquier objeto, porque se puede descomponer un objeto simple en varias partes, cada una que contiene un subconjunto de sus atributos y enlaces, y realizar un diagrama de estados para cada uno de esos subconjuntos. Al igual que antes, el estado del objeto vendrá representado por la composición de estos diagramas de estados.

En algunos casos, se puede necesitar que se muestre la concurrencia entre las actividades que se realizan dentro de un estado, y a su vez que un estado en la actividad a realizar pueda ser descompuesto en una serie de pasos concurrentes. En este caso se divide el estado en dos o más subestados concurrentes. La sincronización de las transiciones de salida de cada uno de los subestados se hace juntando estas transiciones para formar la transición de salida del objeto compuesto.

c. Herencia de Diagramas de estados

Cuando dos clases están relacionadas mediante generalización, entre las propiedades que se heredan de la clase padre está también el comportamiento, por lo que las subclases heredan el diagrama de estados de la superclase, igual que se hace con los objetos compuestos.

Puede existir la posibilidad de conflicto: si la subclase redefine el comportamiento de los objetos respecto a alguno de los atributos heredados. No se pueden incluir transiciones o estados nuevos en el diagrama de estados heredado puesto que entonces no habría una correspondencia entre ambos diagramas de estados. Estas situaciones, en las que las clases herederas modifican tienen un comportamiento distinto al de la clase padre y no es posible establecer una correspondencia entre ambos, son bastante frecuentes en la práctica. En estos casos no se puede hablar de herencia del diagrama de estados sino de una redefinición completa del mismo. Esto es una limitación del modelo que no está resuelta y que contrasta con las posibilidades de redefinición de operaciones y atributos al especializar una clase.

d. Desarrollo de un modelo dinámico

Los pasos para el desarrollo de un modelo dinámica se indican a continuación: (CHÁVEZ & OLIVARES, 2015)

- Preparar escenarios para las secuencias de interacción típicas.
- Identificar eventos entre objetos y preparar trazos de eventos para cada escenario.
- Preparar un diagrama de flujo de eventos para el sistema.
- Desarrollar un diagrama de estados para cada clase que tenga un comportamiento dinámico importante.
- Verificar que los eventos compartidos entre diagramas de estado sean consistentes y correctos.

2.2.2.3. Modelo funcional

"Describe las transformaciones de valores de datos (funciones, correspondencias, restricciones y dependencias funcionales) que ocurren dentro del sistema. Captura lo que hace el sistema, independientemente de cuando se haga o de la forma en que se haga. Se representa mediante diagramas de flujo de datos." (CHÁVEZ & OLIVARES, 2015)

a. Diagrama de Flujo de Datos

La existencia de un modelo funcional basado en el uso de DFDs (Diagrama de Flujo de Datos) distingue OMT de otras metodologías de desarrollo orientado a objetos. Sin embargo, no se puede considerar esto como algo positivo de OMT puesto que no es fácil la adecuación del modelo de procesos del análisis estructurado al tipo de código existente en un sistema orientado a objetos. Por este motivo apenas se usa y, de hecho, escritos recientes de Rumbaugh descartan el uso de DFDs para representar el modelo funcional del sistema, optando, en su lugar, por incluir casos de uso, diagramas de interacción de objetos y especificaciones de operaciones similares a las PSPECs (Process Specification) del análisis estructurado. (Mendoza, 2015)

b. Construcción de un modelo funcional

Los pasos para la construcción de un modelo funcional se indican a continuación: (CHÁVEZ & OLIVARES, 2015)

- Identificar valores de entrada y salida.
- Usar diagramas de flujo de datos para mostrar dependencias funcionales.
- Describir las funciones.
- Identificar restricciones.
- Especificar criterios de optimización.

2.2.3. Relación entre los modelos

A continuación se detalla la relación entre los modelos: (Mendoza, 2015)

1. **Relaciones con el modelo de objetos.**

   El modelo funcional muestra las operaciones que se realizan en cada clase y los argumentos de estas operaciones. El modelo dinámico muestra los estados de cada objeto y las operaciones que éstos realizan al recibir eventos y cambiar de estado.

2. **Relaciones con el modelo dinámico.**

   El modelo funcional muestra las definiciones de las acciones y actividades del modelo dinámico. El modelo de objetos muestra los objetos que sufren o realizan las acciones y actividades del modelo dinámico.

3. **Relaciones con el modelo funcional.**

   El modelo de objetos muestra las entidades que realizan o padecen las funciones del modelo funcional. El modelo dinámico muestra la secuencia en que se realizan las funciones del modelo funcional.

2.3. UML

UML (Unified Modeling Language) o Lenguaje Unificado de Modelado determina un conjunto de notaciones y diagramas estándar, para modelar sistemas orientados a objetos, y describe la semántica esencial de estos diagramas y los símbolos en ellos utilizados. UML es llamado un lenguaje de

modelado, no un método. Los métodos consisten de ambos de un lenguaje de modelado y de un proceso. Es decir, el lenguaje de modelado es la notación (principalmente gráfica) que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a un diseño. (GONZÁLEZ, 2008)

El lenguaje UML es la sucesión de una serie de métodos de análisis y diseño orientadas a objetos que aparecen a fines de los 80's y principios de los 90s, que luego se realiza una estandarización o consolidación de muchas notaciones y modelos usados anteriormente. Se debe a los trabajos de Grade Booch, James Rumbaugh e Ivar Jacobson, que habían sido los creadores de otras tres metodologías orientadas a objetos. El UML en la actualidad se convierte en el lenguaje estándar utilizado por la industria y las grandes empresas, porque UML incrementa la capacidad de lo que se puede hacer con otros métodos de análisis y diseño orientados a objetos. (GONZÁLEZ, 2008)

UML ofrece 9 tipos de diagramas con los cuales se pueden modelar sistemas, los cuales se indican a continuación: (LAMARCA, 2015).

- **Diagrama de Casos de Uso.** Para modelar los procesos "business"
- **Diagrama de Secuencia.** Para modelar el paso de mensajes entre objetos
- **Diagrama de Colaboración.** Para modelar interacciones entre objetos
- **Diagrama de Estado.** Para modelar el comportamiento de los objetos en el sistema
- **Diagramas de Actividad.** Para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- **Diagrama de Clases.** Para modelar la estructura estática de las clases en el sistema
- **Diagrama de Objetos.** Para modelar la estructura estática de los objetos en el sistema
- **Diagramas de Componentes.** Para modelar componentes
- **Diagrama de Implementación.** Para modelar la distribución del sistema

Los diagramas más usados son los de Casos de Uso, Clases y Secuencia, por lo que en este Proyecto de Grado se centrará en éstos, a continuación se realizará una breve descripción de cada uno, utilizando ejemplos de un sistema de venta de entradas de cine por Internet: (HERNÁNDEZ, 2015)

- **El diagrama de Casos de Usos**. Representa gráficamente los Casos de Uso que tiene un sistema informático. Se define un Caso de Uso como cada interacción supuesta con el sistema a desarrollar, que representan los requisitos funcionales. Es decir, se está diciendo lo que tiene que hacer un sistema y cómo. En la *Figura 100* se muestra un ejemplo de Casos de Uso, donde se muestran tres actores (los clientes, los taquilleros y los jefes de taquilla) y las operaciones que pueden realizar (sus roles).

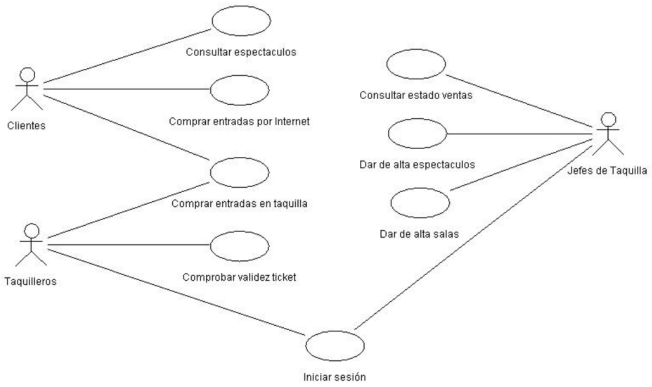

Figura 10: Diagrama de Casos de Uso

(HERNÁNDEZ, 2015)

- **El diagrama de Clases.** Muestra un conjunto de clases, interfaces y sus relaciones. Este diagrama es el más común a la hora de describir el diseño de los sistemas orientados a objetos. En la *Figura 11* se muestran las clases globales, sus atributos y las relaciones de una posible solución al problema de la venta de entradas.

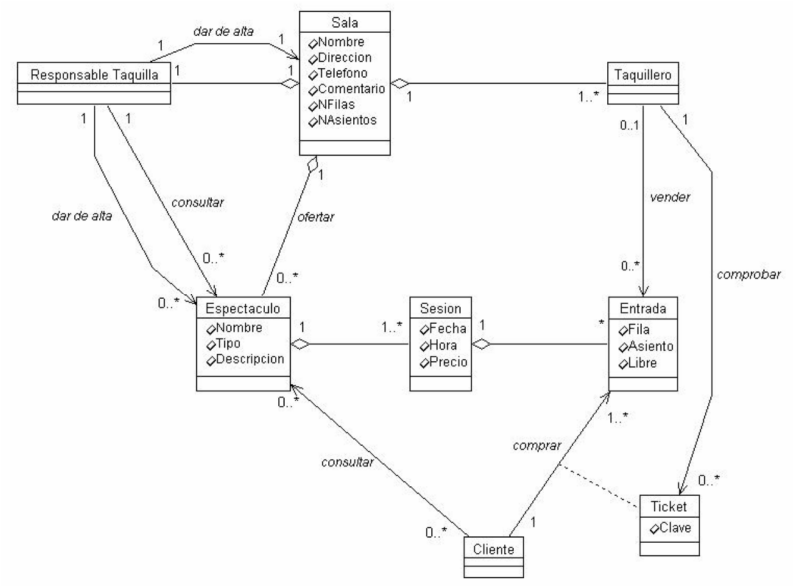

Figura 11: Diagrama de Clases

(HERNÁNDEZ, 2015)

- **En el diagrama de Secuencia**. Se muestra la interacción de los objetos que componen un sistema de forma temporal. Siguiendo el ejemplo de venta de entradas, la *Figura 12* muestra la interacción de crear una nueva sala para un espectáculo.

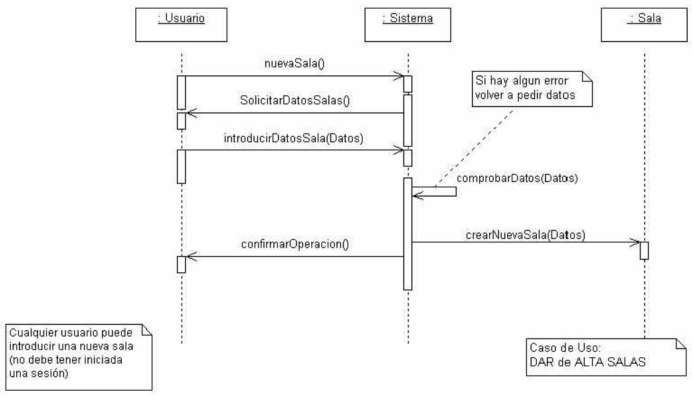

Figura 12: Diagrama de Secuencia

(HERNÁNDEZ, 2015)

El resto de diagramas muestran distintos aspectos del sistema a modelar. Para modelar el comportamiento dinámico del sistema están los de interacción,

colaboración, estados y actividades. Los diagramas de componentes y despliegue están enfocados a la implementación del sistema. (LAMARCA, 2015)

Para lograr un intercambio exitoso de modelos de información entre herramientas del modelado visual de objetos, se definió a UML una semántica y una notación. Una herramienta de UML debe mantener la consistencia entre los diagramas en un mismo modelo. Bajo esta definición una herramienta que solo dibuje, no puede cumplir con la notación de UML. (GONZÁLEZ, 2008)

Además, la estandarización de un lenguaje de modelado es invaluable, porque es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Ambas partes, desarrollador y cliente, deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo.

Los principales beneficios de UML son: (ERIKSSON & PENKER, 2015)

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos.
- La ventaja principal de UML, es unificar las distintas notaciones previas tales como: Rumbaugh, Jacobson, Meyer, Harel, Wirfs-Brock, Fusion, Embly, Gamma et. al., Shlaer-Mellor, Odell y Booch; por lo que es un estándar de amplio uso hoy día y una herramienta fundamental en desarrollos de software de gran envergadura.

Los inconvenientes de UML son: (BOOCH, RUMBAUGH, & JACOBSON, 2015)

- Es demasiado extenso.
- Carece de significados precisos para los elementos representados.
- Dificulta para representar algunos tipos de sistemas de software o elementos.
- Falta de integración con otras técnicas (por ejemplo: diseño de interfaces de usuario)
- Es excesivamente complejo y no está del todo libre de ambigüedades, porque “el 80% de los problemas puede modelarse usando alrededor del 20% de UML”.

2.4. Cooperativa de Ahorro y Crédito

Una Cooperativa de Ahorro y Crédito es una sociedad cooperativa con un fin social que apoya las necesidades financieras de sus socios y fomenta el ahorro de los mismos, valiéndose de actividades propias de una entidad de ahorro y crédito. El ámbito de acción de las cooperativas está limitado al ámbito local.

Las diferencias entre una cooperativa de ahorro y crédito y un banco están dadas básicamente en tres aspectos:

1. **Formación.** Un banco es formado por los dueños que son los accionistas, en tanto, que la cooperativa se forma por sus asociados.
2. **Objetivos.** Los objetivos de un banco están enfocados a ganar dinero para cubrir sus gastos, pago de sueldos a sus empleados y los excedentes son para invertir en el banco o distribuir entre sus accionistas, en tanto, que el objetivo de una cooperativa es servir; por consiguiente las ganancias sirven para cubrir gastos, pagar a sus empleados, y retribuir a sus socios con tasas más bajas en los préstamos y servicios adicionales.

3. **Operación.** En cuento a la operación, se refiere al ámbito en el que operan, pues los bancos al ser instituciones grandes y fuertes, se pueden encontrar en varias ciudades de un país e incluso se pueden encontrar en varios países como son los casos: Banco de China, HSBC, JPmorgan Chase, Banco Santander, City Bank, Wells Fargo, etc. Una cooperativa generalmente va a operar en un ámbito regional o específico para el cual fue creado.

La primera cooperativa de ahorro y crédito fue abierta en el año 1909 en Estados Unidos, proporcionando servicios financieros a costos bajos a muchas personas quienes no tenían acceso a las instituciones bancarias tradicionales o les brindaban servicios insuficientes.

Las cooperativas de ahorro y crédito están autorizadas a realizar toda clase de operaciones activas, pasivas y de servicios; que son permitidas a otras entidades de crédito pero con atención exclusiva a las necesidades financieras de sus asociados. Actualmente estas cooperativas continúan siendo instituciones financieras únicas con una filosofía de servir sin fines de lucro.

Las cooperativas de ahorro y crédito ofrecen una variedad de productos y servicios que le mantienen a las necesidades de sus socios, lo que les permiten mantenerse en el mercado y continuar creciendo. Entre los servicios ofrecidos se tienen:

- Cuentas de ahorro
- Certificados de depósitos a plazo
- Cuentas de jubilación
- Tarjetas de débito y en algunos casos tarjetas de crédito
- Servicios Electrónicos
- Concesión de préstamos personales o de consumo
- Préstamos hipotecarios
- Préstamos comerciales

2.5. Oracle Database

Oracle Database es un sistema de gestión de base de datos objeto-relacional (u ORDBMS (Object-Relational Data Base Management System), desarrollado por Oracle Corporation. (WIKIPEDIA, 2015)

Se considera a Oracle Database como uno de los sistemas de bases de datos más completos, destacando: (WIKIPEDIA, 2015)

- Soporte de transacciones
- Estabilidad
- Escalabilidad
- Soporte multiplataforma.

"Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco; recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux. La única edición gratuita es la Express Edition, las demás ediciones de Oracle Database 10gR2 y Oracle Database 11g. La última versión de Oracle es la versión 11g, liberada en el mes de julio de 2012, es un RDBMS portable ya que se puede instalar en los sistemas operativos y tiene la capacidad de BDD es alta ya que soporta hasta 4 peta bytes de información." (WIKIPEDIA, 2015)

"Oracle Database es una potente herramienta cliente/servidor para la gestión de Bases de Datos para su utilización primero sería necesario la instalación de la herramienta servidor (Oracle 9i) y posteriormente podríamos atacar a la Bases de Datos desde otros equipos con herramientas de desarrollo como Oracle Designer y Oracle Developer, que son las herramientas básicas de programación sobre Oracle. Para desarrollar en Oracle utilizamos PL/SQL un lenguaje de 5ª generación, bastante potente para tratar y gestionar la base de

datos, también por norma general se suele utilizar SQL al crear un formulario. Es posible lógicamente atacar a la base de datos a través del SQL plus incorporado en el paquete de programas Oracle para poder realizar consultas, utilizando el lenguaje SQL.” (PILATAXI, 2015)

“Oracle Database, se puede implementar en cliente/servidor con muchas arquitecturas de red, pero las más usadas con TCP e IPX/SPX. La razón de TPC es clara: es el standard de red usado a nivel internacional en internet. En el caso de IPX resulta de la compatibilidad de Oracle Database para su funcionamiento con Novel/Netware. Oracle Database, posee su propio lenguaje de red, que se asienta de igual manera sobre casi cualquier protocolo; este es Net8 (antiguo Sql-Net). Este protocolo permite la configuración, e implementación sobre otros protocolos debido a su versatilidad, es decir, se adapta a los tamaños de tramas de red, y resulta una buena solución de envío y recepción de datos en cualquier red a nivel LAN.” (PILATAXI, 2015)

Entre las herramientas de Oracle Database 9i, se tienen las siguientes: (PILATAXI, 2015)

- **Browser de objetos.** Este elemento configurable, de tres vistas, muestra toda la información que es relevante.
- **Optimización de rendimiento.** Para optimizar el rendimiento del código que el usuario puede usar el PL/SQL Profiler.
- **Manuales HTML.** Oracle ofrece manuales online en formato HTML. El usuario puede integrar estos manuales en PL/SQL.

Para más información sobre la documentación de Oracle Database, puede acceder a la página Web de Oracle, que tiene la siguiente dirección URL: http://www.oracle.com/technetwork/es/documentation/index.html.

En este proyecto de grado se va utilizar Oracle Database versión 9i, por consiguiente su instalación debe ser consultada en el manual de usuario correspondiente. Se recomienda la página web con la dirección URL: http://flanagan.ugr.es/docencia/2005-2006/2/developer/TutorialInstalacion.html.

2.6. Oracle Developer

Oracle Developer es una herramienta visual para la creación fácil de aplicaciones que trabajen sobre ORDBMS DE Oracle Database. Dentro de Oracle Developer, se distingue dos partes fundamentalmente: (GARCÍA, 2015)

- **Forms Developer.** Permite construir formularios, compilarlos y ejecutarlos; con los que se puede acceder a la Base de Datos. En dichos formularios se puede hacer consultas, codificaciones, inserciones y borrados sobre elementos de la base de datos.
- **Reports Developer.** Permite construir informes con los que se puede presentar e imprimir los datos de una forma ordenada y con un formato visualmente agradable.

Cuando se trabaja en grupos sobre formularios o informes se debe copiarlos regularmente en una carpeta compartida para todos, de modo que, cuando quieran realizar algún cambio, se deberá copiarlos de dicha carpeta y luego volverlos a subir a la misma. Este sistema es muy engorroso y poco fiable pues es bastante normal que las versiones se pierdan y se desintegren con frecuencia. (PILATAXI, 2015)

Los diseñadores pueden hacer uso de nuevos rasgos versátiles, tales como: (GARCÍA, 2015)

- La generación de código automáticamente
- Diseñadores poderosos y magos
- Contexto-sensible la ayuda en línea
- El fácil uso de procedimientos para guardarlos en el editor

Oracle Developer 6i ofrece ventajas con respecto a sus competidores, siguiendo un estándar y una normalización de la seguridad de acceso a datos, como puede ser: (PILATAXI, 2015)

- Un standard de programación plenamente adaptado a las normas de SQL Oracle
- Un entorno de programación mejorado para facilitar el desarrollo
- La posibilidad de desarrollar junto al servidos IAS (Internet Aplication Server) una solución para internet.

Las principales características de Oracle Developer 6i son: (PILATAXI, 2015)

- **Poderoso Editor PL/SQL.** Con su sintaxis destacada, SQL y PL/SQL help, descripción de objetos y muchas otras sofisticadas características, con el editor impresionante.
- **Depurador (debugger) integrado.** Ofrece todas las opciones que se pueda desear: Step In, Step Over, Step Out, etc.
- **Query Builder.** Esta herramienta gráfica hace fácil crear nuevas expresiones o modificar las existentes.
- **PL/SQL Beautifier.** Permite formatear el código a través de unas reglas definidas por el usuario.
- **SQL Window.** Permite ingresar cualquier expresión SQL y ver y editar los resultados fácilmente.
- **Command Window.** Para desarrollar y ejecutar scripts sin tener que dejar el confortable PL/SQL Developer IDE.
- **Reportes.** Permite usar fácilmente reportes estándar o reportes creados por el usuario.
- **Proyectos.** PL/SQL le permite organizar los items de proyectos que el usuario necesite, compilarlos, moverlos de un proyecto a otro.

Para más información sobre la documentación de Oracle Database, puede acceder a la página Web de Oracle, que tiene la siguiente dirección URL:

http://www.oracle.com.

En este proyecto de grado se va utilizar Oracle Developer versión 6i (Forms Developer y Reports Developer), por consiguiente su instalación debe ser consultada en el manual de usuario correspondiente. Se recomienda la página web con la dirección URL:
http://flanagan.ugr.es/docencia/2005-2006/2/developer/TutorialInstalacion.html.

2.7. Cacoo

Cacoo es una herramienta muy útil para desarrollar una gran cantidad de diagramas online, que brinda sus funcionalidades completamente gratis, sin necesidad de descargar o instalar otro software. Cacoo también provee de diversas funcionalidades como la de trabajo colaborativo simultáneo, donde puedes editar un diagrama en conjunto con otras personas al mismo tiempo, y sobre todo, cuenta con una interfaz muy fácil de utilizar. Realiza diversos tipos de diagramas, entre esquemas, diagramas UML, prototipo de pantallas, entre otros. Además, es completamente gratuita y contiene una diversa variedad de paletas de diferentes objetos. (SIERRA, 2014)

Cacoo es una herramienta online para hacer diagramas de diversos tipos, desde el esquema de una oficina hasta diagramas UML, pasando por el prototipado de pantallas. Que hace interesante esta herramienta: (PÉREZ, 2010)

- Cacoo es totalmente gratuita sin limitaciones y dispone de una gran variedad de paletas de objetos diferentes.
- Cacoo permite el trabajo colaborativo simultáneo, es decir un mismo diagrama se puede editar a la vez por más de una persona.
- Facilidad de uso, siendo muy sencillo alinear los elementos, unirlos mediante líneas, cambiar de tamaño, rotar, etc.

Las ventajas son las siguientes: (SIERRA, 2014)

- Una de las grandes ventajas de Cacoo es que permite trabajar de manera colaborativa, y trabajar sobre el mismo archivo en tiempo real.
- Online
- Fácil y sencilla
- Rápida a la hora de utilizar para desarrollar
- Herramientas complejas
- Gratuita

Las desventajas son las siguientes: (SIERRA, 2014)

- Otros pueden utilizarla
- Tener que crear una cuanta para poder utilizarla
- Nivel de complejidad

El tutorial está escrito usando el siguiente entorno: (PÉREZ, 2010)

- **Hardware:** Portátil MacBook Pro 17' (2.93 GHz Intel Core 2 Duo, 4GB DDR3 SDRAM, 128GB Solid State Drive).
- NVIDIA GeForce 9400M + 9600M GT with 512MB
- **Sistema Operativo:** Mac OS X Snow Leopard 10.6.3
- Safari 5

En este proyecto de grado se va utilizar Cacoo con licencia libre, por consiguiente su instalación debe ser consultada en el manual de usuario correspondiente.

2.8. PowerDesigner

PowerDesigner, es una herramienta para el análisis, diseño inteligente y construcción sólida de una base de datos, y un desarrollo orientado a modelos de datos a nivel físico y conceptual; que da a los desarrolladores Cliente/Servidor la más firme base para aplicaciones de alto rendimiento. (ECURED, 2015). Su estructura modular brinda la facilidad, a las organizaciones, de utilizar las herramientas que ellas necesiten según el tamaño y alcance de sus proyectos.

Las características de PowerDesigner son los siguientes: (ECURED, 2015)

- Es nombrada la herramienta líder en modelamiento empresarial
- Permite a las empresas, de manera más fácil, visualizar, analizar y manipular metadatos, logrando una efectiva arquitectura empresarial de información.
- Brinda un enfoque basado en modelos, el cual permite alinear al negocio con la tecnología de información, facilitando la implementación de arquitecturas efectivas de información empresarial.
- Brinda potentes técnicas de análisis, diseño y gestión de metadatos a la empresa.
- Combina varias técnicas estándar de modelamiento con herramientas líder de desarrollo, como .NET, Sybase WorkSpace, Sybase Powerbuilder, Java y Eclipse, para darle a las empresas soluciones de análisis de negocio y de diseño formal de base de datos.
- Trabaja con más de 60 bases de datos relacionales.

Los beneficios de PowerDesigner son los siguientes: (ECURED, 2015)

- **Constituye una Elección Segura.** El compromiso de PowerDesigner con el modelamiento de datos, UML y de negocio, además de estar comprobado en el mercado, lo hace la elección segura para todos los requerimientos de modelamiento. PowerDesigner es el estándar en muchas organizaciones a nivel mundial.
- **Mejora la Productividad Individual.** El enfoque orientado a modelos de PowerDesigner incorpora una serie de generadores DDL y de código personalizables, y capacidades de ingeniería reversa y sincronización de código, que reduce significativamente los esfuerzos de creación, mantenimiento y reingeniería manual de código.
- **Brinda Facilidad de Uso Gráfica.** La interfaz gráfica es altamente personalizable, hace que las tareas comunes sean muy fáciles y le da el

poder a los usuarios avanzados de tener acceso rápido a todas las funciones.

- **Alinea el Negocio con el Área de Tecnología.** Facilita el alineamiento del negocio con el área de tecnología a través de técnicas de colaboración en grupo.
- **Mejora la Productividad en Grupo.** Brinda a todos los modeladores un ambiente ideal para compartir recursos a través de un repositorio de meta-datos único, completo y seguro para todos los tipos de modelos.
- **Documenta los Sistemas Existentes:** Adopta una mayor colaboración a nivel empresarial a través de generación de reportes flexible y basada en asistentes, o RTF / HTML multi-modelo.
- **Brinda Soporte Abierto.** Permite el entendimiento de sistemas heterogéneos con el soporte a los principales estándares de lenguajes de desarrollo, XML, base de datos y procesos, con una sola herramienta e infraestructura.
- **Es Altamente Personalizable.** Puede ser fácilmente "programado" para asegurar los estándares y prácticas corporativas o legales, a través del soporte a "scripts" VB, una interfaz COM completamente programable, un meta-modelo personalizable y un API totalmente documentado.
- **Reduce el Impacto del Cambio.** Reduce significativamente el costo y tiempo al implementar cualquier cambio a través de una vista exacta, bi-direccional y multi-modelo para análisis de impacto que integra todos los modelos de requerimientos, análisis, base de datos y aplicación.

La característica de análisis y diseño flexible de PowerDesigner permite, entre otras cosas, crear una base de datos eficazmente y de manera estructurada, sin necesidad de adoptar una metodología específica. PowerDesigner incluye seis herramientas altamente integradas, que facilita a individuos y/o a miembros de un equipo de trabajo, desarrollar proyectos que satisfagan sus necesidades de manera efectiva; estos módulos son: (PUERTA, 2015)

- **PowerDesigner ProcessAnalyst:** Permite analizar el flujo de datos de toda la empresa, a través de los departamentos hasta el usuario final.
- **PowerDesigner DataArchitect:** Provee a los diseñadores de las bases de datos una manera eficiente para la creación inteligente, depuración e ingeniería de reversa del modelado, tanto conceptual como físico de los datos.
- **PowerDesigner AppModeler:** Permite el diseño y ajuste de los componentes de objetos y datos en aplicaciones de uso común como PowerBuilder, Power++, Visual Basic y Delphi, ajustando el modelo de base de datos. Junto con la aplicación de servidor PowerDynamo (incluido) se pueden publicar las bases de datos en Internet directamente del modelo de base de datos. Esta herramienta también puede generar páginas de servidor activas para Microsoft Internet Information Server.
- **PowerDesigner WarehouseArchitect:** Provee un poderoso datawarehousing para el diseño e implementación de una base de datos. Cuenta con soporte para bases de datos tradicionales DBMS y bases de datos en plataformas de sistemas analíticos usando modelados dimensionales, esquemas de "estrella" y "nieve", particionamiento y agregación. También cuenta con un alto desempeño en el indexamiento de esquemas.
- **PowerDesigner MetaWorks:** Permite fácilmente ver y compartir la información del modelado de datos con una definición constante de objetos. También puede comparar y mezclar dos modelos de datos paso a paso.
- **PowerDesigner Viewer:** Crea reportes de los modelos físicos, conceptuales y procesos del modelado de la base de datos. También permite generar reportes para Internet en HTML. Este producto cuenta con demos directos de sitio de Sybase en Internet para su evaluación.

Además de todas estas características, PowerDesigner ofrece las posibilidades de: (PUERTA, 2015)

- Soporte para tipos de datos abstractos: PowerDesigner soporta la identificación de tipos de datos abstractos con ingeniería inversa de aplicaciones para Oracle.
- Soporte para usuarios de bases de datos: Los usuarios de bases de datos pueden ser recogidos de una base de datos existente y luego almacenados en un modelo físico de datos. Ahora, es posible añadir nuevos usuarios y también asignar usuarios como propietarios y vistas.
- Mayor selectividad en ingeniería inversa: PowerDesigner permite seleccionar no sólo las tablas que se desean cargar, sino todo tipo de objetos de la base de datos.
- Cálculo del tamaño de las bases de datos: Puede calcular y definir el tamaño definitivo de bases de datos de nuevo diseño y construcción, incluyendo tamaños detallados de índices y tablas.

Para más información sobre la documentación de PowerDisigner, puede acceder a la página Web de MTBase Sybase de Colombia, que tiene la siguiente dirección URL:
http://www.mtbase.com/productos/modelamientometadatos/powerdesigner.

En este proyecto de grado se va utilizar PowerDisigner versión 15, por consiguiente su instalación debe ser consultada en el manual de usuario correspondiente.

CAPITULO 3

3. ANÁLISIS Y DISEÑO

3.1 Fase de Análisis

En base a la metodología OMT planteada para el diseño del proyecto la cual define las técnicas y métodos, se va a definir los diagramas, necesarios para representar y modelar el Sistema Cooperativista dentro del alcance planteado, los cuales describirán sus respectivas funcionalidades para la implementación del sistema.

La definición de las funcionalidades cubiertas en este proyecto, se realizó con la persona encargada de delinear los procesos de la cooperativa.

3.1.1 Especificación de Requerimientos

3.1.1.1 Propósito

El propósito de esta sección consiste en hacer una descripción completa y clara del funcionamiento del sistema cooperativista de acuerdo, el alcance definido.

Con la especificación de los requerimientos de la cooperativa se establecerá la naturaleza del funcionamiento del sistema, cómo interactuará el sistema con su entorno, cuáles van a ser sus estados y su funcionamiento; de modo que se pueda obtener un esquema con el cual direccionar este proyecto y establecer un punto de acuerdo con la Cooperativa.

La especificación de requerimientos está dirigida a la cooperativa y al desarrollador, por lo que el lenguaje a utilizar debe ser comprensible para las partes involucradas y así estar todos en sintonía con las necesidades que se tiene al momento y los que se espera obtener.

3.1.1.2 Identificación de Roles y Tareas

1. Roles

a) Administrador:

El Administrador es aquel que posee privilegios y permisos para administrar el sistema, desde el punto de vista creación y administración en lo que se refiere a la seguridad del sistema; por tanto se encargará de la creación y mantenimiento de perfiles o roles de usuarios del sistema, en donde a su vez se define las transacciones a las que tiene acceso y horarios. También se encargará de la administración de los parámetros del sistema como registro de tasas, montos de manejo por productos, entre otros.

b) Cajero Bancario:

El Cajero es aquella persona que interactúa con el socio en las transacciones en las que interviene dinero en efectivo y cheques.

c) Jefe de Caja:

Es la persona que trabaja administrando la bóveda, que es el encargado de entregar el dinero a los cajeros para su apertura, de recibir el dinero recaudado ya sea por cierre de caja o por exceso de dinero en caja.

d) Autorizador

El Autorizador es el encargado de revisar aquellas transacciones que requieren un nivel de autorización, por tanto tiene la potestad de autorizar o negar las solicitudes de los oficiales que trabajan atendiendo a los socios.

e) Ejecutivo de Cuenta

El Ejecutivo de Cuenta es aquella persona que atiende al socio en actividades tales como recepción de documentos, aperturas de cuentas,

asegurarse de que los trámites bancarios se cumplan de acuerdo con las normativas y políticas de la Cooperativa, consulta de saldos, emisiones de certificados bancarios, cierres de cuenta, apertura de certificados a plazo, solicitudes de préstamos, conocer los servicios que brinda la cooperativa e informar al socio.

f) Operador:

Este rol es para aquella persona encargada de correr un proceso por lotes, cuyo objetivo es ejecutar el Cierre de Operaciones del día, que consiste en realizar el cálculo de provisiones diarias para el pago de intereses, capitalización de los mismos dependiendo de la frecuencia de pagos definida en el producto, generación de estados de cuenta, generación de saldos promedio, cobro de cargos, entre otros.

2. Tareas:

a) Administrador:

Las tareas del Administrador son:

- Creación de roles de usuario
- Creación de usuarios y asignación de roles
- Mantenimiento de usuarios
- Mantenimiento de parámetros del sistema

b) Cajero Bancario:

Las tareas del Cajero Bancario son:

- Control de la caja
- Registro de todas las operaciones que recibe y emite
- Reportar al Jefe de Cajas
- Manejo de dinero, cheques y otros documentos bancarios
- Cuadratura diaria de la caja a su cargo

c) Jefe de Caja:

Las tareas que tiene a carago el Jefe de Caja son:

- Control de la Bóveda
- Entrega de dinero a los cajeros para su apertura
- Recepción y control del cierre de caja
- Cuadratura de la Bóveda a su cargo
- Reportar al subgerente los movimientos en bóveda

d) Autorizador

El autorizador es el encargado de revisar y autorizar aquellas transacciones que requieren autorización.

e) Ejecutivo de Cuenta

Las tareas que desempeña el Ejecutivo de Cuenta son:

- Apertura de cuentas
- Creación de socios nuevos
- Mantenimiento de datos del socio
- Creación de cuentas
- Consulta de saldos
- Transferencias
- Consultas en el sistema

f) Operador

La tarea del Operador es ejecutar el proceso de cierre de operaciones diarias y reportar los errores que se hayan producido.

3.1.2 Especificación de Requerimientos Funcionales

Esta sección es muy importante para limitar la funcionalidad y el diseño de la aplicación, el cual requiere de un nivel de detalle suficiente amplio como para crear un marco de trabajo en donde se vaya desarrollando los escenarios y los casos de uso; por otro lado la cooperativa pueda ir revisando y validando el diseño, en el caso de ser necesario que vaya creando nuevos requerimientos, y finalmente en el caso de existir algún tipo de inconsistencia o error, realizar las respectivas observaciones para ajustar el diseño.

Se va a iniciar presentando un par de escenarios que son genéricos para todos los roles del sistema que es la validación del usuario en el momento que este ingresa al sistema y la validación de la transacción que el usuario del sistema necesita ejecutar:

3.1.2.1 Ingreso al Sistema:

El sistema debe autentificar al usuario que está ingresando a la aplicación, para lo cual debe proveer el nombre del usuario y su clave.

3.1.2.2 Validación de Transacciones:

Cada vez que un usuario intenta ingresar a una transacción, el sistema debe validar los siguientes puntos:

- Comprobar que el usuario de acuerdo a su perfil esté habilitado para ingresar a la transacción.
- Validar el horario de trabajo para garantizar que se encuentre dentro de los límites permitidos.

3.1.2.3 Definición de escenarios de acuerdo al rol

A continuación se procede a detallar los escenarios definidos por cada rol:

a. Rol Administrador

1. Gestionar Roles de Usuario

Los Roles de Usuarios permiten definir tipos de usuarios que tienen las mismas funciones y los mismos privilegios como sería el caso de los oficiales de Cuenta, que sin importar en qué lugar trabajen, las funciones y las transacciones a las que tienen acceso son las mismas. Mediante la gestión de Roles de Usuario se delimita las transacciones a las que se tiene acceso.

Las procesos que se pueden realizar en este escenario de Gestionar Roles de Usuario son: creación de roles, consulta, mantenimiento de roles y eliminación de roles.

2. Gestionar Usuarios

Esta es otra transacción muy importante en lo que se refiere a seguridad del sistema, pues aquí se deberá permitir crear los usuarios que podrán acceder al sistema, se define el rol que van a desempeñar y el horario de trabajo. Los datos a registrar son información general del usuario como nombre, número del documento de identificación, dirección, nivel de estudio, profesión, fecha de ingreso, fecha de salida, fecha de próximo cambio de clave, estado del usuario y sobre todo la asignación del rol a desempeñar.

La gestión de usuarios del sistema incluye también las transacciones de consulta de usuarios del sistema, mantenimiento de usuarios y retirar el acceso del usuario al sistema, pues por cuestiones de auditoría un usuario no debe ser eliminado físicamente.

3. Gestionar Productos del Sistema

La gestión de productos permite crear los productos financieros que la cooperativa ofrece a sus socios y las características que conforman al producto.

En este escenario se debe contemplar la creación, consulta, mantenimiento y eliminación del producto siempre y cuando no existan cuentas de socios asociadas al producto.

4. Gestionar Tasas Pasivas

Este escenario es para permitir el mantenimiento de las tasas pasivas con las que los productos del módulo de Cuentas van a trabajar.

5. Gestiona Catálogos del Sistema

A través de este escenario se pretende mantener los catálogos que se van a utilizar en el sistema, estos catálogos se utilizan ampliamente para la generación de listas de valores en todos los módulos del sistema.

b. Rol Cajero Bancario

1. **Abrir caja**

Mediante este proceso el cajero puede registrar el dinero con el cual va a iniciar sus operaciones y también habilita a la caja para empezar a operar en el sistema.

2. Recibir Depósitos

Cuando una persona realiza un depósito en una cuenta, el cajero recibe el dinero y la papeleta de depósito, revisa la papeleta que respalda el depósito, hace una revisión de la cantidad recibida, comprueba la validez de los billetes y registra en el sistema el valor recibido en la cuenta indicada en la papeleta.

El sistema por su lado verifica la existencia de la cuenta, registra el depósito e incrementa el saldo de la cuenta involucrada.

3. Gestionar Retiros

Si un socio necesita realizar un retiro de su cuenta, el cajero recibe la papeleta y el documento de identificación, revisa que los datos sean correctos y registra la transacción en el sistema. El sistema primero valida la existencia de la cuenta y que tenga el saldo suficiente para realizar esta transacción; Si es así, la transacción es grabada y el cajero entrega el dinero al socio.

4. Cuadrar Caja

Mediante este proceso el cajero puede registrar el valor que tiene en efectivo versus lo que se ha registrado en el sistema, con esto se va a determinar el estado de la caja, que puede ser: cuadrada, con sobrante o con faltante.

En el caso de que la suma de transacciones registradas sea igual al valor que tiene el cajero a esto se le conoce como Caja Cuadrada.

Si el saldo que tiene el cajero es superior al saldo registrado en el sistema, entonces se dice que existe un sobrante y el sistema debe generar una entrada contable que registre este sobrante.

Si el saldo del cajero es menor al saldo que tiene registrado el sistema, entonces se dice que se ha producido un faltante. En este caso igualmente el sistema debe generar una entrada contable que deje registrado el faltante.

5. Cerrar caja

El proceso de cierre de caja sirve para registrar la entrega del saldo efectivo y cheques que ha recibido durante el día de trabajo.

c. Rol Jefe de Caja

1. Apertura de Bóveda

Este proceso se realiza por una sola vez y se indica el valor con el que se apertura la bóveda, si la bóveda no ha sido abierta no se puede iniciar las operaciones.

2. Recibir dinero de los cajeros

Cuando un cajero tiene dinero en exceso o una vez realizado el cierre de caja, el jefe de caja realiza la recepción del dinero del cajero.

3. Autorizar transacciones por montos

Si una transacción hecha por el cajero requiere autorización por monto, entonces el Jefe de Caja es el encargado de autorizar dicha transacción.

4. Entregar dinero a los cajeros

El Jefe de caja puede entregar dinero al cajero en las siguientes situaciones:

- Abrir la caja para iniciar operaciones
- Cuando el cajero en su operación ya no dispone de efectivo para continuar procesando las transacciones de los socios.

d. Rol Autorizador

1. Revisa, Autoriza o niega transacciones

Existen transacciones que requieren autorización como la aprobación de una solicitud para abrir una cuenta, entonces el Autorizador es el encargado de revisar los datos en la solicitud de apertura de la

cuenta, si estos son correctos, se realiza la autorización y por tanto la apertura de la cuenta.

e. Rol Operador

1. Corre el proceso de cierre de día

A través de este proceso se puede realizar el cierre del día, en el cual se ejecuta varios sub-procesos muy importantes tales como la provisión de interés de acuerdo a los saldos de las cuentas, capitalización de intereses, cobro de cargos y generación de reportes como estados de cuenta.

f. Rol Ejecutivo de Cuenta

1. Gestionar Cuentas

Cuando un socio nuevo o ya existente necesita abrir una cuenta, se acerca a un Ejecutivo de Cuenta, el cual le puede ayudar en el proceso en este requerimiento; la apertura de cuenta deberá cumplir los requisitos mínimos definidos en el producto como son monto mínimo y presentación de ciertos documentos.

2. Gestionar Socios

En este proceso se considera tanto la creación de socios nuevos como el mantenimiento de información de un socio. La creación del socio involucra el registro de información general del socio, registro de su dirección, teléfono, referencias, información del trabajo, entre otros.

3. Transferencias entre cuentas de Socios

El ejecutivo puede gestionar la transferencia de valores entre cuentas de un mismo socio o transferencias a cuentas de terceros dentro de la cooperativa.

4. **Consultas de Saldos**

   A través de este proceso, se debe permitir consultar los saldos de las cuentas del socio.

3.1.3 Especificación de Requerimientos No Funcionales

- Se debe garantizar la integridad de la información, de modo que esta solo pueda ser modificada por quien está autorizado para el efecto y de manera controlada.
- Confidencialidad para que la información esté al alcance solo de aquellas personas autorizadas para su uso.
- Disponibilidad en el momento que sea requerida.
- Auditable, de modo que los cambios y registros de datos, por parte de cualquier usuario, puedan ser atribuibles a una persona.

3.1.4 Restricciones

El sistema informático financiero cooperativista, cumple con las siguientes restricciones:

- A cada usuario del sistema se le asigna un usuario único, con clave secreta y su almacenamiento en forma encriptada. El ingreso de la clave debe mostrar asteriscos.
- El o los operadores del sistema, igualmente tendrán un usuario único, con clave secreta que le dará acceso a labores de generación de reportes o ejecución del proceso de cierre diario de operaciones.
- El ingreso de los usuarios está restringido a horarios, estos son definidos por usuario, mas no por el rol de este.
- El usuario le da acceso a funcionalidades específicas, de acuerdo al rol que desempeña el usuario en la Cooperativa. Esto se logra a través de permitir el ingreso a transacciones de acuerdo a roles de usuarios.

3.2 Fase de Diseño

3.2.1 Diagramas de Caso de Uso

Dentro de la metodología OMT se contemplan distintos aspectos para la representación de requerimientos y su posterior implementación, por lo que se considera utilizar los Diagramas de Caso de Uso para especificar las funcionalidades del sistema y como se relacionan con su entorno, pues estos diagramas representa los distintos requerimientos que hacen los usuarios al sistema.

En la figura 13, se empezará definiendo los casos de uso que describen funcionalidades de tipo genéricas como son el ingreso al sistema y la validación de la transacción que el sistema va a ejecutar, y posteriormente, a partir de la Figura 14 hasta la Figura 19, se definirá las funcionalidades de acuerdo a cada rol.

3.2.1.1 Diagrama de Caso de Uso de funcionalidades Genéricas:

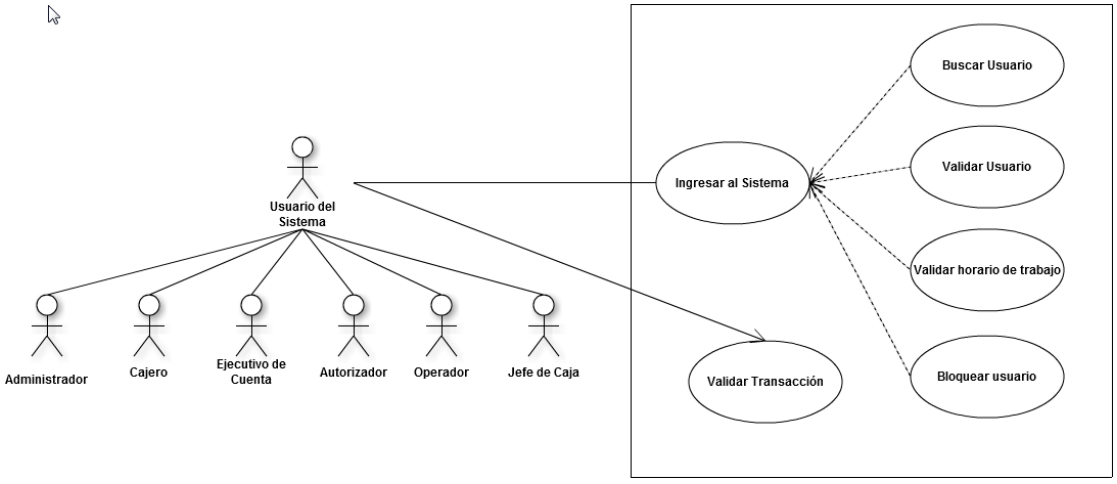

Figura 13: Diagrama de Caso de Uso - Ingreso al Sistema y validación transacciones

3.2.1.2 Diagrama de caso de uso - Rol Administrador:

En la Figura 14, se presenta el diagrama del caso de uso del rol Administrador.

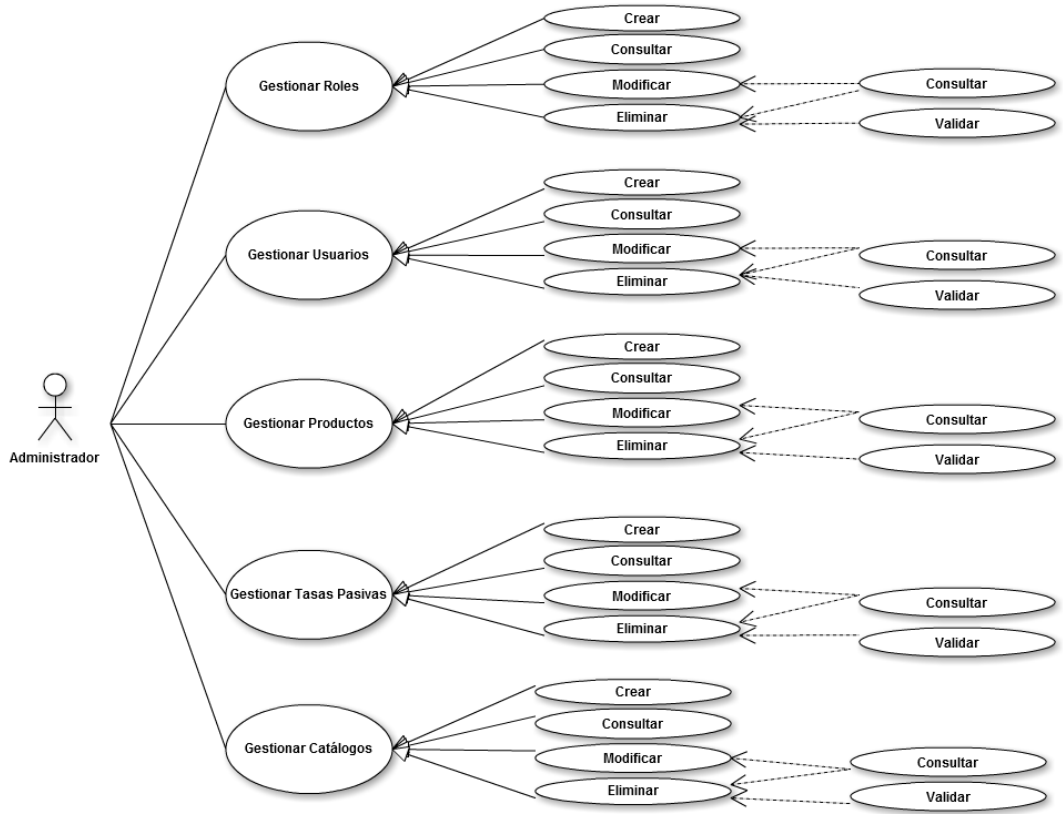

Figura 14: Diagrama de caso de uso del Rol Administrador

3.2.1.3 Diagrama de caso de uso - Rol Cajero

En la Figura 15, se presenta el caso de uso del Rol Cajero.

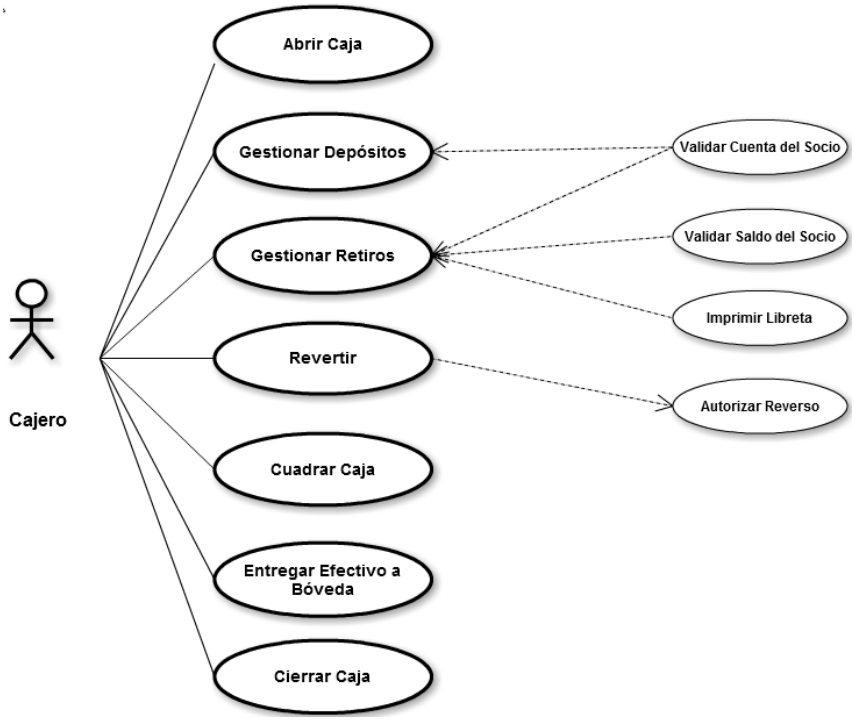

Figura 15: Diagrama de caso de uso del Rol Cajero

3.2.1.4 Diagrama de Caso de Uso – Rol Jefe de Caja.

En la Figura 16 se presenta el caso de uso del Rol del Jefe de Caja.

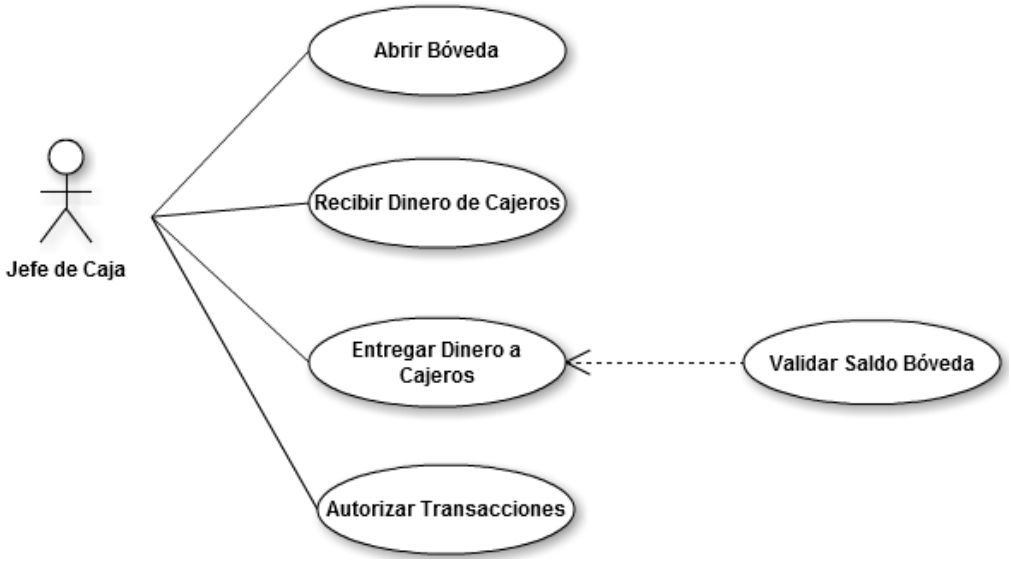

Figura 16: Diagrama de caso de uso del Rol Jefe de Caja

3.2.1.5 Diagrama de Caso de Uso – Rol Ejecutivo de Cuenta

En la Figura 17 se presenta el diagrama del caso de uso del rol que realiza el Ejecutivo de Cuenta.

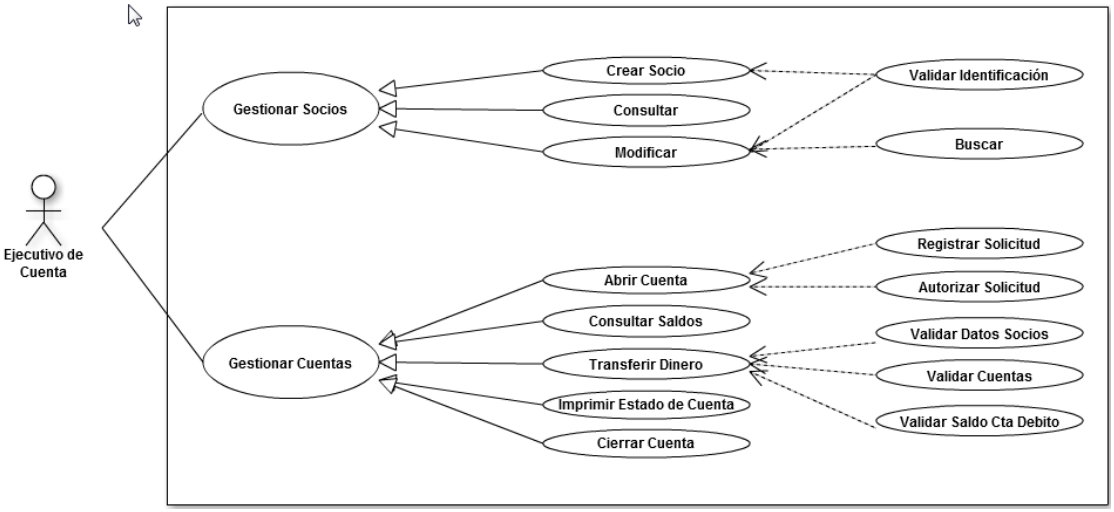

Figura 17: Diagrama de caso de uso del Rol Ejecutivo de Cuenta

3.2.1.6 Diagrama de Caso de Uso – Rol Operador

En la Figura 18, se presenta el diagrama de caso de uso del rol que realiza el Operador del sistema.

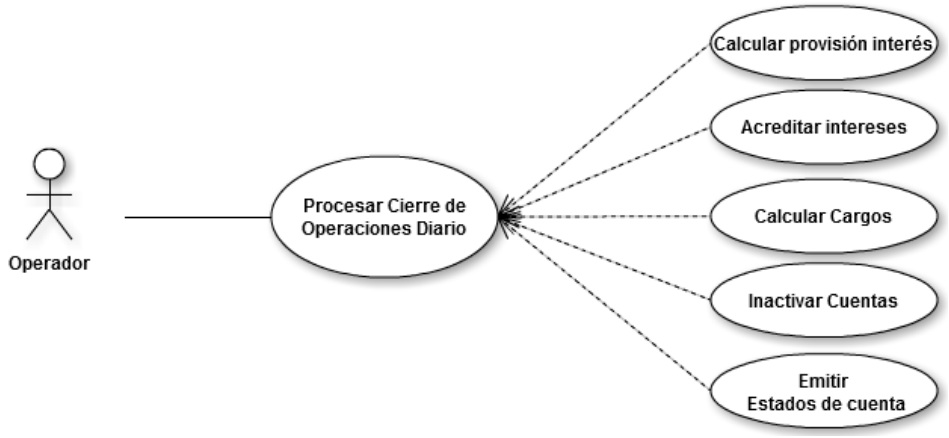

Figura 18: Diagrama de caso de uso del Rol Operador

3.2.2 Especificación de Casos de Uso para el diseño

A continuación se va a especificar paso a paso las secuencias de acciones que forman parte de la funcionalidad del sistema. Se inicia describiendo aquellas funcionalidades genéricas en el sistema, que son independientes del rol que lo ejecuta.

3.2.2.1 Especificación de Caso de Uso: Ingreso al Sistema

En la tabla 1, se va a presentar la funcionalidad del ingreso al sistema, en el momento que cualquier usuario intente conectarse.

Tabla 1:

Caso de uso Ingresar al sistema

Descripción	Ingresar al sistema
Objetivo	Una vez que un usuario intenta ingresar al sistema a través del ingreso de su usuario y su clave, el sistema debe autentificar la valides del usuario, la clave y horarios de acceso garantizando la seguridad del sistema. Para la validación de la clave del usuario, se debe contar con una rutina de modo que no permita que el usuario repita su clave revisando con las 6 últimas claves registradas, además la clave del usuario se debe registrar en forma encriptada a través de un algoritmo.
Actores	Administrador, Cajero, Jefe de Caja, Ejecutivo de Cuenta, Operador
Flujo de Eventos	
Flujo Básico	Ingresar el código de usuario asignado para el ingreso al sistema y la clave.

CONTINUA 

	Si el usuario intenta el ingreso con el usuario correcto pero falla por tres veces consecutivas en el ingreso de la clave, el usuario debe ser bloqueado.
Flujos Alternativos	
Precondiciones	El actor que intenta conectarse al sistema debe ser un usuario del mismo.
Condición de Éxito	El usuario se conecta al sistema correctamente.
Condición de Fallo	El sistema no permite el ingreso ya sea por error en el usuario, clave incorrecta o problemas con el horario de acceso.

3.2.2.2 Especificación de Caso de Uso: Validar transacción

En la tabla 2, se va a presentar la funcionalidad del ingreso al sistema, en el momento que cualquier usuario intente conectarse.

Tabla 2:

Caso de uso Validar transacción

Descripción	Validar la transacción que se desea ejecutar el usuario del sistema, de acuerdo al perfil que tenga asignado.
Objetivo	El objetivo de este caso de uso es que el sistema contemple las siguientes validaciones cada vez que un usuario intenta ejecutar una transacción: • Verificar la existencia de la transacción que se intenta ejecutar. • Validar que la transacción no exceda el monto permitido.

CONTINUA ⟹

	• Validar que el usuario se encuentre dentro de su horario de trabajo. Es decir, cada vez que cualquier usuario que se conecte al sistema sin importar el rol que tenga, el sistema debe revisar si el rol del usuario tiene permitido ejecutar la transacción a la que se está intentando ingresar, así como validar nuevamente si el horario en el que se intenta ejecutar la transacción es un tiempo válido de acuerdo a los horarios registrados para este usuario.
Actores	Administrador, Cajero, Jefe de Caja, Ejecutivo de Cuenta, Operador. Es decir cualquier persona que intenta ejecutar una transacción en el sistema.
Flujo de Eventos	
Flujo Básico	Ingresar el código del módulo y de la transacción con el que se desea trabajar.
Flujos Alternativos	
Precondiciones	Que el actor conectado al sistema haya sido validado exitosamente.
Condición de Éxito	El usuario puede ejecutar la transacción.
Condición de Fallo	El usuario no puede ejecutar la transacción y se emite un mensaje de error indicando el problema.

3.2.2.3 Especificación de caso de uso para gestionar roles

En la Tablas 3, 4, 5 y 6, se va a presentar la secuencia de acciones que conforman las funcionalidades que fueron identificadas en el rol Administrador.

Tabla 3:

Caso de uso Crear Rol

Descripción	Registrar Rol
Objetivo	Un rol sirve para delimitar las transacciones a las que el usuario del sistema va a tener acceso de acuerdo a su posición / función en la Cooperativa. Para esto, se debe registrar un rol, que consiste en ingresar un código y una descripción que definirá al rol, y a continuación se debe permitir registrar las transacciones que le serán permitidas el acceso en este rol, el monto máximo que puede manejar el rol del usuario en esa transacción y si esa transacción requiere autorización de otro rol.
Actores	Administrador
Flujo de Eventos	
Flujo Básico	Los pasos que involucra este caso de uso, son: • El administrador ingresa al sistema • Ingresa a la opción para crear roles • Registra el nuevo rol. • Asocia al rol las transacciones del sistema que le están permitidas a ese rol.
Flujos Alternativos	
Precondiciones	Debe estar conectado al sistema.
Condición de Éxito	Si los datos requeridos fueron ingresados completamente y son correctos, el sistema permitirá que se grabe el rol.

CONTINUA

Condición de Fallo	Si la información no está completa o uno de los datos no es correcto, el sistema debe emitir el correspondiente mensaje de error.

Tabla 4:

Caso de uso Consultar Rol

Descripción	Consulta rol de usuario
Objetivo	Poder consultar los roles creados en el sistema y las transacciones permitidas en cada rol.
Actores	Administrador/Gerente
Flujo de Eventos	
Flujo Básico	Ingresar a la transacción en modo de consulta, permitiendo visualizar todos los roles definidos en el sistema y las transacciones permitidas en cada rol. El sistema debe permitir también ingresar criterios de búsqueda ya sea por rol o por transacción. En el caso de que se consulte por transacción, ayudará para determinar cuáles son los roles que las contienen.
Flujos Alternativos	El rol que se intenta consultar no existe y el sistema emite mensaje de error.
Precondiciones	El usuario debe estar ingresado al sistema y tener permiso para realizar esta consulta.
Condición de Éxito	El sistema permite realizar la consulta
Condición de Fallo	El sistema emite mensaje de error indicando cuál fue la causa.

Tabla 5:

Caso de uso Modificar Rol

Descripción	Modificar Rol de Usuario
Objetivo	Permite consultar los roles creados en el sistema.
Actores	Administrador/Gerente
Flujo de Eventos	
Flujo Básico	• Ingresar a la transacción de Modificación del Rol. • Consultar el rol que se desea modificar. • Modificar la descripción del rol o las transacciones asociadas al rol. • Grabar los cambios hechos.
Flujos Alternativos	
Precondiciones	El usuario debe estar ingresado al sistema y tener permiso para realizar la modificación de datos del rol.
Condición de Éxito	El sistema valida los cambios, guarda los datos en la base de datos y emite un mensaje de éxito.
Condición de Fallo	El sistema emite mensaje de error indicando cuál fue la causa.

Tabla 6:

Caso de uso Eliminar Rol

Descripción	Eliminar Rol de Usuario
Objetivo	Permite eliminar un rol.
Actores	Administrador/Gerente

CONTINUA 

Flujo de Eventos	
Flujo Básico	• Ingresar a la transacción para eliminar el Rol. • Consultar el rol que se desea eliminar. • Borra el registro del rol. • Guarda el cambio.
Flujos Alternativos	
Precondiciones	El usuario debe estar ingresado al sistema y tener permiso para eliminar roles.
Condición de Éxito	El sistema debe validar el registro borrado, almacenar los datos en la base de datos y emitir un mensaje de éxito.
Condición de Fallo	El sistema va a validar si el rol que se desea eliminar, está asignado a un usuario, entonces el sistema no debe permitir borrar el rol. El sistema emite mensaje de error indicando el problema.

3.2.2.4 Especificación de caso de uso: Gestionar Usuario.

En las tablas 7, 8, 9 y 10, se van a presentar la secuencia de acciones existentes en la gestión de usuarios.

Tabla 7:

Caso de uso Registrar un Usuario

Descripción	Registrar en el sistema un nuevo usuario.
Objetivo	Registrar en el sistema a todos los usuarios que puedan interactuar con el sistema y definir el rol que van a desempeñar; teniendo en cuenta que un usuario debe ser un empleado registrado en el sistema, para lo cual se debe ingresar la

CONTINUA

	siguiente información: nombre y apellido, identificación, sexo, nacionalidad, profesión, fecha de nacimiento, fecha de ingreso, fecha de salida, rol asignado dentro del sistema, definición de horario de trabajo por día de la semana, clave de acceso encriptada y fecha de caducidad de su clave, Una vez que se registra al usuario, el sistema genera automáticamente el código de ingreso.
Actores	Administrador
Flujo de Eventos	
Flujo Básico	Los pasos a seguir son los siguientes: • El administrador registra los datos del usuario. • El administrador registra el horario de trabajo permitido para el usuario. • Graba el registro • El sistema genera un código de usuario que es entregado al respectivo usuario.
Flujos Alternativos	
Precondiciones	El administrador recibe la disposición para crear el nuevo usuario con los datos respectivos.
Condición de Éxito	Si todos los datos registrados son correctos, el usuario se graba y el sistema genera el código de acceso para aquel usuario registrado en el sistema.
Condición de Fallo	Si uno de los datos ingresados no es correcto o si la información no es completa, el sistema no permite que se cree el usuario y emite mensaje de error.

Tabla 8:

Caso de uso Consultar un Usuario

Descripción	Consulta de un usuario del sistema.
Objetivo	Poder consultar un usuario de acuerdo a los parámetros de búsqueda ingresados.
Actores	Administrador
Flujo de Eventos	
Flujo Básico	El administrador ingresa a la opción de consulta de usuarios, donde el sistema le muestra todos los usuarios permitidos en el sistema. Si desea buscar un usuario en especial, tiene la opción de consultar ingresando el código de usuario.
Flujos Alternativos	La búsqueda de usuarios puede ser global, es decir si no se introduce una condición de búsqueda el sistema muestra a todos los usuarios o la búsqueda puede ser individual.
Precondiciones	El administrador ingresa a la opción de consulta de usuarios. El sistema tienen la opción de mostrar todos los usuarios o de acuerdo a un criterio de búsqueda.
Condición de Éxito	Si la búsqueda es individual y se encuentra al usuario buscado, el sistema debe listar los datos del usuario. Si la búsqueda fue general el sistema lista los datos de todos los usuarios.
Condición de Fallo	Si la búsqueda es individual y el sistema no encuentra al usuario buscado, el sistema muestra el respectivo mensaje de error.

Tabla 9:

Caso de uso Modificar un Usuario

Descripción	Modificar datos de un usuario.
Objetivo	Dar mantenimiento a uno o más datos de un usuario.
Actores	Administrador
Flujo de Eventos	
Flujo Básico	Consultar un usuario en particular
Flujos Alternativos	Listar todos los usuarios y de ahí escoger el usuario que se desea modificar.
Precondiciones	Administrador ingresado en el sistema.
Condición de Éxito	Si los datos modificados con correctos y completos el sistema permite grabar los cambios.
Condición de Fallo	Si los datos no son correctos o completos el sistema emite un mensaje de error.

Tabla 10:

Caso de uso Eliminar Usuario

Descripción	Eliminar los datos de un usuario del sistema.
Objetivo	Si por cualquier razón se debe retirar el acceso al sistema a un usuario, se debe permitir cambiar el estado del usuario y poner una fecha límite de ingreso al sistema.
Actores	Administrador
Flujo de Eventos	
Flujo Básico	Consultar al usuario que se desea eliminar.
Flujos Alternativos	
Precondiciones	El usuario no debe tener transacciones monetarias para poder ser eliminado.

CONTINUA 

Condición de Éxito	Si el usuario a ser eliminado cumple las precondiciones, entonces se borra el registro del usuario y el sistema emite mensaje de error.
Condición de Fallo	Si el usuario no cumple con las precondiciones el sistema debe emitir mensaje de error.

3.2.2.5 Especificación de caso de uso del Rol Cajero

En las Tablas 11, 12, 13 y 14, se presentan los casos de uso de aquellas funcionalidades que realiza el Rol Cajero.

Tabla 11:

Caso de uso Abrir Caja

Descripción	Abrir caja para iniciar operaciones del cajero.
Objetivo	Habilitar a un cajero para que pueda iniciar su trabajo, recibiendo y/ o entregando dinero mediante depósitos y retiros, según los requerimientos de los socios.
Actores	Jefe de Cajas / Cajero
Flujo de Eventos	
Flujo Básico	Por un lado el Jefe de Cajas debe iniciar el proceso, registrando la cantidad de dinero que va a entregar al cajero. El cajero debe registrar en el sistema el valor que está recibiendo del Jefe de Caja y que es el saldo de efectivo con el que va a iniciar operaciones.
Flujos Alternativos	
Precondiciones	Tanto el Jefe de Caja como el Cajero deben tener sus usuarios creados en el sistema y los

CONTINUA 

	permisos necesarios para ejecutar la Apertura de Caja por parte del Cajero y el rol Bóveda debe tener permiso para registrar la Entrega de Efectivo.
Condición de Éxito	Si la transacción se ejecuta sin problemas, el sistema debe indicar que la transacción se ejecutó correctamente.
Condición de Fallo	Si la transacción presentó un error, el sistema debe mostrar un error claro que ayude a identificar el problema que se ha presentado.

Tabla 12:

Caso de uso Gestionar Depósitos

Descripción	Recepción de Depósitos en Efectivo
Objetivo	Recibir depósitos y registrar en las cuentas de los socios.
Actores	Cajero
Flujo de Eventos	
Flujo Básico	El flujo básico es el siguiente: • Ingresar a la transacción que permite registrar depósitos. • Ingresar el número de cuenta del socio a la que se le va a realizar el depósito. • Validar que el número de cuenta exista en el sistema y que pertenezca al socio que va a realizar el depósito. • Registrar el valor del depósito. • Incrementar el saldo de la cuenta.
Flujos Alternativos	

CONTINUA

Precondiciones	La cuenta en la que se va a realizar el depósito debe estar registrada en el sistema y en estado Activo.
Condición de Éxito	Si la transacción fue exitosa, el sistema debe mostrar un mensaje indicando que la transacción se aplicó correctamente.
Condición de Fallo	Si al intentar grabar la transacción, se produjo algún error, el sistema debe emitir un mensaje indicando el problema presentado, y por tanto no se debe grabar la transacción ni tampoco afectar el saldo de la cuenta.

Tabla 13:

Caso de uso Gestionar Retiros

Descripción	Registro de Retiros
Objetivo	Registrar retiros de dinero de las cuentas de los socios.
Actores	Cajero
Flujo de Eventos	
Flujo Básico	El flujo básico de este caso de uso es el siguiente: • Ingresar a la transacción de retiro. • Ingresar el número de cuenta del cual se va a realizar el retiro. • Registrar el valor a retirar. • Validar que la cuenta exista. • Validar que la cuenta tenga el saldo en efectivo mayor o igual al valor requerido, más la suma del valor mínimo en la cuenta, dependiendo de la parametrización del

CONTINUA 

	producto al que pertenece la cuenta de la que se va a realizar el retiro. • Registrar el retiro. • Afectar el saldo en efectivo de la cuenta.
Flujos Alternativos	
Precondiciones	La cuenta tiene que ser una cuenta válida y en estado activa para permitir el retiro. El saldo en efectivo de la cuenta debe ser mayor o igual a la suma del valor mínimo a mantener en la cuenta, más el valor del retiro.
Condición de Éxito	Si se cumple con las condiciones definidas en el ítem Precondiciones y si la transacción se graba sin generar problemas, el sistema debe emitir un mensaje indicando que el retiro se generó satisfactoriamente.
Condición de Fallo	Si no se cumple con las condiciones indicadas en el ítem Precondiciones o se produce algún tipo e error al grabar el retiro, el sistema debe mostrar un mensaje indicando claramente el problema que se presentó sin afectar ni la cuenta ni el saldo.

Tabla 14:

Caso de uso Cuadrar Caja

Descripción	Cuadre de Caja
Objetivo	Determinar si el saldo que tiene el cajero en efectivo como en cheques, es el valor esperado de acuerdo al saldo de apertura más las transacciones procesadas.
Actores	Cajero
Flujo de Eventos	
Flujo Básico	El flujo básico del caso de uso, es el siguiente: • Ingresar a la transacción de Cuadre de Caja. • Registrar el saldo resultante al final del día tanto para efectivo como para cheques. • Validar contra lo que el sistema tiene registrado. • Si el valor es correcto el sistema debe marcar a la caja como cerrada de modo que le permita hacer el cierre pertinente y así entregar el balance al Jefe de Caja. • Si el saldo que tiene el cajero supera al saldo registrado en el sistema, significa que el cajero tiene un sobrante. A pesar de que el cajero esté sobregirado, el sistema debe marcar a la caja como cuadrada para que pueda seguir el proceso de cierre. • Igualmente si el cajero tiene un saldo menor al esperado de acuerdo al registro de las transacciones realizadas por el cajero, a esta situación se le conoce como faltante, igualmente el sistema tiene que marcar a la

CONTINUA 

	caja como cuadrada para continuar el proceso de cierre.
Flujos Alternativos	
Precondiciones	La caja que se va a cuadrar debe haber pasado por el proceso de apertura de caja.
Condición de Éxito	Si la transacción se realizó correctamente, el sistema debe mostrar el respectivo mensaje indicando que la transacción se guardó satisfactoriamente.
Condición de Fallo	Si la transacción presentó algún tipo de error, el sistema debe mostrar el respectivo mensaje.

Tabla 15

Caso de uso Cerrar Caja

Descripción	Cierre de Caja
Objetivo	Registrar la entregar del saldo en efectivo que tiene el Cajero al Jefe de Caja.
Actores	Cajero / Jefe de Caja
Flujo de Eventos	
Flujo Básico	El flujo básico consta de los siguientes pasos: • Ingresar a la transacción de Cuadre de Caja. • Registrar el saldo resultante al final del día tanto para efectivo como para cheques. • Validar el efectivo que tiene el cajero con lo que el sistema tiene registrado. • Si el valor es correcto el sistema debe marcar a la caja como cerrada, de modo que le permita hacer el cierre pertinente y así entregar el balance al Jefe de Caja.
Flujos Alternativos	

CONTINUA

Precondiciones	La caja debe haber sido cuadrada antes de ejecutar esta transacción.
Condición de Éxito	Si la transacción se realizó correctamente, el sistema debe mostrar el respectivo mensaje indicando que la transacción se guardó satisfactoriamente.
Condición de Fallo	Si la transacción presentó algún tipo de error, el sistema debe mostrar el respectivo mensaje.

3.2.2.6 Especificación de caso de uso: Apertura de Bóveda

En las tablas 16, 17, 18 y 19, se presentan los casos de uso de las funcionalidades que realiza un Jefe de Caja.

Tabla 16

Caso de uso Abrir Bóveda

Descripción	Apertura de Bóveda
Objetivo	Permitir iniciar las operaciones de la bóveda y registrar el efectivo con el que arranca la bóveda.
Actores	Jefe de Caja
Flujo de Eventos	
Flujo Básico	Se debe registrar el valor con el que se inicia la bóveda y guardar el registro.
Flujos Alternativos	
Precondiciones	El Jefe de Caja debe ser un usuario válido en el sistema.
Condición de Éxito	La bóveda queda abierta para recibir operaciones.

CONTINUA 

Condición de Fallo	El sistema debe emitir el mensaje de error correspondiente y no permitir que se inicie operaciones con dicha bóveda.

Tabla 17

Caso de uso Recibir Dinero de Cajeros

Descripción	Recibir Dinero de Cajeros
Objetivo	Recibir el dinero de los cajeros, ya sea por concepto de cierre de caja o porque el cajero tiene un monto superior al valor permitido para manejar en caja.
Actores	Jefe de Caja Cajero
Flujo de Eventos	
Flujo Básico	El cajero realiza el cierre de caja o devolución de efectivo. El Jefe de Caja registra la recepción del efectivo.
Flujos Alternativos	
Precondiciones	El cajero debe haber realizado una de las dos transacciones indicadas en Flujo Básico.
Condición de Éxito	La transacción se registra sin ningún problema.
Condición de Fallo	El cajero no puede registrar la recepción del dinero.

Tabla 18

Caso de uso Entregar Dinero a Cajeros

Descripción	
Objetivo	Proveer de dinero a los cajeros para que puedan interactuar con los socios en el manejo de transacciones monetarias.
Actores	Jefe de Caja
Flujo de Eventos	
Flujo Básico	El Jefe de Caja define el valor con el que va a iniciar un cajero, que podría ser cero si el cajero solo va a recibir depósitos o con un valor de dinero específico, si el cajero va a atender retiros.
Flujos Alternativos	
Precondiciones	En bóveda debe existir un saldo mayor o igual al que se desea entregar al cajero.
Condición de Éxito	Se registra el valor a entregar al cajero por parte del Jefe de Caja y se guarda la transacción sin ningún problema.
Condición de Fallo	El valor que se desea entregar supera el valor existente en la bóveda. El sistema debe emitir el respectivo mensaje de error y deshacer la transacción.

Tabla 19

Caso de uso Autorizar Transacciones

Descripción	Autoriza Transacciones
Objetivo	Si el cajero requiere una autorización, como por ejemplo, al procesar transacciones fuera de horario o fuera del monto permitido, el Jefe de Caja es el encargado de emitir esa validación.
Actores	Jefe de Caja
Flujo de Eventos	
Flujo Básico	Si existen transacciones que requieran autorización, estas son pasadas al Jefe de Caja para que las autorice.
Flujos Alternativos	
Precondiciones	El cajero registra una transacción que requiere autorización, ya sea por monto o por estar fuera de horario.
Condición de Éxito	El Jefe de Caja aprueba el requerimiento de autorización y se guarda el registro de la transacción que se autorizó.
Condición de Fallo	La transacción no se autoriza por tanto esta no afecta saldos. El sistema notifica que no se realizó la autorización.

3.2.2.7 Especificación de caso de uso: Registrar Socios.

En las Tablas 21, 21 y 22, se presentan los casos de uso para las funcionalidades que realiza el Ejecutivo de Cuenta, en lo que tiene que ver con la Gestión de Socios.

Tabla 20

Caso de uso Registrar Socios

Descripción	Registro de un socio nuevo
Objetivo	Incluir en los registros de socios de la cooperativa a un nuevo socio, proporcionando como mínimo la información básica del socio, como: nombres, apellidos, número de cédula, sector, dirección, número de teléfono, y otros datos opcionales como profesión, sector económico, referencias, entre otros.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	• Ingreso al sistema. • Ingreso a la opción para crear nuevos socios • Ingreso de información del socio solicitada en el sistema.
Flujos Alternativos	
Precondiciones	
Condición de Éxito	Si la información está completa y los datos ingresados son correctos, el sistema grabará los datos del socio y mostrará mensaje indicando que el socio fue grabado correctamente.
Condición de Fallo	Si la información requerida no está completa o es incorrecta o se presenta cualquier otro tipo de error, el sistema debe emitir el respectivo mensaje informando el problema que se presentó y que el socio no fue guardado.

Tabla 21

Caso de uso Consultar Socios

Descripción	Consulta de Socios
Objetivo	Buscar un socio o varios de acuerdo a un criterio de búsqueda, y mostrar una lista con la información de aquellos socios que cumplan dicha condición.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	El flujo básico está compuesto por: • Ingresar un criterio de búsqueda. • Listar aquellos socios que cumplan la condición de búsqueda. • Si se desea ingresar a un nivel inferior con mayor detalle de información del socio.
Flujos Alternativos	
Precondiciones	
Condición de Éxito	Si luego de ejecutar la búsqueda en la base de datos, del o los socios que cumplan con el criterio ingresado, y si se encuentra datos, entonces el sistema listará a todos los registros que cumplen la condición; de ahí se podrá entrar a un nivel inferior donde se pueda visualizar toda la información del socio.
Condición de Fallo	Si no se encuentra socios que cumplan la condición de búsqueda, el sistema no listará nada y mostrará un mensaje indicando que no hay registros de socios que cumplan la condición.

Tabla 22

Caso de uso Modificar datos del socio

Descripción	Mantenimiento a los datos del socio
Objetivo	Modificar los datos del socio, dentro de este proceso se contempla el hecho de que un socio no puede ser eliminado, en su lugar se debe cambiar el estado del socio
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	• Ingresar a la opción que permite el mantenimiento de datos del socio. • Realizar una consulta para encontrar al socio requerido.
Flujos Alternativos	Salir de la transacción sin grabar los cambios.
Precondiciones	El socio debe estar registrado en el sistema.
Condición de Éxito	Si los datos que fueron modificados o eliminados son correctos y completos, el sistema guardará la información y mostrará un mensaje indicando que los cambios fueron guardados.
Condición de Fallo	Si la información modificada no es correcta o está incompleta o si se presenta algún otro tipo de error al guardar la información, el sistema debe mostrar el respectivo mensaje de error y no guardará los datos.

3.2.2.8 Especificación de caso de uso Gestionar Cuentas

En las tablas 23, 24, 25, 26, 27 y 28, se va a presentar la secuencia de acciones que conforman la funcionalidad para Gestionar Cuenta, que está a cargo del rol Ejecutivo de Cuenta.

Tabla 23

Caso de uso Crear Cuenta

Descripción	Creación de cuenta a la vista para un socio de la cooperativa.
Objetivo	Crear una cuenta a la vista, en el producto indicado en el ingreso de datos, para un socio en particular. Si el producto seleccionado tiene un saldo mínimo para abrir la cuenta, este saldo puede ser recibido; ya sea por transferencia desde otra cuenta del socio o por caja.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	El flujo básico contempla los siguientes puntos: • Ingresar a la opción de Creación de Cuentas. • Escoger el producto en el que se desea crear la cuenta. • Registrar los datos necesarios. • Si el producto en el que se está creando la cuenta, tiene parametrizado un saldo mínimo y se escogió recibir el dinero por caja, para que la cuenta se active se debe recibir el dinero en caja.
Flujos Alternativos	
Precondiciones	El socio debe estar registrado en el sistema.
Condición de Éxito	Si los datos provistos son completos y correctos, el sistema grabará la nueva cuenta y mostrará el número de cuenta asignado. Si la cuenta exige un valor mínimo, y se lo va a recibir por caja, el sistema debe mostrar un mensaje

CONTINUA

	indicando que debe terminar la transacción en caja, de lo contrario, si no exige saldo mínimo o el saldo mínimo fue recibido por transferencia, se debe marcar a la cuenta como activa.
Condición de Fallo	Si la información no es correcta o completa, el sistema debe mostrar el respectivo mensaje de error.

Tabla 24

Caso de uso Consultar Saldos

Descripción	Consulta de saldos del socio
Objetivo	Consultar los saldos que el socio mantiene en las cuentas que posee en la Cooperativa. Esta consulta permitirá escoger la cuenta e ingresar a consultar a un nivel más bajo; donde muestre el detalle de los movimientos en cada cuenta.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	El flujo básico consiste en: • Ingresar a la opción de consulta de saldos. • Ingresar el número de socio.
Flujos Alternativos	
Precondiciones	El socio debe estar registrado en el sistema y tener cuentas activas.
Condición de Éxito	Si el socio tiene cuentas activas, el sistema debe mostrar el saldo al momento de la consulta.
Condición de Fallo	Si el socio no tiene cuentas, no debe listar nada y mostrar un mensaje indicando esta situación.

Tabla 25

Caso de uso Transferir Dinero

Descripción	Transferencia de valores entre cuentas del socio o cuentas de terceros.
Objetivo	Mover el dinero entre cuentas del mismo socio o cuentas de otros socios dentro de la cooperativa.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	Los pasos a seguir son: • El socio manifiesta su necesidad de debitar valores de su cuenta y transferirla a otra cuenta; ya sea de su propiedad o de terceros. • Ingresar en la opción de transferencias. • Ingresar el número de cuenta a debitar que debe ser del socio, registrar el valor y el número de cuenta al que se desea acreditar. • El sistema debe validar si el saldo de la cuenta a debitar es mayor o igual al saldo a transferir, más el valor mínimo a mantener en la cuenta; de acuerdo a los parámetros del producto donde está creado la cuenta del socio.
Flujos Alternativos	
Precondiciones	El socio que solicita la transferencia debe ser el dueño de la cuenta a debitar. La cuenta debe tener un saldo mayor o igual a la suma del valor a transferir, más el valor mínimo a mantener en la cuenta; en caso de que

CONTINUA

	el producto exija un valor mínimo a mantener en la cuenta.
Condición de Éxito	Si se cumple con el saldo establecido en las precondiciones, el sistema puede grabar la transferencia afectando los saldos de las cuentas implicadas, y emitir un mensaje indicando que la transacción se efectuó correctamente.
Condición de Fallo	Si no cumple las condiciones establecidas en el literal precondiciones, entonces el sistema debe emitir el respectivo mensaje de error. O en caso de que una de las cuentas o las dos no existan en el sistema, igualmente debe dar un mensaje de error y no registrar la transferencia.

Tabla 26

Caso de uso Emitir Estado de Cuenta

Descripción	Emitir Estado de Cuenta
Objetivo	Emitir un reporte que contenga los movimientos de una cuenta específica de un socio.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	• Ingresar a la transacción para emitir estados de cuenta. • Registrar la cuenta de la que se desea imprimir el estado de cuenta. • Ejecutar el reporte e imprimir. • Se cobra cargo por la emisión del reporte.
Flujos Alternativos	

CONTINUA

Precondiciones	
Condición de Éxito	Se tiene la impresión del estado de cuenta del socio.
Condición de Fallo	El sistema no emite el reporte de estado de cuenta.

Tabla 27

Caso de uso Cerrar Cuenta

Descripción	Cierre de cuenta
Objetivo	Liquidar los saldos de una cuenta y marcar la cuenta como CERRADA.
Actores	Ejecutivo de Cuenta
Flujo de Eventos	
Flujo Básico	• Ingresar a la transacción para cerrar la cuenta. • Ingresar el número de cuenta que se desea cerrar. • El sistema determina el valor del devengo por pagar que tiene la cuenta. • Registrar el valor del devengo como interés capitalizado. • El socio indica la forma en la que se va a liquidar el saldo de la cuenta. Las opciones son: por ventanilla o transferencia a otra cuenta.
Flujos Alternativos	
Precondiciones	
Condición de Éxito	Si no se ha producido ningún error, y el socio escogió que el pago del saldo sea por transferencia, el sistema emite un mensaje

CONTINUA 

	indicando que la cuenta fue Cerrada. Si el pago del saldo se va a realizar por caja, el sistema debe emitir un mensaje indicando que la transacción va a terminar por caja.
Condición de Fallo	El sistema emite un mensaje de error indicando el problema que se ha presentado y no acredita el interés del valor devengado ni cierra la cuenta.

Tabla 28

Caso de uso Revertir Transacciones

Descripción	Revertir transacciones monetarias
Objetivo	En caso de que por algún error humano no detectado en el momento de registrar la transacción monetaria, o a petición del socio, se necesita revertir la transacción, el sistema debe permitir este proceso, registrando el movimiento contable contrario.
Actores	Ejecutivo de Cuenta / Cajero
Flujo de Eventos	
Flujo Básico	• Ingresar a la transacción para realizar reversos. • Consultar las operaciones monetarias realizadas. • Escoger la transacción a reversar. • Reversar la transacción y grabar.
Flujos Alternativos	
Precondiciones	
Condición de Éxito	Si la transacción se revierte y grabada sin problemas, el sistema debe mostrar un mensaje

CONTINUA

	indicando que se revirtió la transacción satisfactoriamente.
Condición de Fallo	Si la transacción revertida dio errores, entonces el sistema debe indicar claramente; cuál fue el error que se produjo y que no la transacción no fue revertida.

3.2.2.9 Especificación de caso de uso Rol Operador

A partir de la tabla 29 hasta la Tabla 33, se define las actividades que se deben ejecutar, para cumplir con las funcionalidades automáticas, que se llevan a cabo en el proceso de Cierre de Operaciones Diario.

Tabla 29

Caso de uso Calcular Provisión de Interés

Descripción	Calcular Provisión de Intereses
Objetivo	Realizar el cálculo de la provisión de interés, en base al saldo en efectivo que mantiene el socio en cada una de sus cuentas y almacenar dicho valor.
Actores	Proceso Automático
Flujo de Eventos	
Flujo Básico	• Correr el Proceso de cierre de operaciones diario. • En este proceso por lotes debe existir un sub-proceso que seleccione todas las cuentas activas con saldo efectivo mayor a cero y calcular el valor de la provisión diaria de acuerdo a la siguiente fórmula:

CONTINUA 

	$$prov = \frac{C * i}{BC * 100}$$ Donde: *Prov*: Valor de la provisión diaria *C*: Saldo en efectivo de la cuenta *i*: Tasa de interés que aplica a la cuenta *BC*: Base de Cálculo que se aplica a la cuenta • Este valor debe ir registrandose a diario y guardando el valor acumulado hasta la fecha de capitalización donde el acumulador se vuelve a iniciar a cero.
Flujos Alternativos	
Precondiciones	Las precondiciones son las siguientes: • Al iniciar el proceso de cierre de operaciones diarias, se debe confirmar que todas las cajas estén en estado CERRADO. • Debe ser una cuenta activa. • Debe ser una cuenta con un saldo efectivo mayor a cero.
Condición de Éxito	Se calcula el valor de la provisión y se almacena.
Condición de Fallo	El sistema registra el mensaje de error.

Tabla 30

Caso de uso Capitalizar Intereses

Descripción	Capitalizar Interés Ganado
Objetivo	Acreditar a la cuenta del socio el interés que se ha venido provisionando durante un período, proceso conocido como capitalización de interés.
Actores	Proceso automático.
Flujo de Eventos	
Flujo Básico	Dependiendo de la frecuencia de capitalización definida para el producto, y si es fin de mes, y cumple con la frecuencia indicada, entonces se busca todas las cuentas activas y por cada cuenta encontrada, se determina el valor acumulado de provisión para esa cuenta. Una vez determinado el valor de la provisión, se realiza un crédito a la cuenta que se está procesando y se re-inicia el valor del acumulador de la provisión para el nuevo período. Las frecuencias pueden ser las siguientes: • Diaria • Mensual • Trimestral • Semestral • Anual Dependiendo de la frecuencia, las fechas de capitalización son:

CONTINUA

	• Diaria -→ Cada día • Mensual → El último día de cada mes • Trimestral → Los últimos días de los meses de marzo, junio, septiembre y diciembre. • Semestral → Último día de los meses de junio y diciembre. • Anual → El último día del mes de diciembre
Flujos Alternativos	
Precondiciones	
Condición de Éxito	Se registra el crédito a la cuenta y se incrementa el saldo de la misma.
Condición de Fallo	Se registra en un log el motivo por el que falló la acreditación de interés.

Tabla 31

Caso de uso Calcular Cargos

Descripción	Cálculo de Cargos
Objetivo	Determinar el valor de los cargos que apliquen a una cuenta y realizar el débito de los mismos.
Actores	Proceso Automático
Flujo de Eventos	
Flujo Básico	• Correr el proceso de cierre de operaciones diario, donde exista un subproceso que determine los cargos que aplican a una cuenta, esto es en la fecha de corte de la cuenta que depende de la frecuencia con la que se parametrizó a la cuenta. • Si el valor del cargo es mayor al saldo efectivo de la cuenta, se deja al valor del cargo en un almacenamiento temporal de

CONTINUA 

	modo que el Proceso de Cierre de Operaciones que se realiza a diario, determine si existe saldo suficiente en la cuenta para realizar el cobro del cargo.
Flujos Alternativos	
Precondiciones	Existir cargos que se apliquen a las cuentas.
Condición de Éxito	Si el saldo es suficiente, se realiza el débito a la cuenta del socio, se registra el movimiento en la cuenta del socio y se realiza la respectiva afectación al saldo del socio.
Condición de Fallo	Si el saldo no es suficiente, el cargo y su valor se registran en un almacenamiento temporal, para que el proceso de cierre de operaciones día a día siga intentando realizar el cobro del cargo; hasta que la cuenta tenga un saldo mayor y poder efectuar el cobro.

Tabla 32

Caso de uso Inactivar Cuenta

Descripción	Inactivar Cuenta
Objetivo	Marcar a una cuenta como inactiva por no haber tenido movimiento, por parte del socio por un lapso de un año. Esto es de acuerdo a la ley ecuatoriana, la cual indica que si una cuenta no ha tenido movimiento desde hace un año atrás, se le marca a la como cuenta inactiva y el saldo se reporta en otras cuentas contables diferente a la de productos activos.

CONTINUA ⟹

Actores	Proceso Automático
Flujo de Eventos	
Flujo Básico	Correr el proceso de cierre de operaciones diario, en el que debe existir un subproceso, que todos los días busque todas las cuentas activas, en las cuales la última fecha en la que el socio movió su cuenta, es mayor al año, entonces, la cuenta tiene que ser reclasificada a un nuevo producto de cuentas inactivadas, y marcar su estado como inactivo para que no permita realizar movimientos con esta cuenta.
Flujos Alternativos	
Precondiciones	La cuenta no debe presentar movimientos por parte del socio desde hace un año o más.
Condición de Éxito	Si se cumple las condiciones de que la cuenta sea activa y que la fecha de último movimiento hecho por el socio sea mayor a un año, se le cambia a la cuenta el producto al que pertenece y se modifica el estado de la misma a Inactiva.
Condición de Fallo	Si no se cumplen estas condiciones, la cuenta se desecha, lo cual significa que la cuenta si tuvo movimientos, por tanto es una cuenta activa. Si la cuenta cumple las condiciones para inactivar y sin embargo al intentar hacer el proceso de inactivación el sistema presenta un error, se debe registrar el error en una bitácora de errores, que sea revisada por el Operador y el Ejecutivo de Cuenta, de ser necesario.

Tabla 33

Caso de uso Emitir Estados de Cuenta

Descripción	Emitir Estados de Cuenta
Objetivo	Permite generar reportes con los estados de cuenta para los socios.
Actores	Proceso automático
Flujo de Eventos	
Flujo Básico	Si la fecha en la que se ejecuta el proceso de cierre de operaciones diario es igual a una fecha de corte, entonces el proceso debe buscar todas las cuentas activas en el sistema que pertenezcan a productos; donde se indica que generan estado de cuenta. Una vez obtenida esta lista, se generan reportes que contengan los movimientos de la cuenta desde la última fecha de corte hasta la fecha de corte actual.
Flujos Alternativos	Este proceso puede ser generado a nivel de cuenta por el Ejecutivo de Cuenta a petición del socio.
Precondiciones	• Que sea fecha de corte. • Cuenta activa en el sistema. • El producto al que pertenece la cuenta es un producto que genera estados de cuenta.
Condición de Éxito	Se imprime los estados de cuenta
Condición de Fallo	No se imprime los estados de cuenta y se genera un alerta para que el operador revise y de solución al problema.

3.2.3 Diagramas de Clases

Los diagramas de clase son importantes porque brindan una visualización del modelo estructural, permitiendo así mirar las clases y sus relaciones.

En la Figura 19, se muestra las clases y las relaciones que se han identificado, para cubrir las necesidades del Sistema Financiero Cooperativo.

3.2.4 Diseño Lógico de la Base de Datos

Basado en el Diagrama de Clases, se obtuvo el Diagrama Lógico, que está expuesto en la Figura 20.

3.2.5 Diseño Físico de la Base de Datos

Basado en el Diagrama de Lógico de la base de datos, se obtuvo el Diagrama Físico, que está expuesto en la Figura 21.

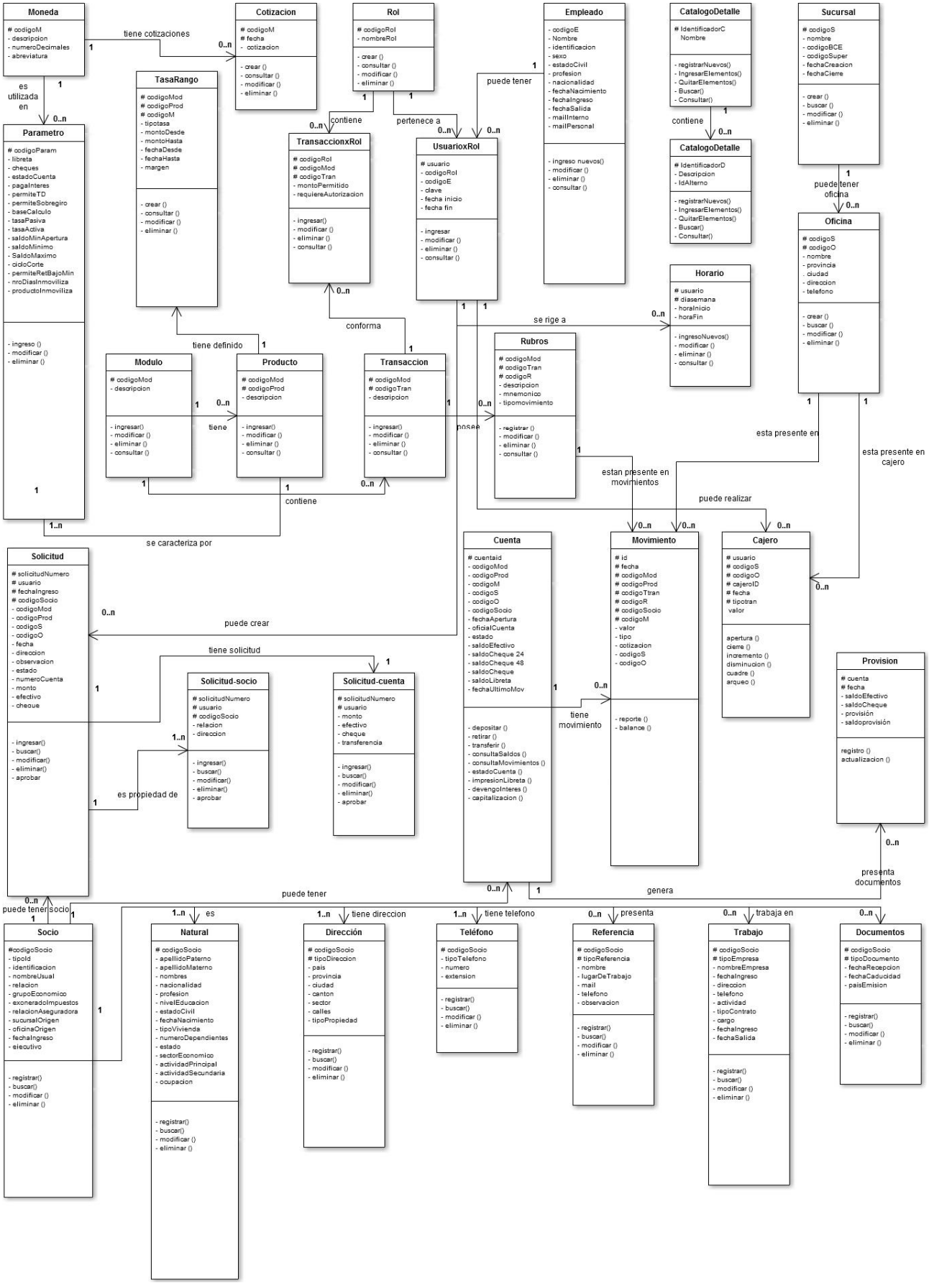

Figura 19: Diagrama de Clases del Sistema Financiero Cooperativista

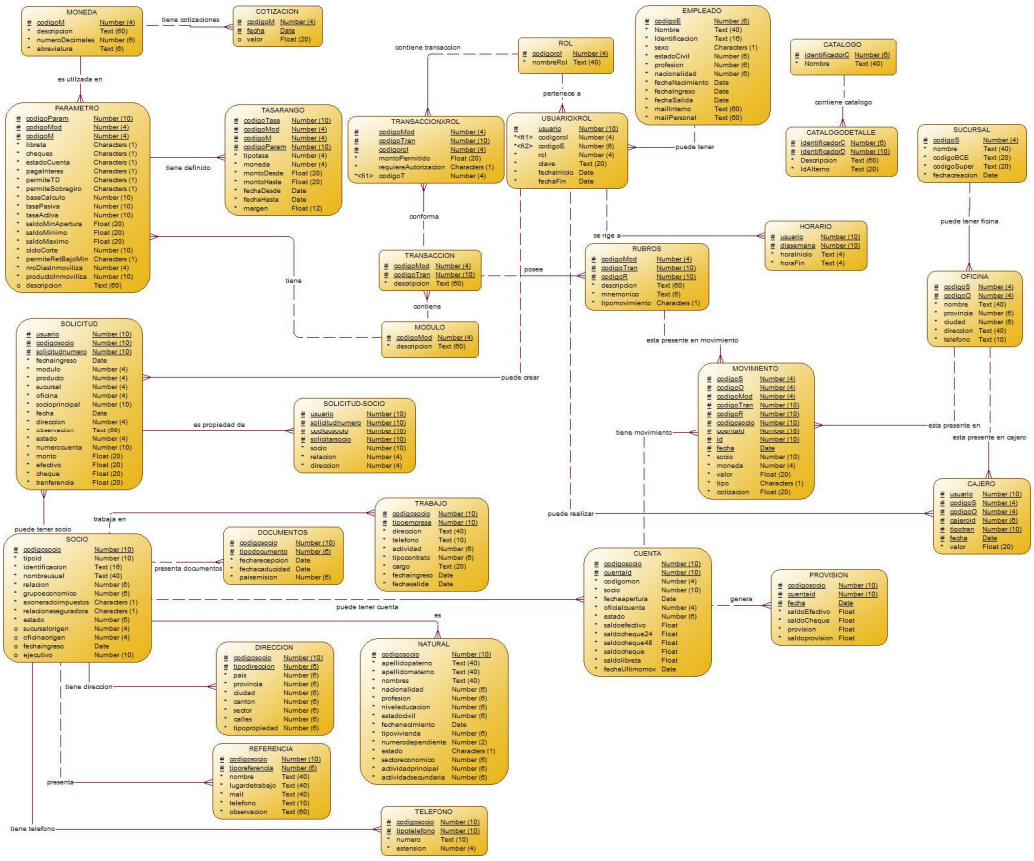

Figura 20: Diseño Lógico

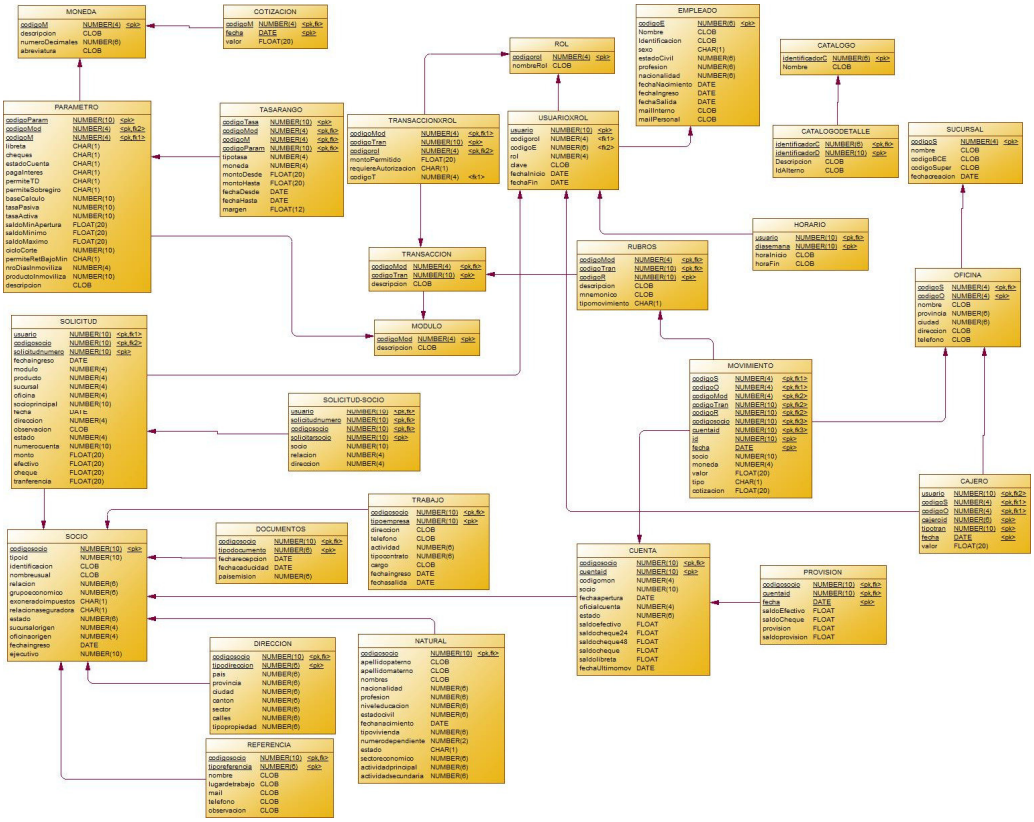

Figura 21: Diseño Físico de la Base de Datos

3.2.6 Diagrama de Secuencia

La importancia de los diagramas de secuencia radica en que permite determinar y modelar la interacción entre los objetos que se utilizan a través del tiempo para implementar el sistema, es decir, este diagrama muestra el conjunto de pasos que se deben llevar a cabo para lograr el funcionamiento correcto del sistema.

En la Figura 22 y 23 se ha graficado los diagramas de secuencia para los casos de uso general que son la validación del ingreso al sistema y la validación de ingreso a una transacción.

3.2.6.1 Diagrama de secuencia del caso de uso Ingresar al Sistema

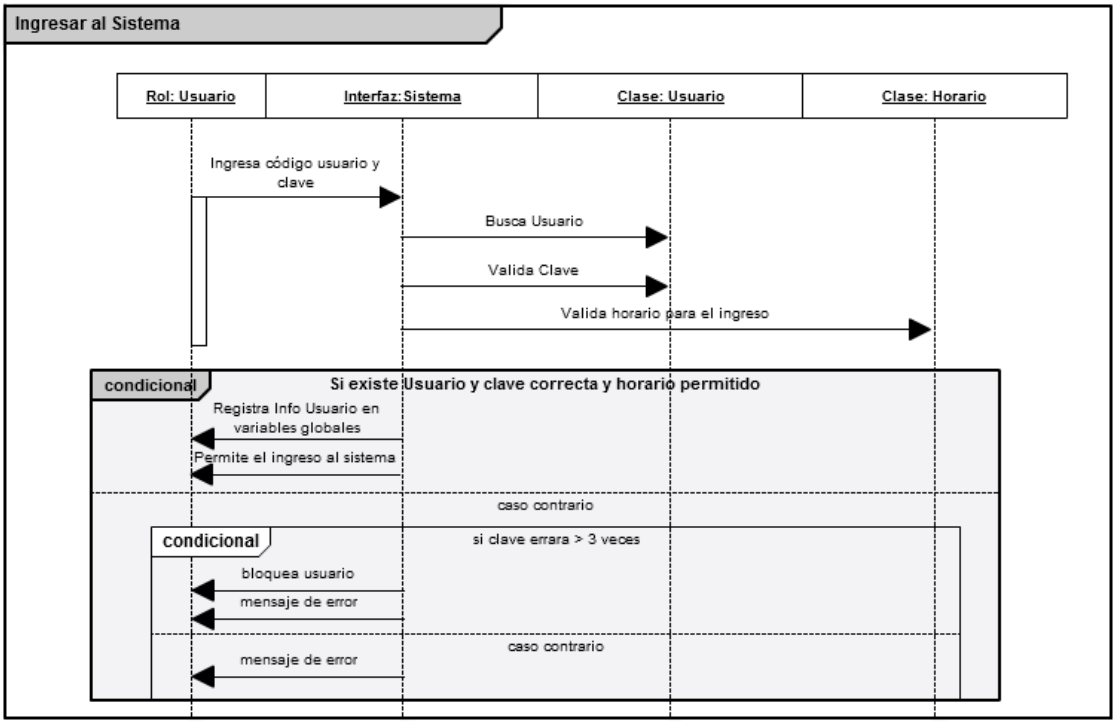

Figura 22: Diagrama de secuencia del caso de uso Ingreso al Sistema

3.2.6.2 Diagrama de secuencia del caso de uso Validar Transacción

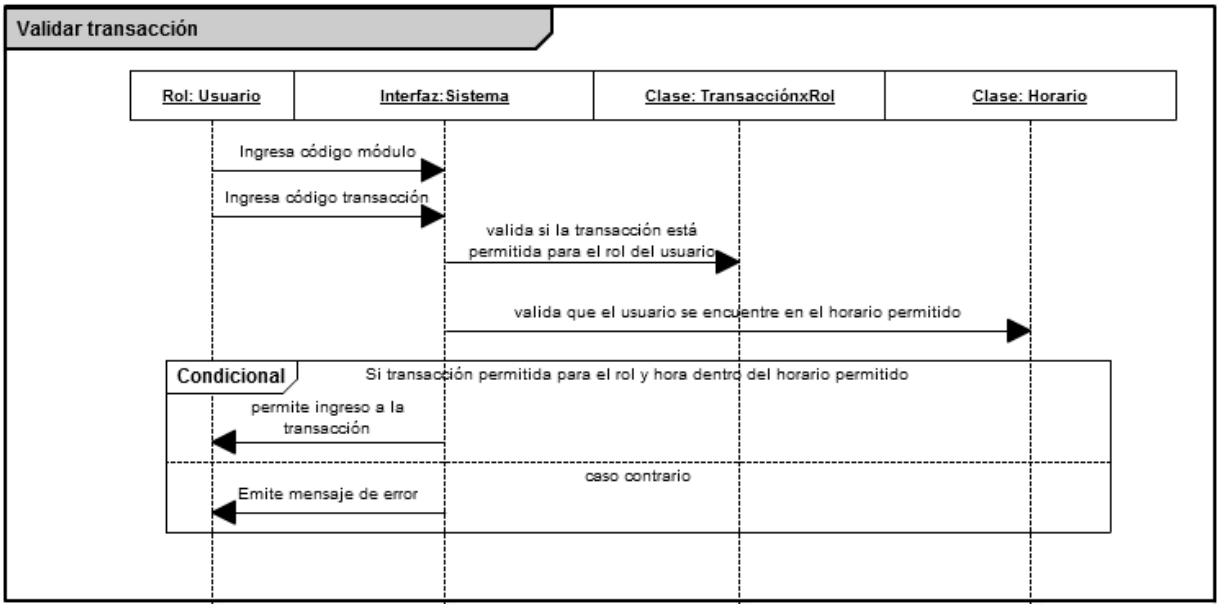

Figura 23: Diagrama de secuencia del caso de uso Validar Transacción

A continuación, en las Figuras 24, 25, 26 y 27, se presentan los diagramas de secuencia pertenecientes al Rol Administrador, en lo que respecta a la gestión de roles.

3.2.6.3 Diagrama de secuencia del caso de uso Registrar Rol

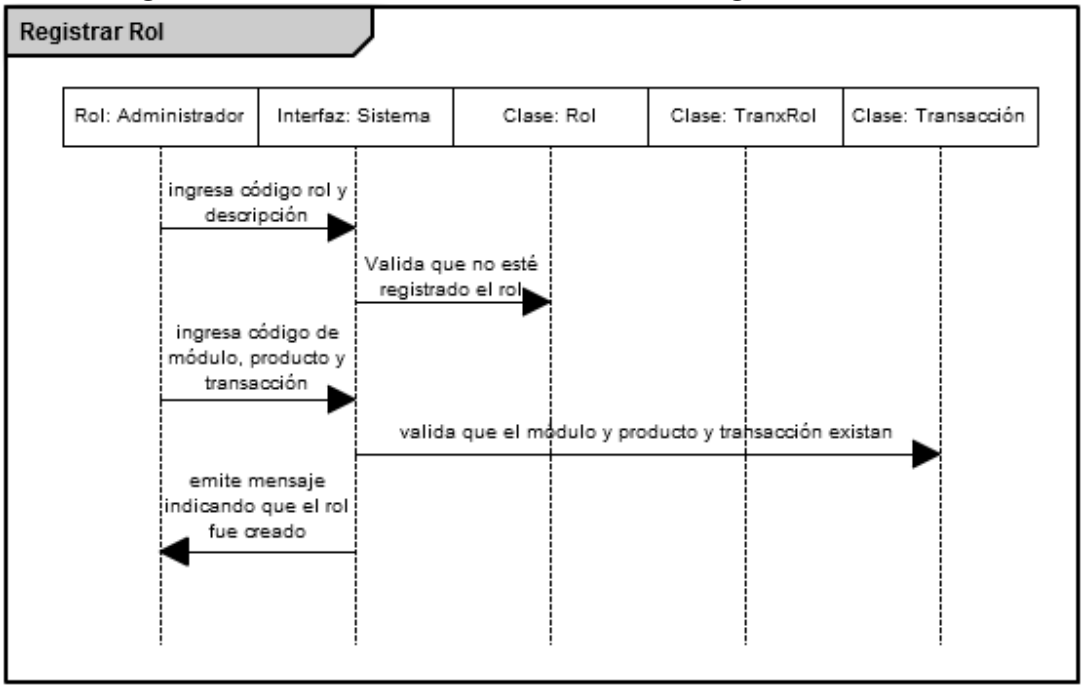

Figura 24: Diagrama de Secuencia del caso de uso Registro del Rol

3.2.6.4 Diagrama de secuencia del caso de uso Consultar Rol

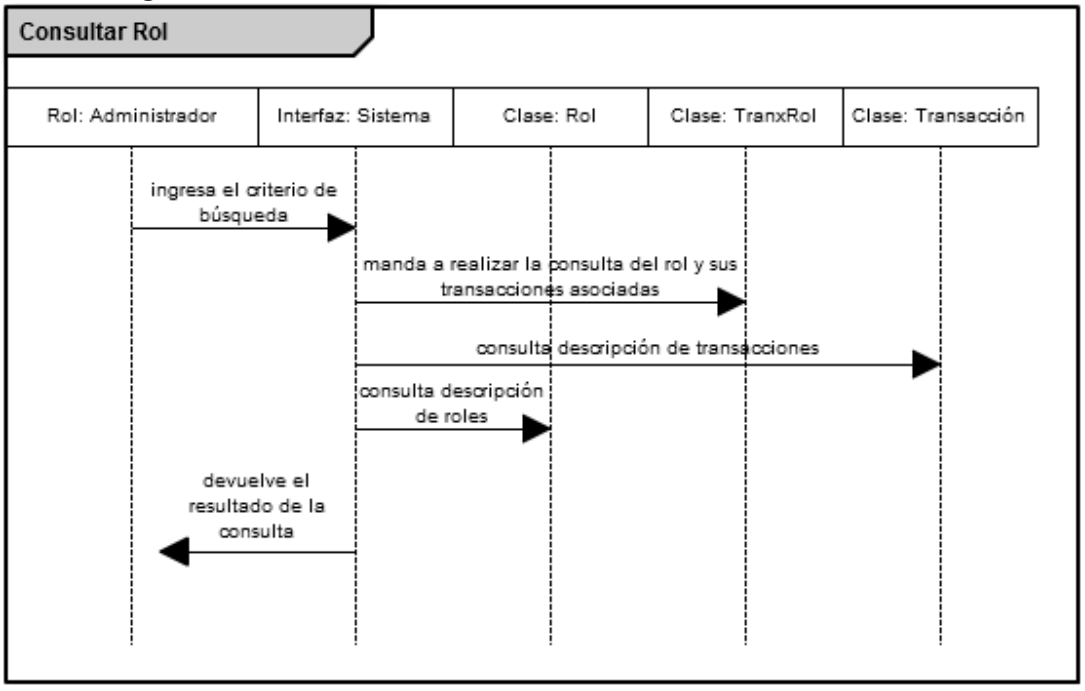

Figura 25: Diagrama de secuencia del caso de uso Registro del Rol

3.2.6.5 Diagrama de secuencia del caso de uso Modificar datos del Rol

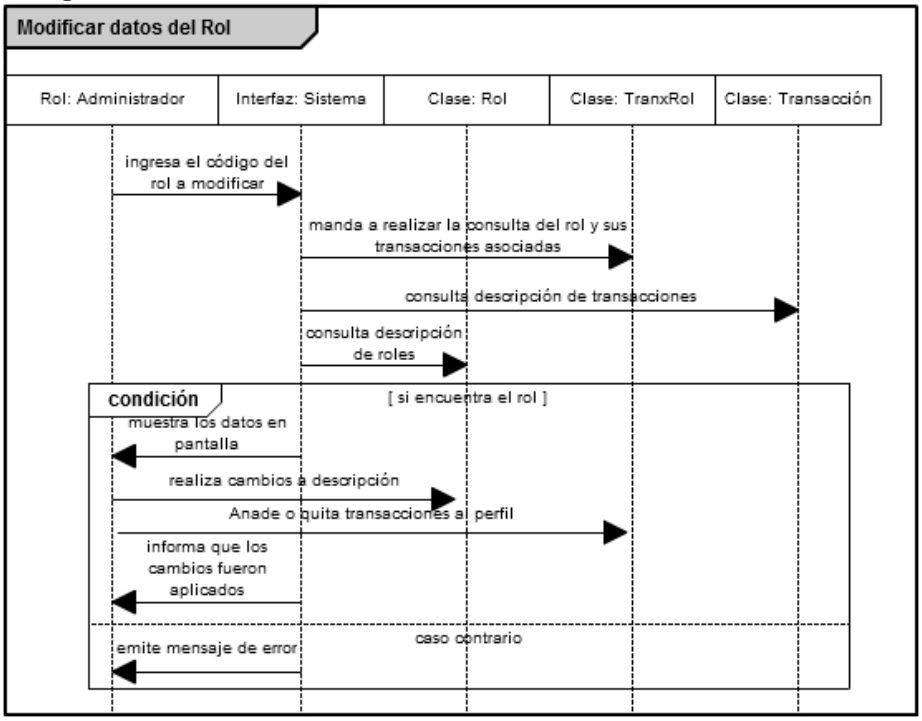

Figura 26: Diagrama de secuencia del caso de uso Registro del Rol

3.2.6.6 Diagrama de secuencia del caso de uso Eliminar Rol

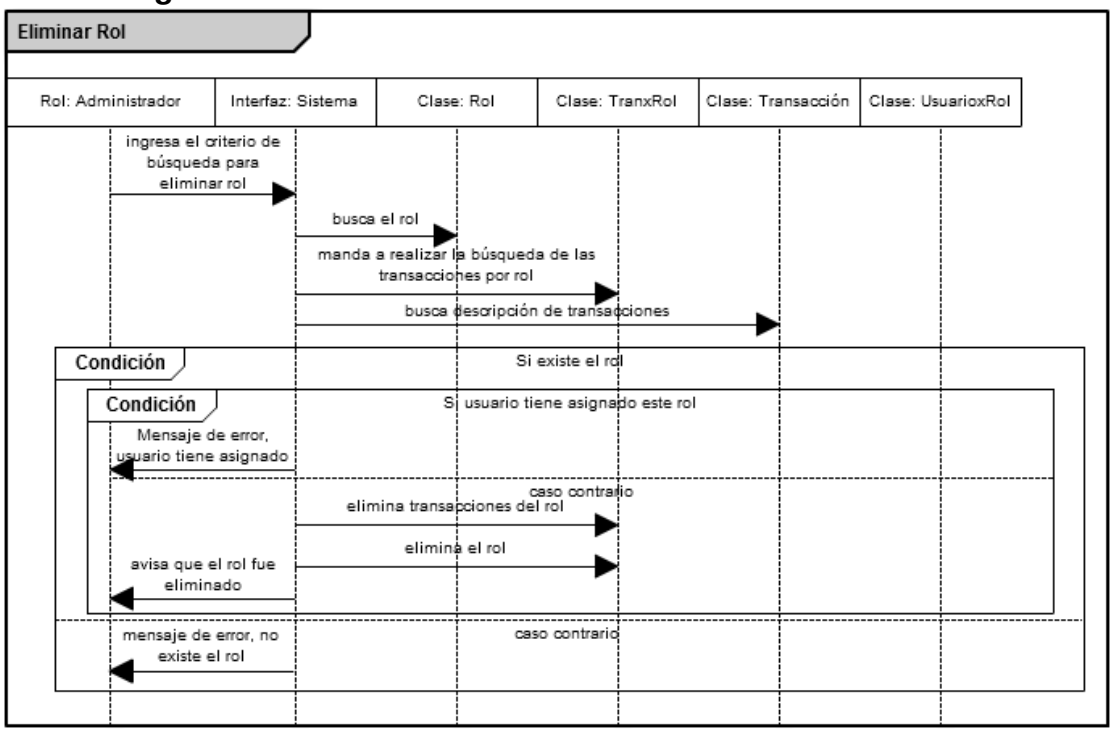

Figura 27: Diagrama de secuencia del caso de uso Registro del Rol

En el siguiente grupo de Figuras que van del 28 al 31, se presenta los Diagramas de Secuencia del Rol Administrador, en los procesos relacionados a la Gestión de Usuarios.

3.2.6.7 Diagrama de secuencia del caso de uso Registrar Usuario

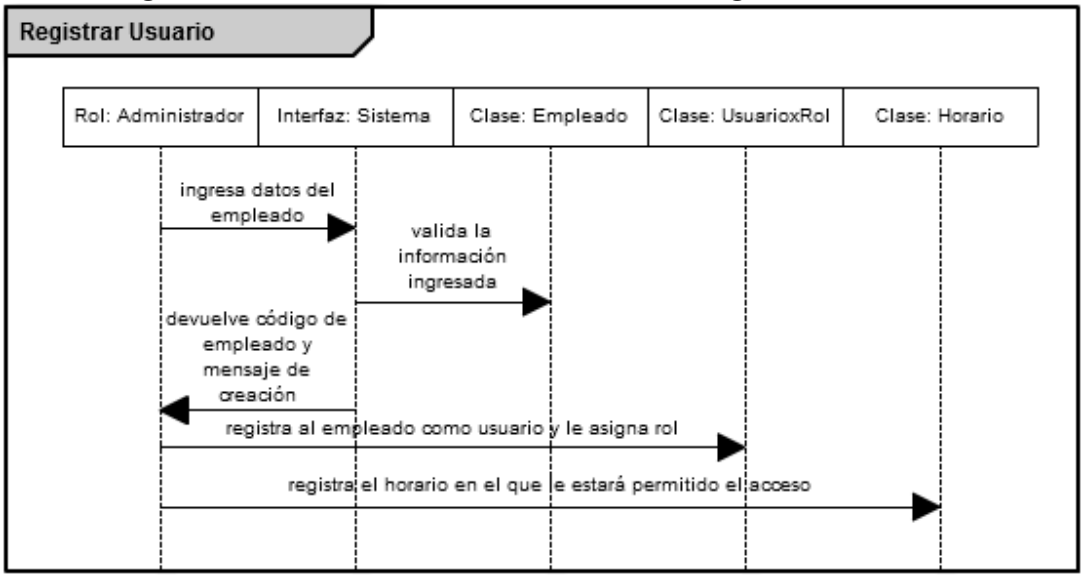

Figura 28: Diagrama de secuencia del caso de uso Registrar Usuario

3.2.6.8 Diagrama de secuencia del caso de uso Consultar Usuario

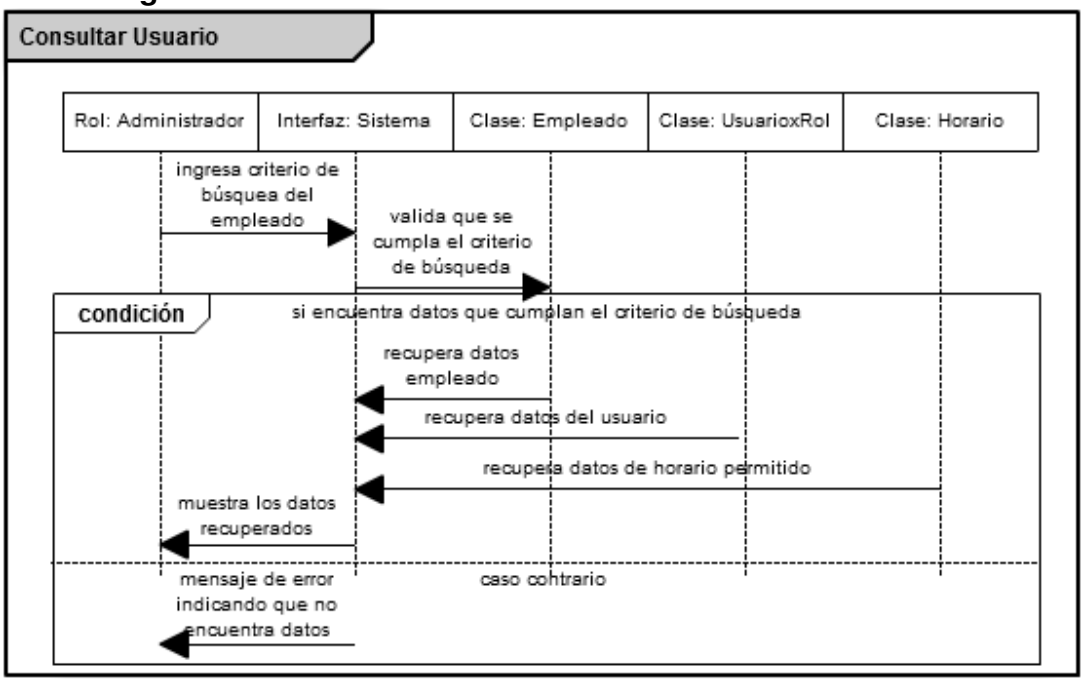

Figura 29: Diagrama de secuencia caso de uso Consultar Usuario

3.2.6.9 Diagrama de secuencia del caso de uso Modificar Usuario

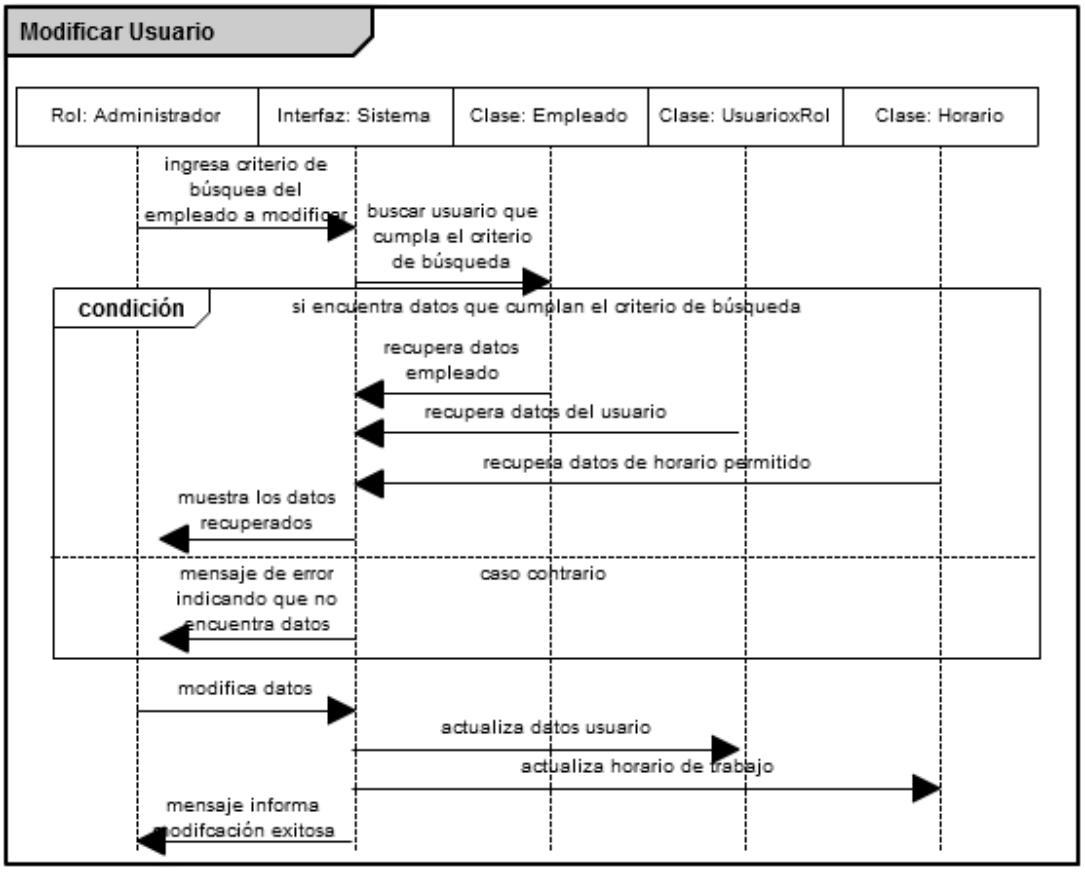

Figura 30: Diagrama de secuencia del caso de uso Modificar Usuario

3.2.6.10 Diagrama de secuencia del caso de uso Eliminar Usuario

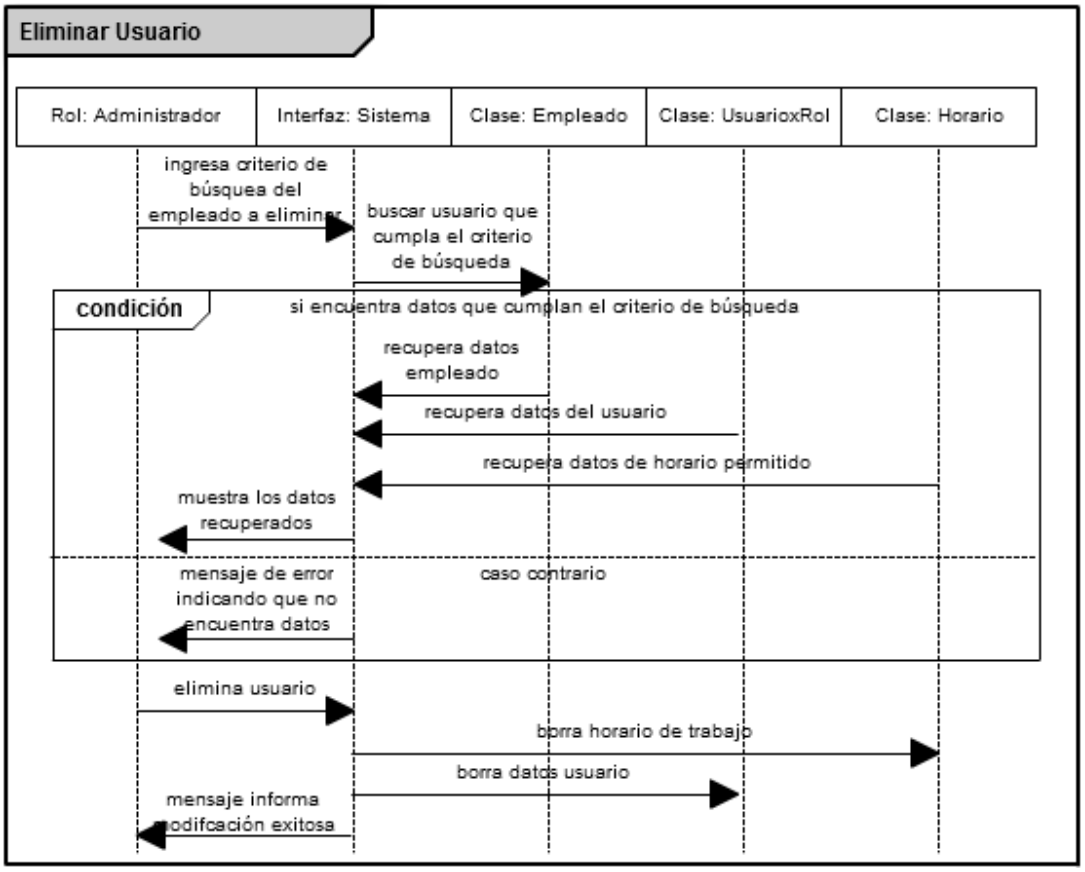

Figura 31: Diagrama de secuencia del caso de uso Eliminar Usuario

A partir de la Figura 32 hasta la Figura 35, se presentan los Diagramas de Secuencia, del proceso para Gestionar Productos, perteneciente al Rol Administrador.

3.2.6.11 Diagrama de secuencia del caso de uso Crear Producto

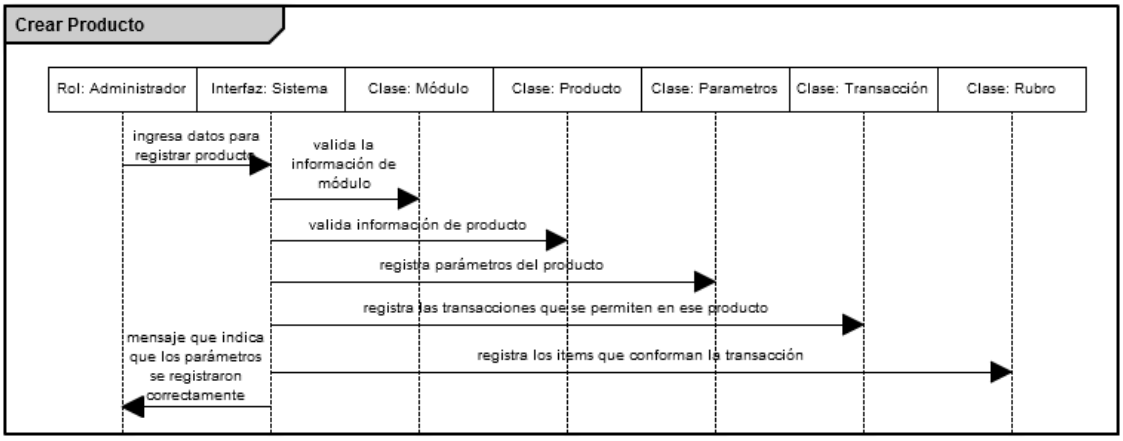

Figura 32: Diagrama de secuencia del caso de uso Crear Producto

3.2.6.12 Diagrama de secuencia del caso de uso Consultar Producto

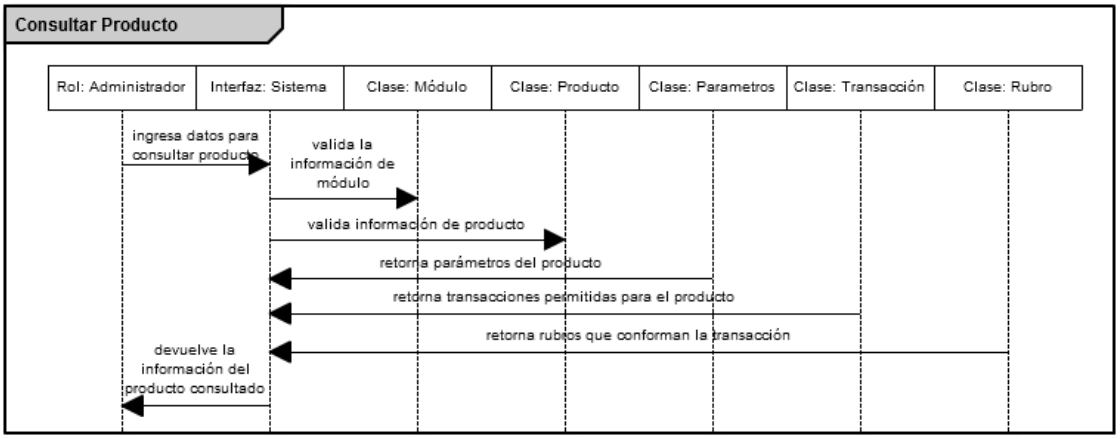

Figura 33: Diagrama de Secuencia de caso de uso Consultar Producto

3.2.6.13 Diagrama de secuencia del caso de uso Modificar Producto

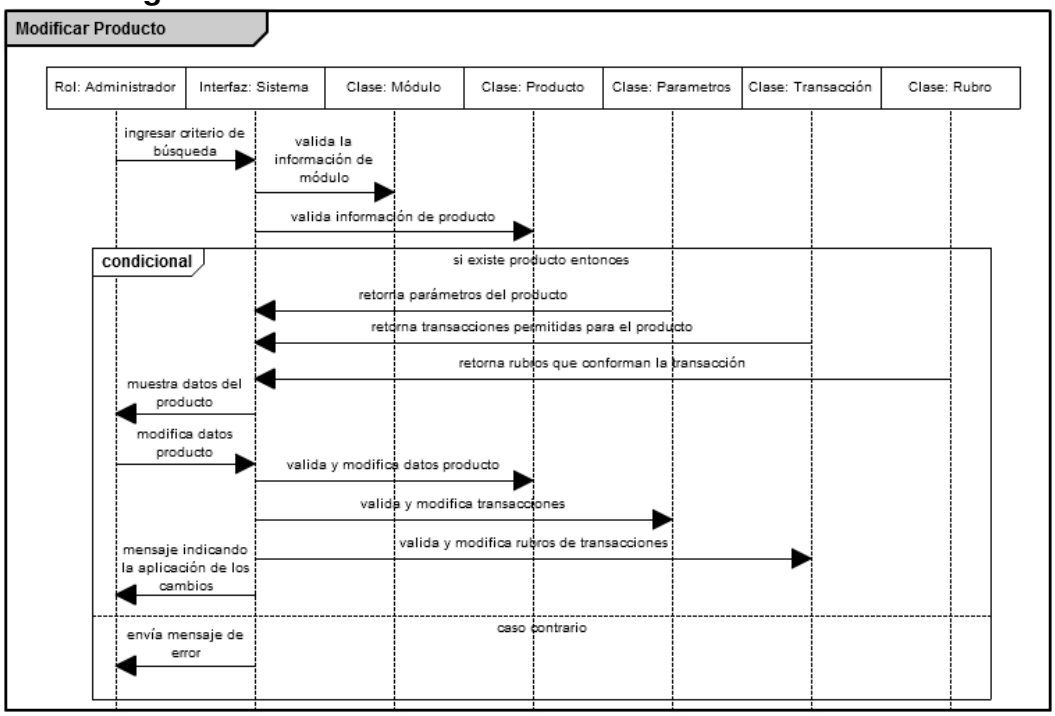

Figura 34: Diagrama de secuencia de caso de uso Modificar Producto

3.2.6.14 Diagrama de secuencia del caso de uso Eliminar Producto

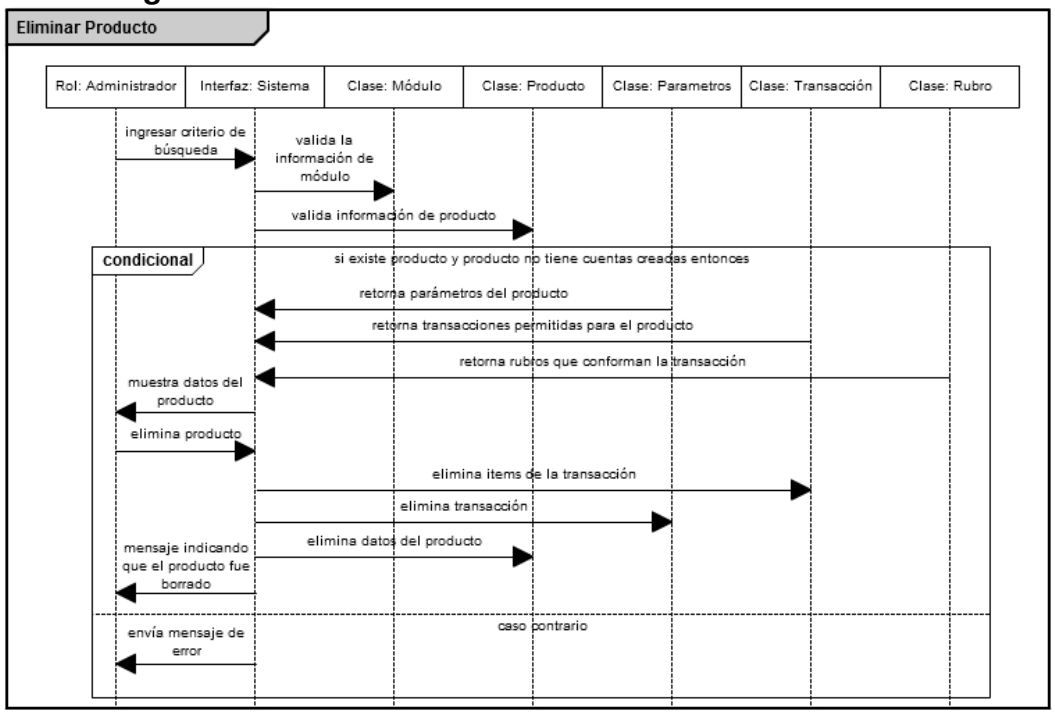

Figura 35: Diagrama de secuencia de caso de uso Eliminar Producto

A continuación, en las Figuras 36, 37, 38 y 39, se presentan los Diagramas de Secuencia, que muestran la lógica a seguir, en los procesos que permiten la Gestión de Tasa, dentro del Rol Administrador.

3.2.6.15 Diagrama de secuencia del caso de uso Crear Tasa

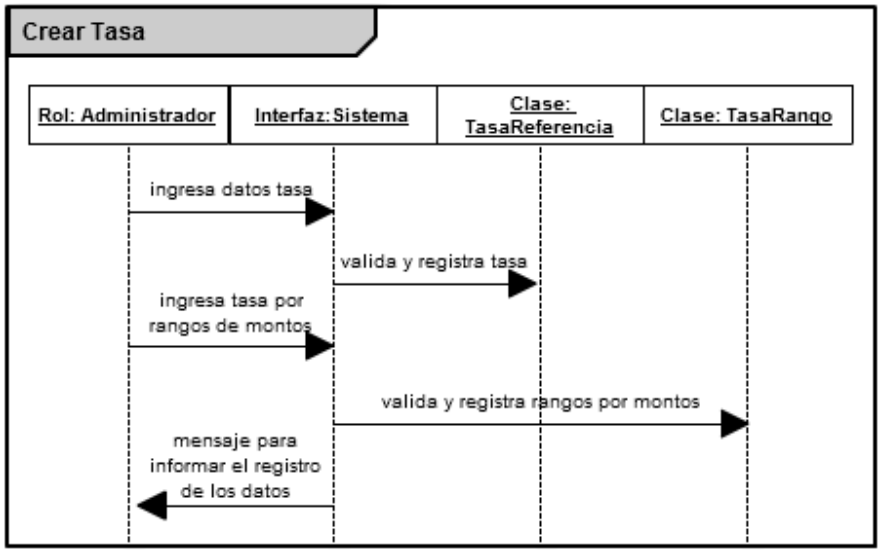

Figura 36: Diagrama de secuencia del caso de uso Crear Tasa

3.2.6.16 Diagrama de secuencia del caso de uso Consultar Tasa

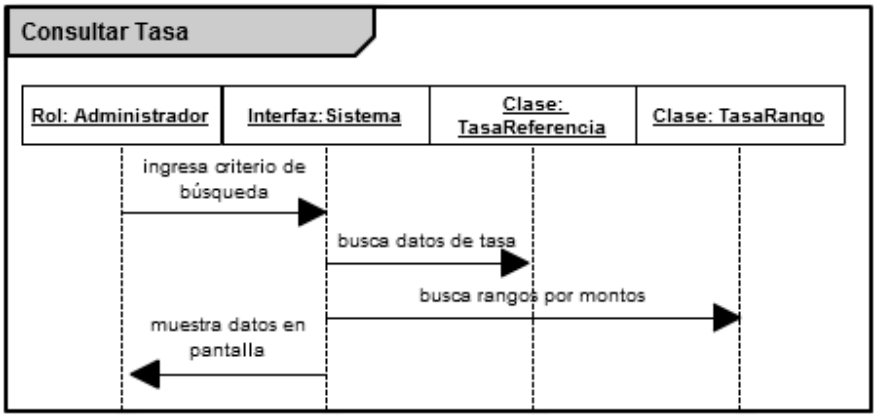

Figura 37: Diagrama de secuencia del caso de uso Consultar Tasa

3.2.6.17 Diagrama de secuencia del caso de uso Modificar Tasa

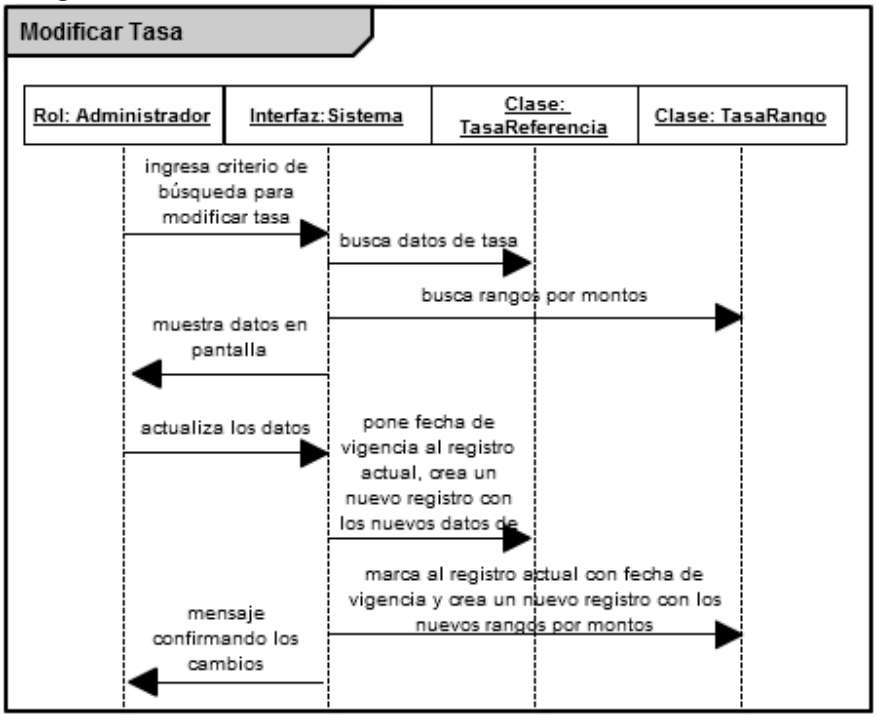

Figura 38: Diagrama de secuencia del caso de uso Modificar Tasa

3.2.6.18 Diagrama de secuencia del caso de uso Eliminar Tasa

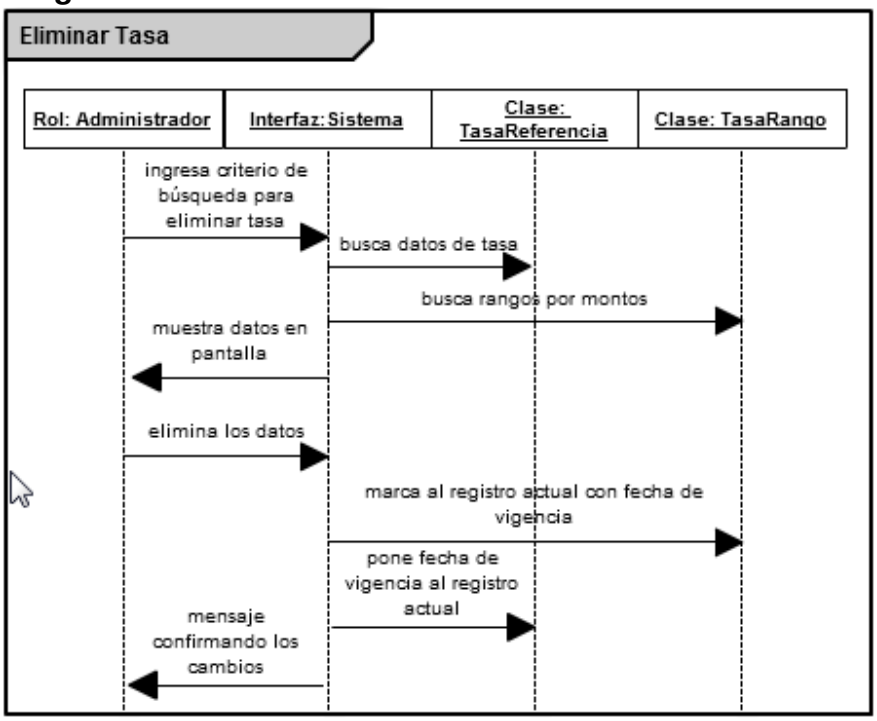

Figura 39: Diagrama de secuencia del caso de uso Eliminar Tasa

Desde la Figura 40 hasta la Figura 43, se presentan los Diagramas de Secuencia, que describen los procesos para Gestionar los catálogos del sistema, que son ampliamente utilizados en todo el sistema, para evitar el ingreso de aquella información que puede ser catalogada en listas de valores.

3.2.6.19 Diagrama de secuencia del caso de uso Crear Catálogo

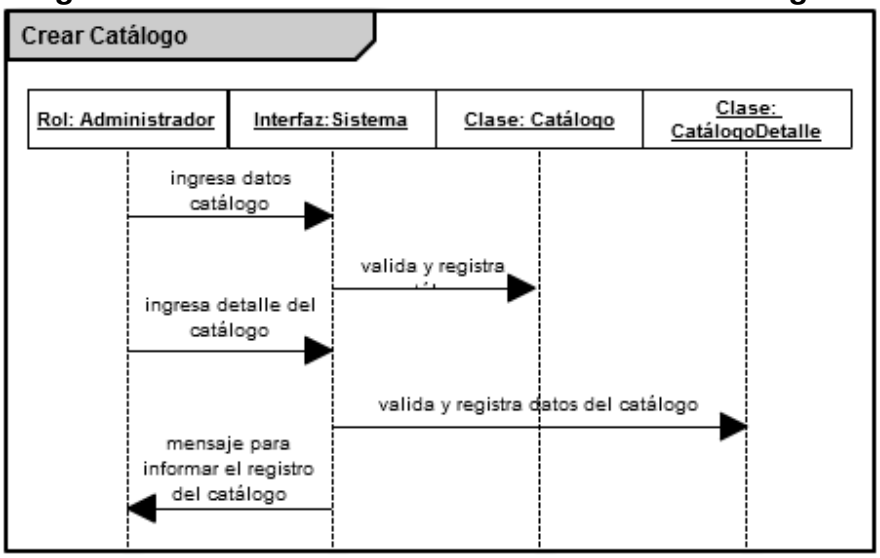

Figura 40: Diagrama de secuencia del caso de uso Crear Catálogo

3.2.6.20 Diagrama de secuencia del caso de uso Consultar Catálogo

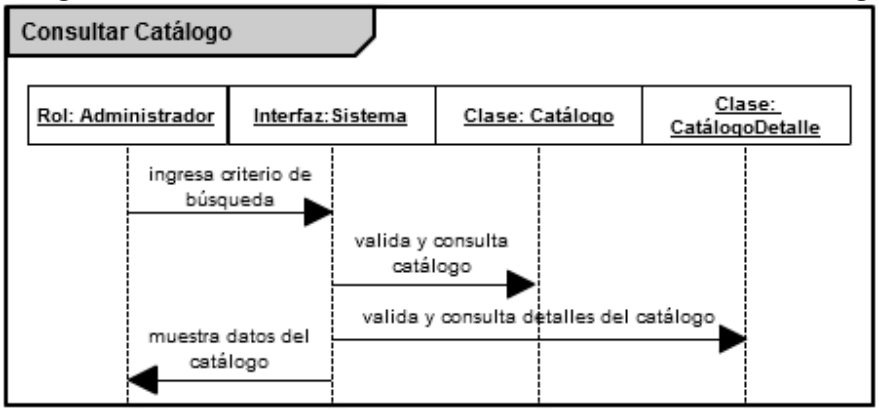

Figura 41: Diagrama de secuencia del caso de uso Consultar Catálogo

3.2.6.21 Diagrama de secuencia del caso de uso Modificar Catálogo

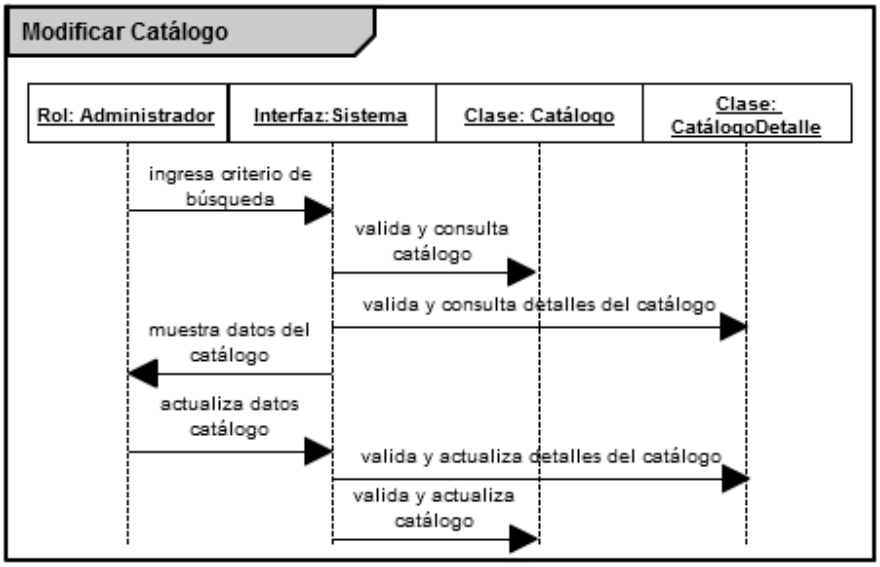

Figura 42: Diagrama de secuencia del caso de uso Modificar Catálogo

3.2.6.22 Diagrama de secuencia del caso de uso Eliminar Catálogo

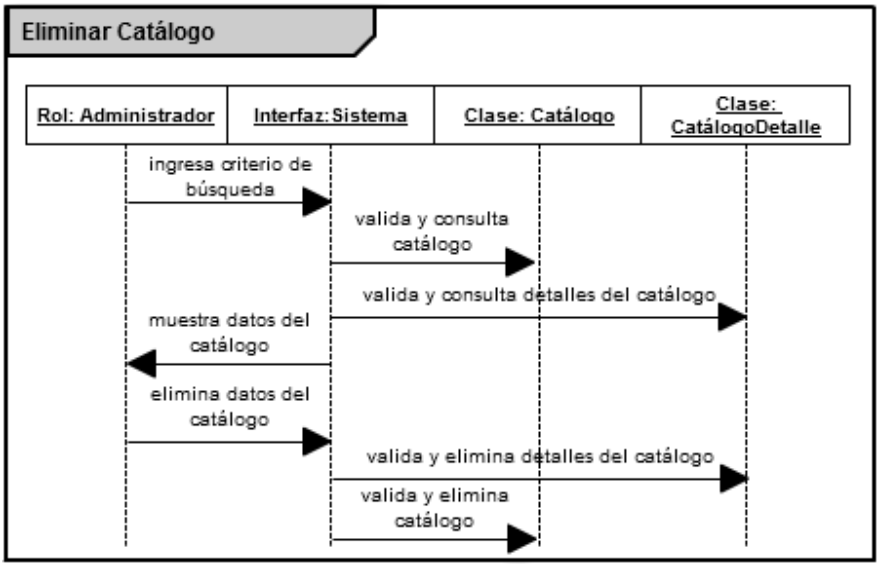

Figura 43: Diagrama de secuencia del caso de uso Eliminar Catálogo

A continuación, desde la Figura 44 hasta la Figura 50, se presentan los diagramas de Secuencia que representan los procesos que realiza el cajero.

3.2.6.23 Diagrama de secuencia del caso de uso Abrir Caja

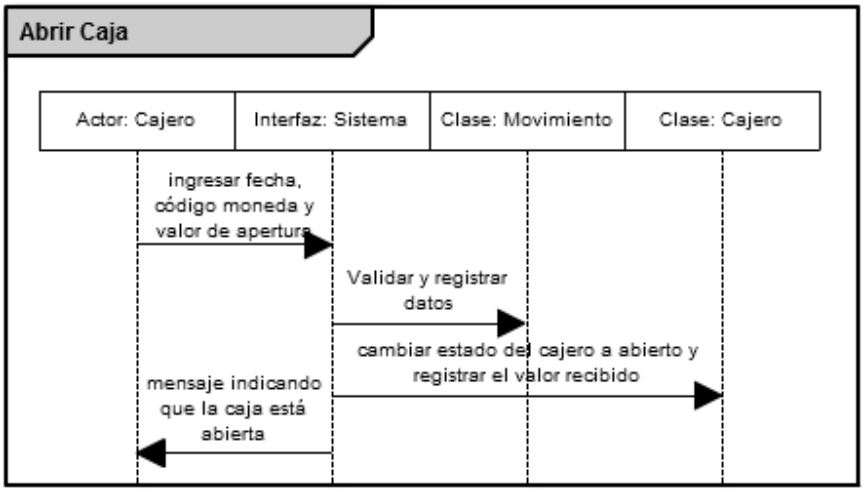

Figura 44: Diagrama de secuencia del caso de uso Abrir Caja

3.2.6.24 Diagrama de secuencia del caso de uso Gestionar Depósitos

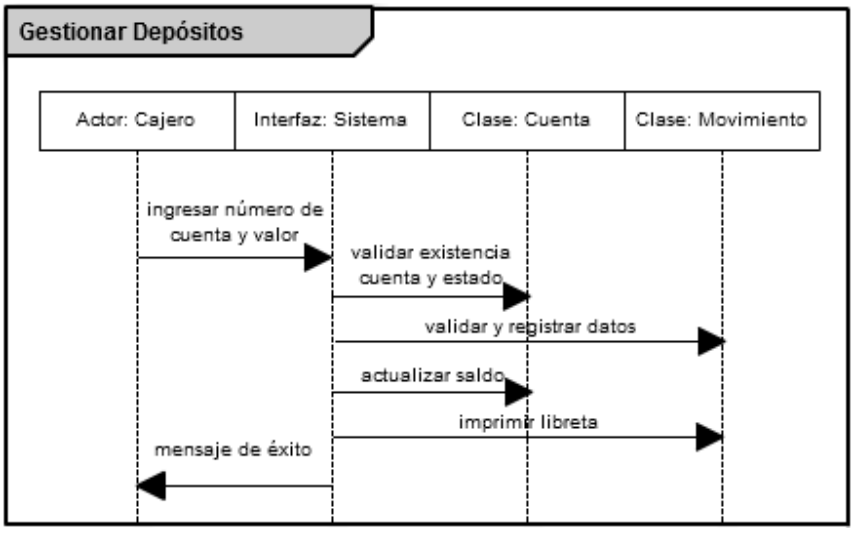

Figura 45: Diagrama de secuencia del caso de uso Gestionar Depósitos

3.2.6.25 Diagrama de secuencia del caso de uso Gestionar Retiros

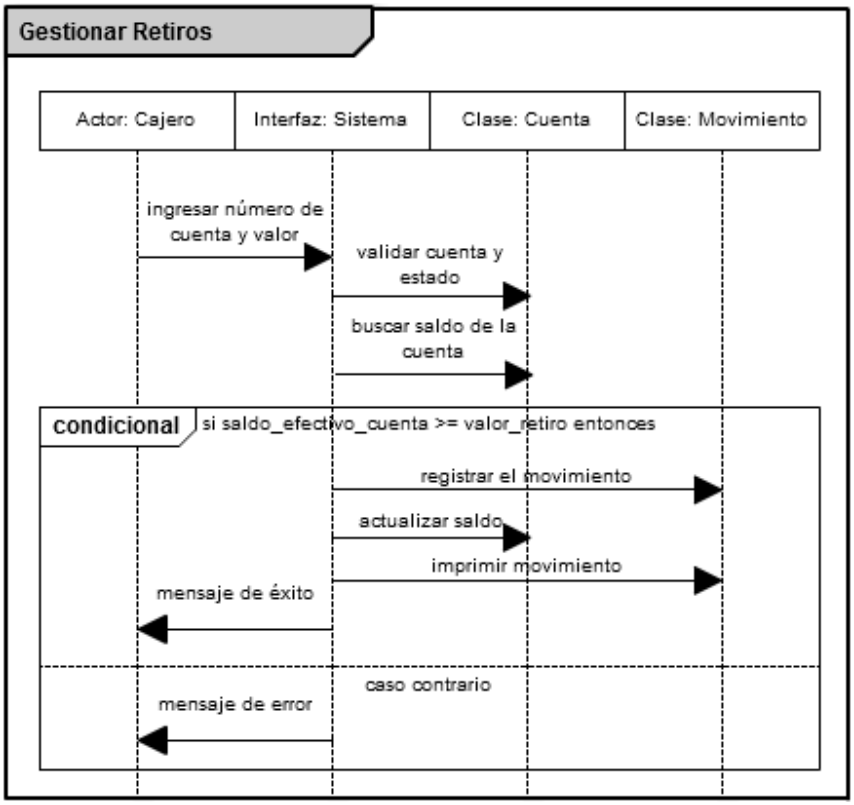

Figura 46: Diagrama de secuencia del caso de uso Gestionar Retiros

3.2.6.26 Diagrama de secuencia del caso de uso Revertir transacciones Monetarias

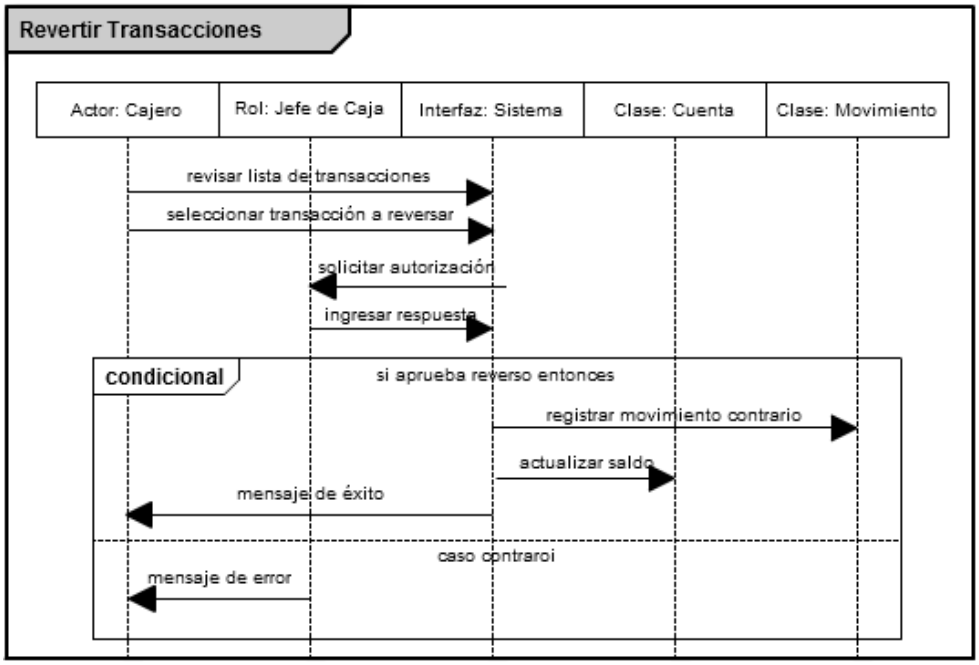

Figura 47: Diagrama de secuencia del caso de uso Revertir Transacciones Monetarias

3.2.6.27 Diagrama de secuencia del caso de uso Cuadrar Caja

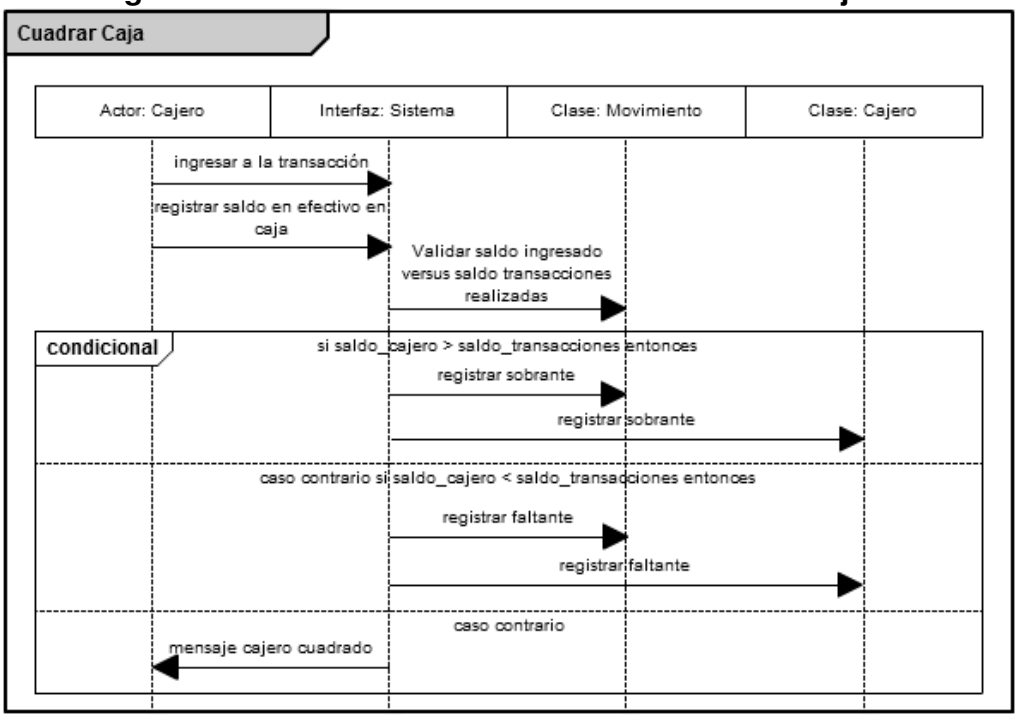

Figura 48: Diagrama de secuencia del caso de uso Cuadrar Caja

3.2.6.28 Diagrama de secuencia del caso de uso Entregar Efectivo a Bóveda

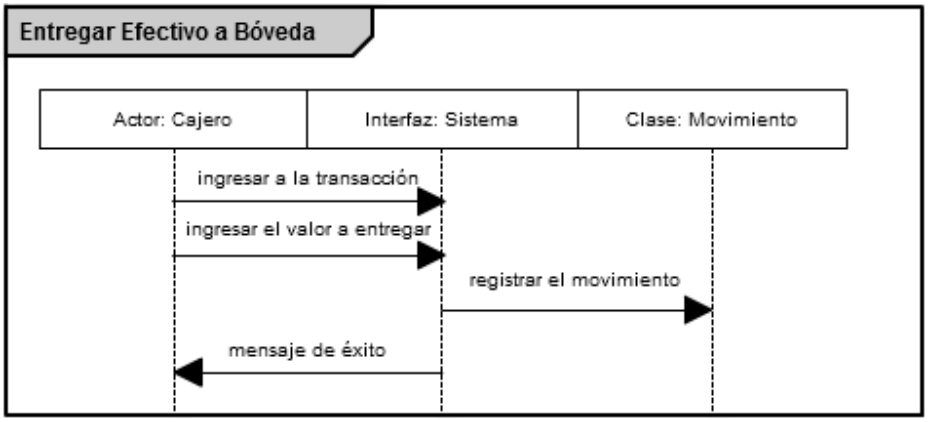

Figura 49: Diagrama de secuencia del caso de uso Entregar Efectivo a Bóveda

3.2.6.29 Diagrama de secuencia del caso de uso Cerrar Caja

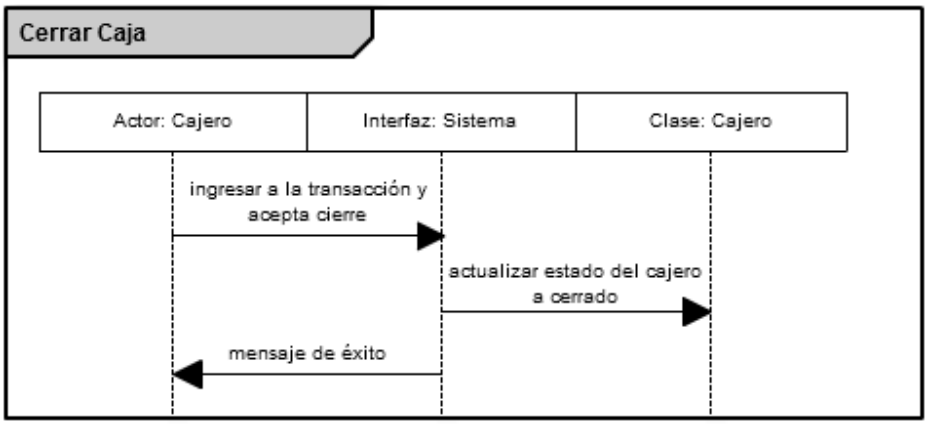

Figura 50: Diagrama de secuencia del caso de uso Cerrar Caja

Desde la Figura 51 a la Figura 53 se presentan los Diagramas de Secuencia para el perfil Jefe de Caja.

3.2.6.30 Diagrama de secuencia del caso de uso Abrir Bóveda

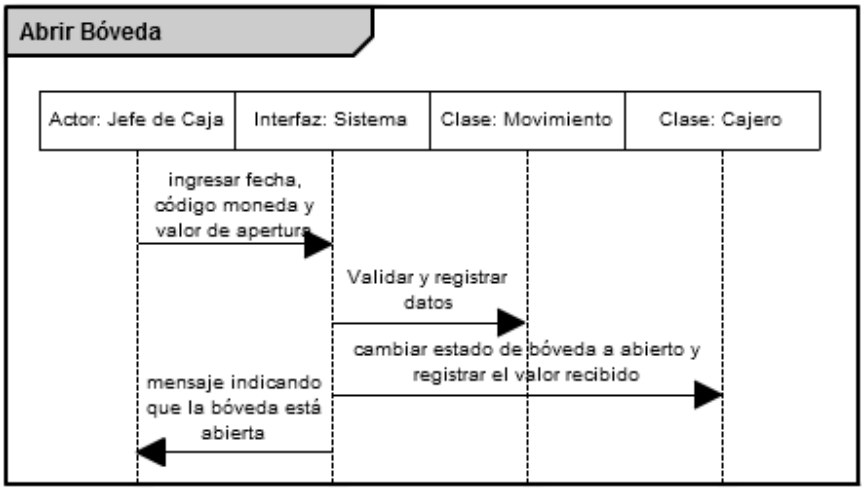

Figura 51: Diagrama de secuencia del caso de uso Abrir Bóveda

3.2.6.31 Diagrama de secuencia del caso de uso Recibir Dinero de Cajeros/Entregar de Dinero a Cajeros

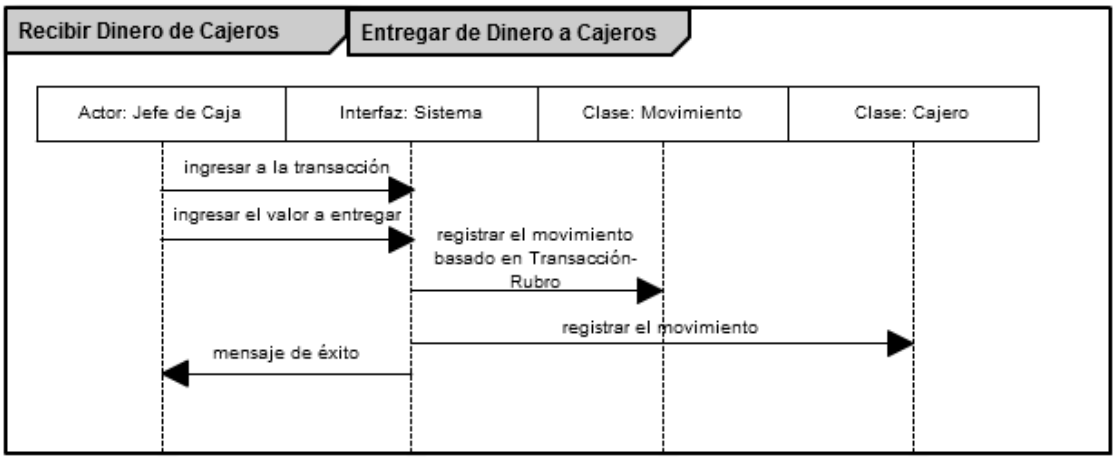

Figura 52: Diagrama de secuencia del caso de uso Recibir Dinero de Cajeros / Entrega de Dinero a Cajeros

3.2.6.32 Diagrama de secuencia del caso de uso Autorizar Transacciones

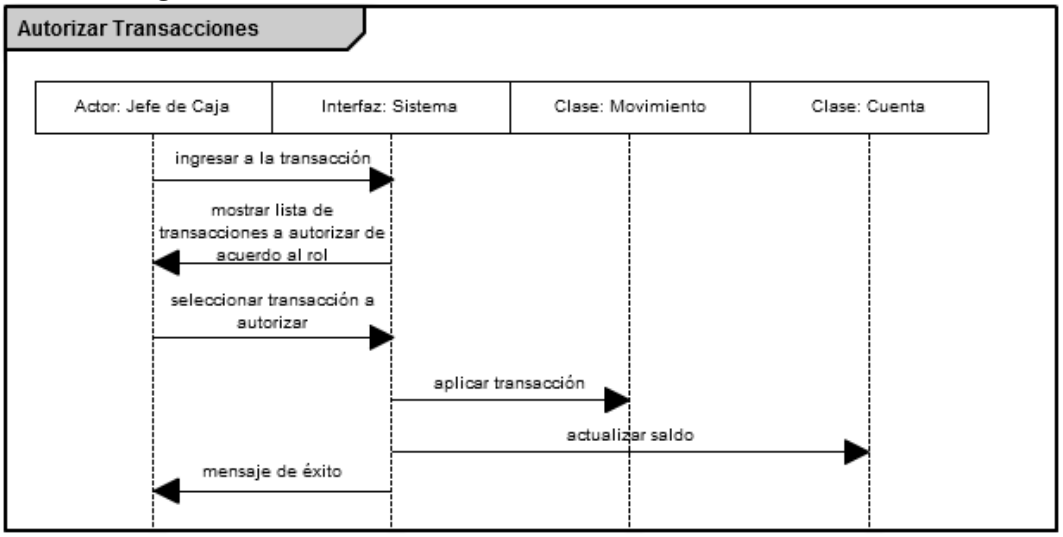

Figura 53: Diagrama de secuencia del caso de uso Autorizar Transacciones

En las Figuras 54, 55 y 56, se presenta el diagrama de secuencia de los casos de uso identificados en la Gestión de Socios.

3.2.6.33 Diagrama de secuencia del caso de uso Registrar Socios

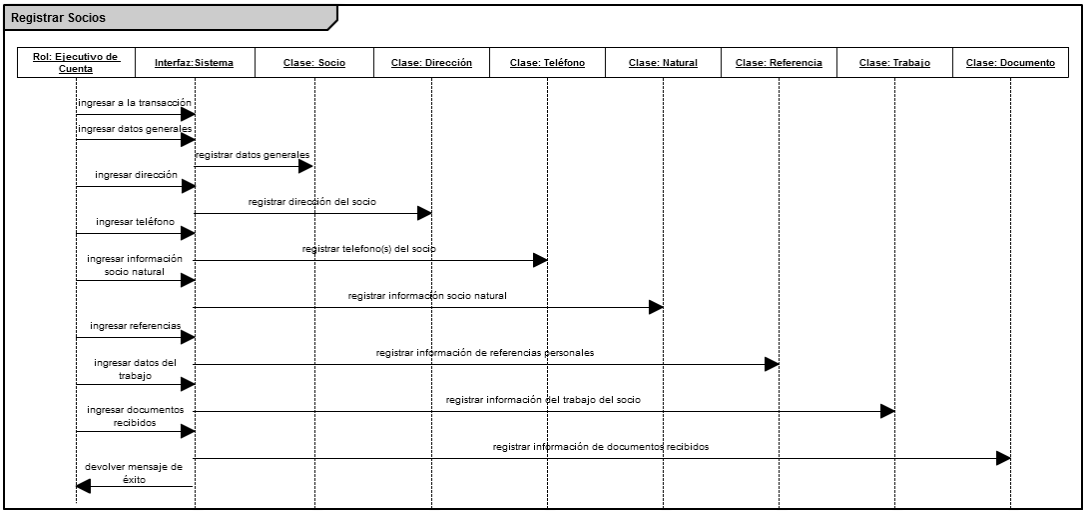

Figura 54: Diagrama de secuencia del caso de uso Registrar Socios

3.2.6.34 Diagrama de secuencia del caso de uso Consultar Socios

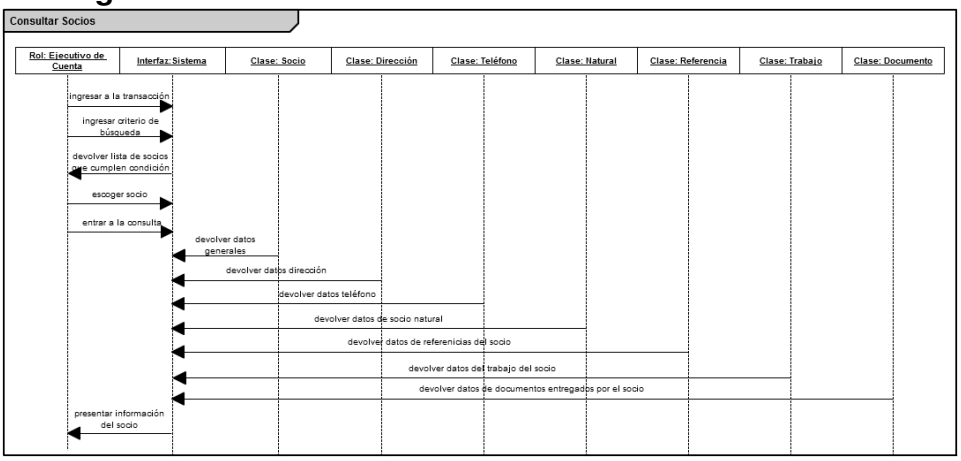

Figura 55: Diagrama de secuencia del caso de uso Consultar Socios

3.2.6.35 Diagrama de secuencia del caso de uso Modificar datos del Socio

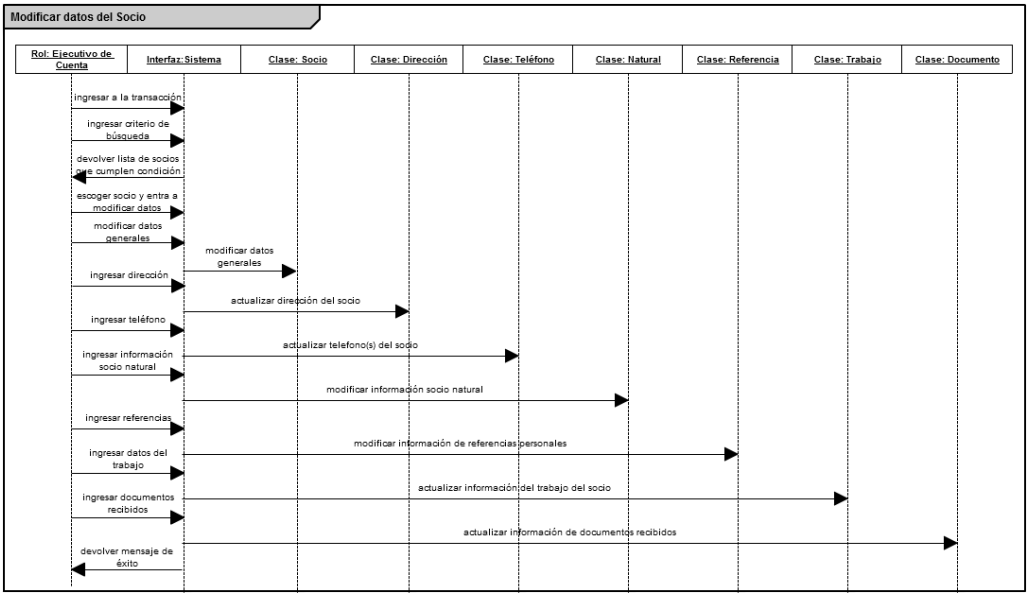

Figura 56: Diagrama de secuencia del caso de uso Modificar datos del Socio

En las Figuras que van desde la 57 hasta la 61, se presentan los diagramas de secuencia, de aquellos casos de uso identificados bajo la Gestión de Cuentas que realiza el Ejecutivo de Cuenta.

3.2.6.36 Diagrama de secuencia del caso de uso Abrir Cuentas

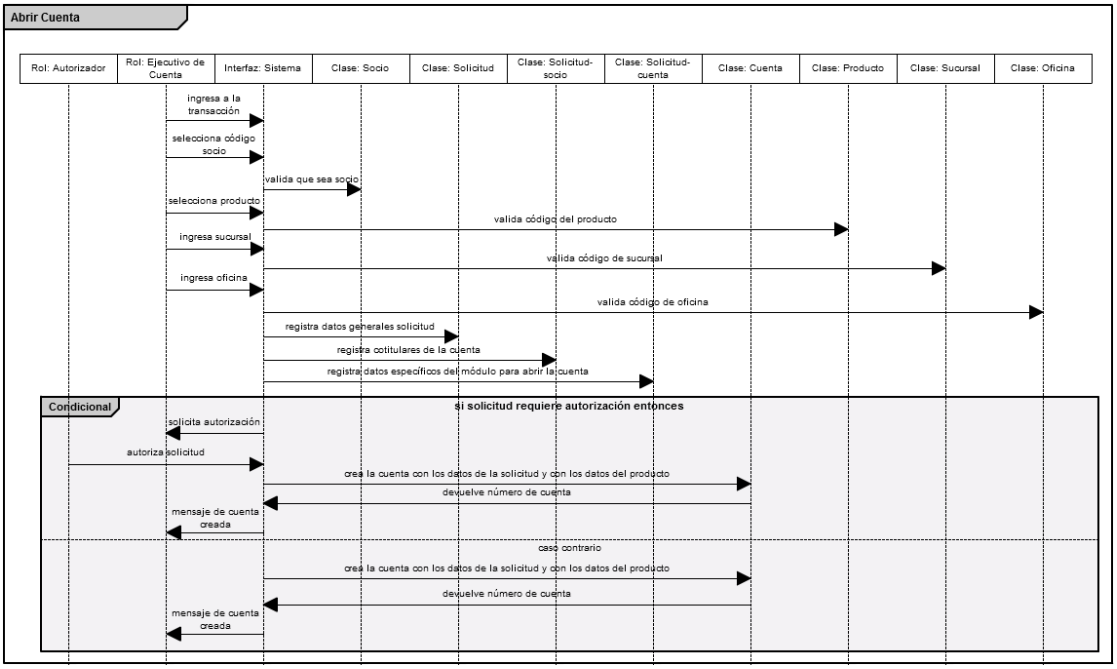

Figura 57: Diagrama de secuencia del caso de uso Abrir Cuenta

3.2.6.37 Diagrama de secuencia del caso de uso Consultar Saldos

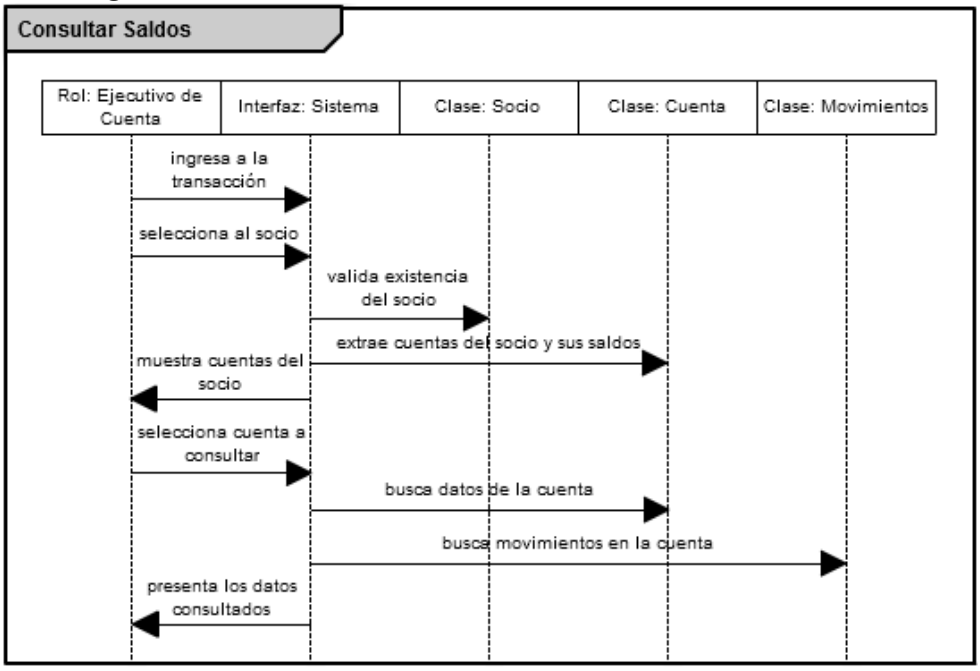

Figura 58: Diagrama de secuencia del caso de uso Consultar Saldo

3.2.6.38 Diagrama de secuencia del caso de uso Transferir Dinero

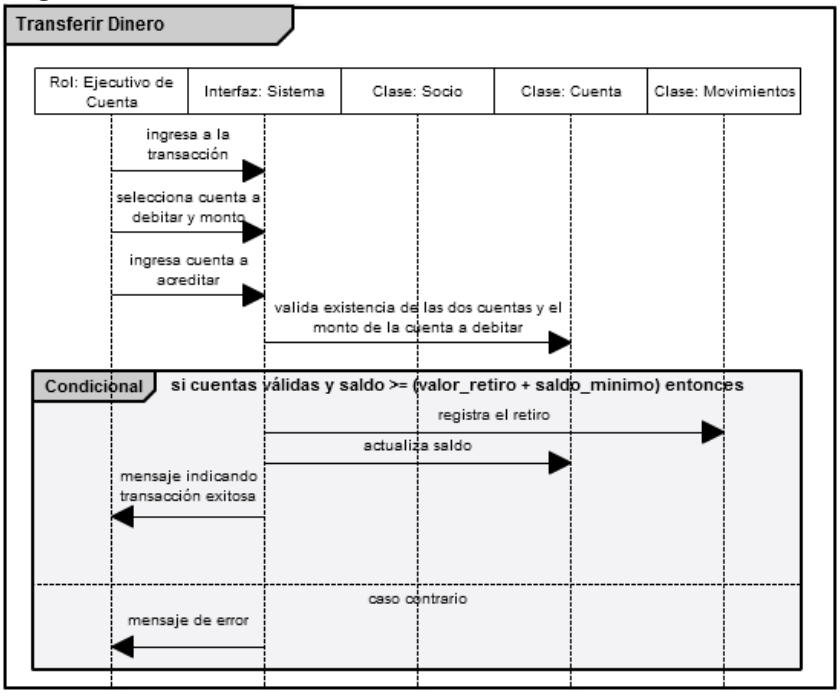

Figura 59: Diagrama de secuencia del caso de uso Transferir Dinero

3.2.6.39 Diagrama de secuencia del caso de uso Imprimir Estado de Cuenta

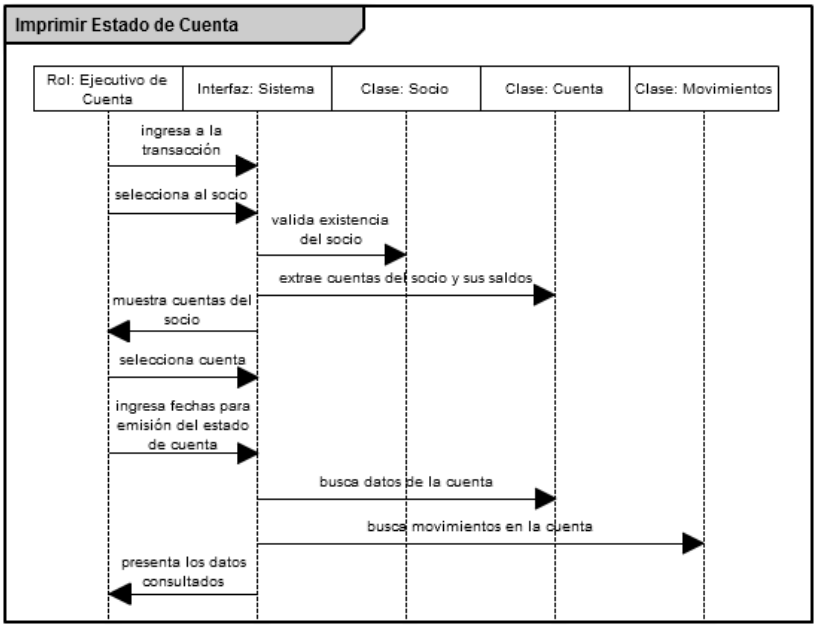

Figura 60: Diagrama de secuencia del caso de uso Imprimir Estado de Cuenta

3.2.6.40 Diagrama de secuencia del caso de uso Cerrar Cuenta

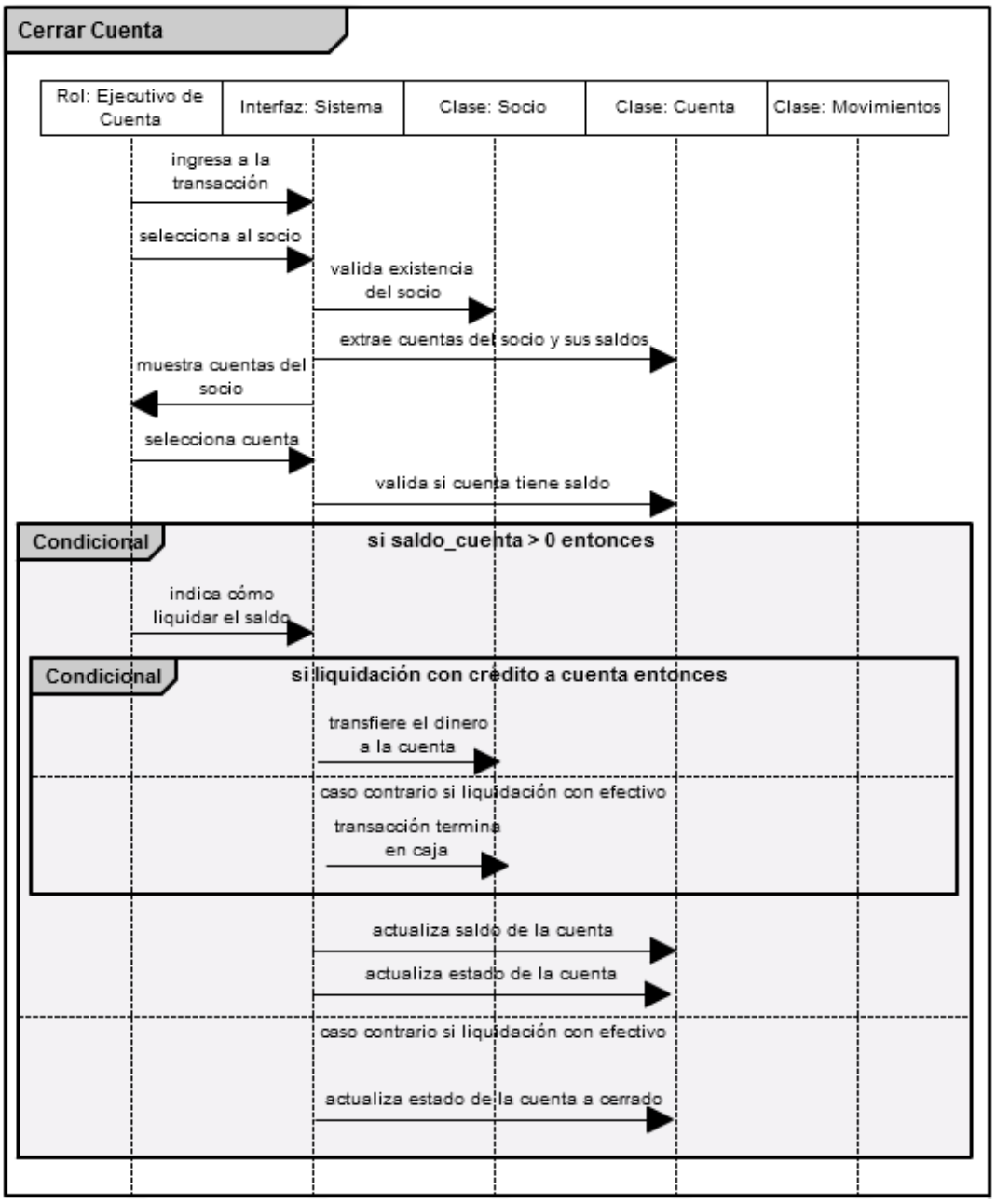

Figura 61: Diagrama de secuencia del caso de uso Cerrar Cuenta

A continuación, en la Figura 62, se presenta el diagrama de secuencia, que representa los casos de uso identificados dentro del Cierre de Operaciones

Diario, que es un proceso automático que se debe ejecutar todos los días por parte del operador del sistema.

3.2.6.41 Diagrama de secuencia del caso de uso Procesar Cierre de Operaciones Diario

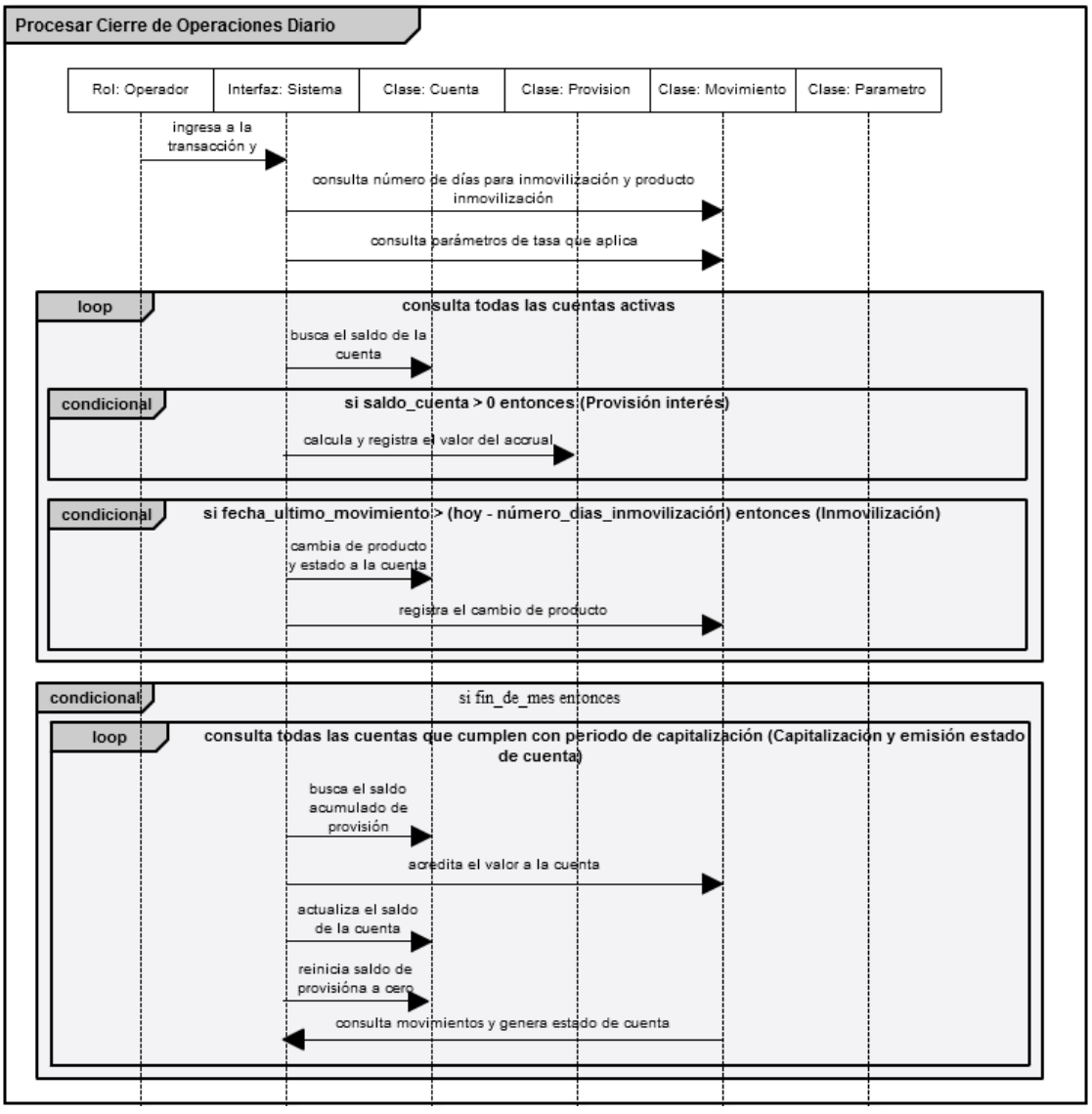

Figura 62: Diagrama de secuencia del caso de uso Procesar Cierre de Operaciones Diario

3.2.7 Diagramas de Estado

Los Diagramas de Estado son muy útiles para mostrar el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación en respuesta a eventos y al tiempo, junto con sus respuestas y acciones, ilustrando de esta manera que eventos pueden cambiar el estado de los objetos de la clase.

A continuación se va a presentar el diagrama de estado para aquellos objetos con un comportamiento significativo, pues para el resto de objetos se consideraría que estos tienen un estado único.

3.2.7.1 Diagramas de Estado de una Cuenta a la Vista

Durante la participación de un socio en la cooperativa, se ha identificado un cierto número de estados definidos partiendo de la idea de que una vez que se registra al socio, este no debe ser eliminado físicamente, solamente sufre un cambio de estado si el socio ha fallecido o si este ha sido expulsado de acuerdo a las normas que rigen a la cooperativa.

La figura 63 se muestra el diagrama de estado de un socio durante su participación con la cooperativa:

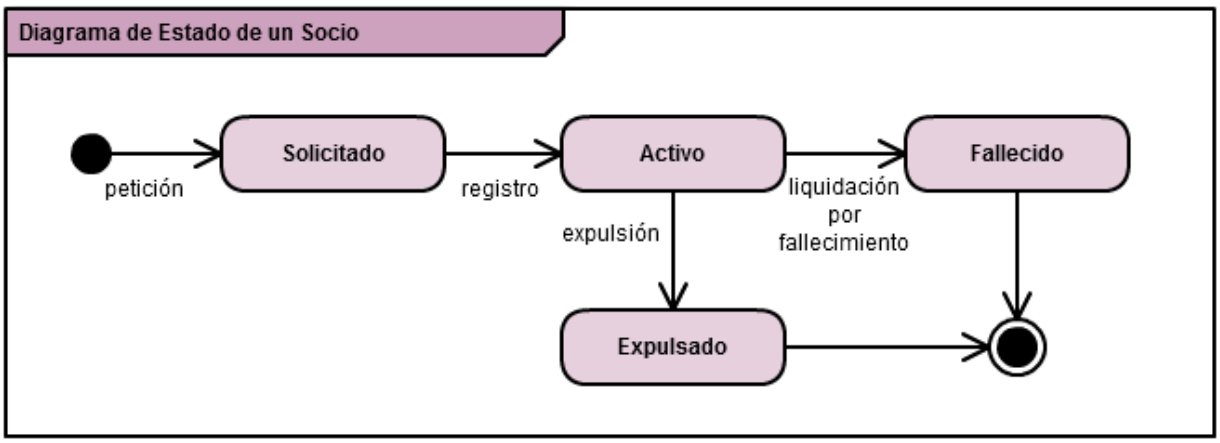

Figura 63: Diagrama de estado de un Socio

3.2.7.2 Diagramas de Estado de una Cuenta a la Vista

En el proceso de creación de una cuenta a la vista, se inicia con una solicitud de creación de una cuenta, la cual puede ser aprobada o negada. Si el producto al que pertenece la cuenta exige que la cuenta se cree con un valor mínimo, entonces, una vez que se reciba el dinero, ya sea por caja o mediante una transferencia, el estado de la cuenta pasará a activa, lo cual significa que la cuenta puede operar con normalidad.

Una cuenta activa puede ser inactivada si ha transcurrido un tiempo definido mediante parámetro sin haber sido movida por su dueño. La cuenta también podría ser bloqueada por diferentes razones como por la pérdida de documentos. Y finalmente la cuenta puede ser cerrada a petición del dueño, por expulsión del socio o por fallecimiento.

En la figura 64, se muestra como el estado de la cuenta puede ir cambiando en el tiempo de acuerdo a los eventos ahí indicados.

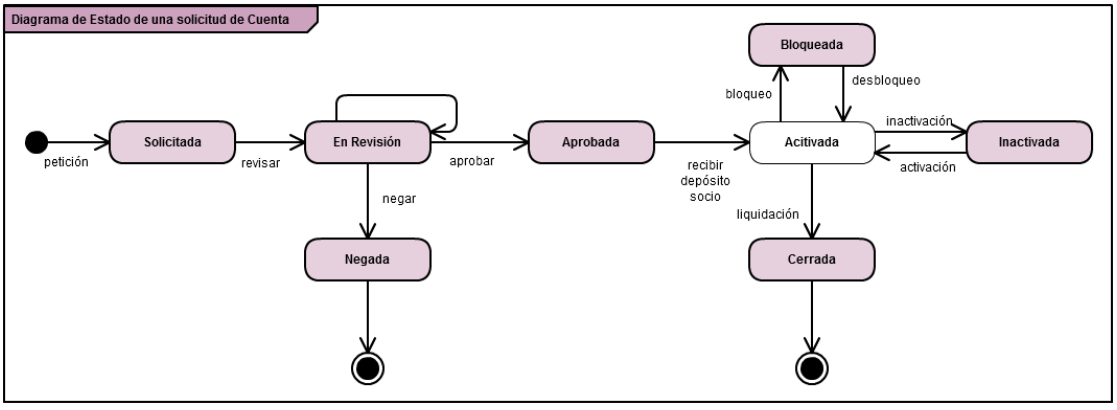

Figura 64: Diagrama de estado de una cuenta

CAPÍTULO 4

4. IMPLEMENTACIÓN DEL SISTEMA

4.1 Instalación del Sistema

Para la implementación del sistema informático financiero cooperativista se utiliza las siguientes herramientas:

- Como motor de base de datos se trabajó con Oracle 9i.
- Para el desarrollo de la aplicación tanto de pantallas y reportes se trabaja con Oracle developer 6i.
- Para la programación de la lógica del negocio, se realiza la mayor parte del código en procedimientos y funciones, registradas como procedimientos almacenados (Store Procedure SP) de la base de datos a través del uso de PL SQL.
- Para almacenar las formas, gráficos y reportes, se utiliza Developer/2000 Server.

A continuación, en la Figura 65 se muestra un esquema de la instalación.

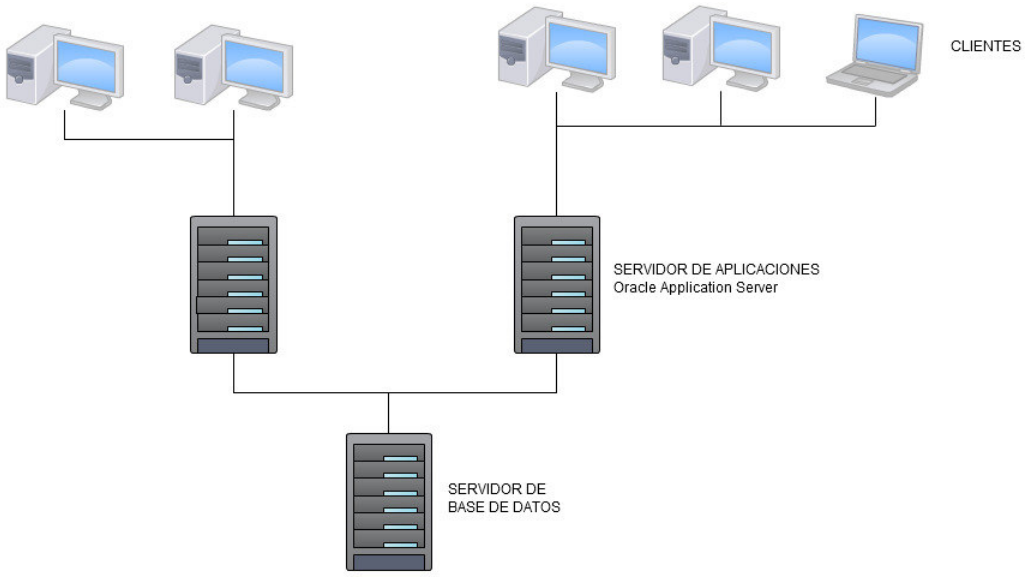

Figura 65: Esquema de la Instalación

4.2 Modelo de Despliegue

El diagrama de despliegue ayuda a modelar el hardware utilizado, dando una visión general de la instalación. A continuación, en la Figura 66 se presenta el Diagrama de Despliegue empleado para la instalación y explotación del sistema.

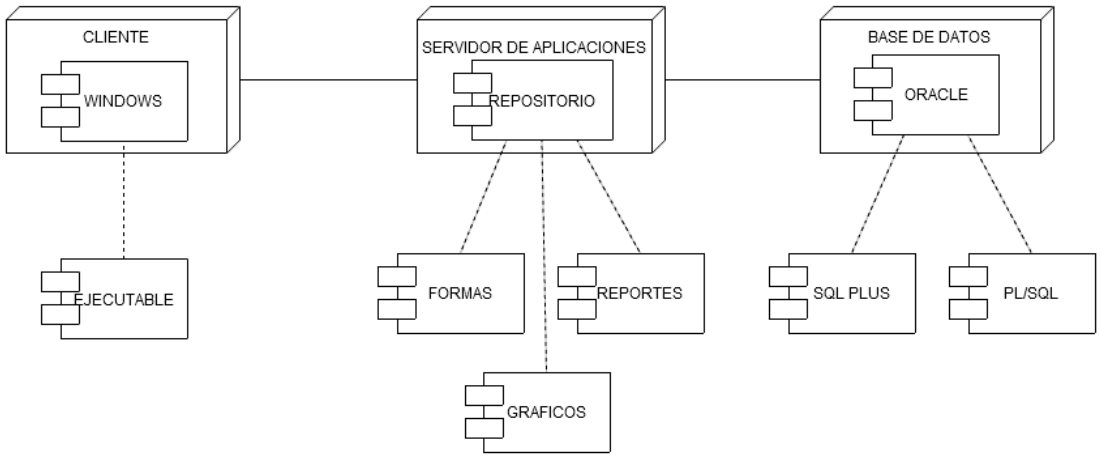

Figura 66: Diagrama de Despliegue

4.3 Modelo de Implementación

El diagrama de implementación, forma parte de los diagramas utilizados para documentar sistemas, y es la ilustración de la arquitectura física del hardware y del software, el cual permite entender la arquitectura que se maneja en el proyecto. En la figura 67, se presenta el modelo que se emplea en la implementación del sistema informático financiero cooperativista.

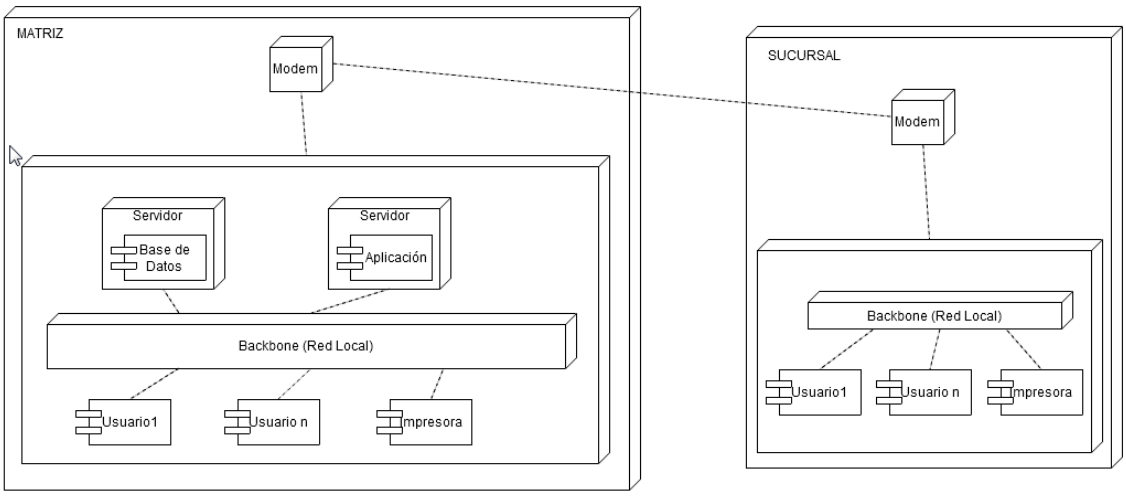

Figura 67: Modelo de Implementación

4.4 Diagrama de Navegación

A continuación, de la figuras 68 a la 72 se presenta los diagramas de navegación del sistema, basado en el rol del usuario. Para la realización de estos diagramas no existe una notación específica, por lo que se utiliza una notación que se considera adecuada, para representar la navegación en el sistema.

4.4.1 Diagrama de Navegación del Administrador del sistema.

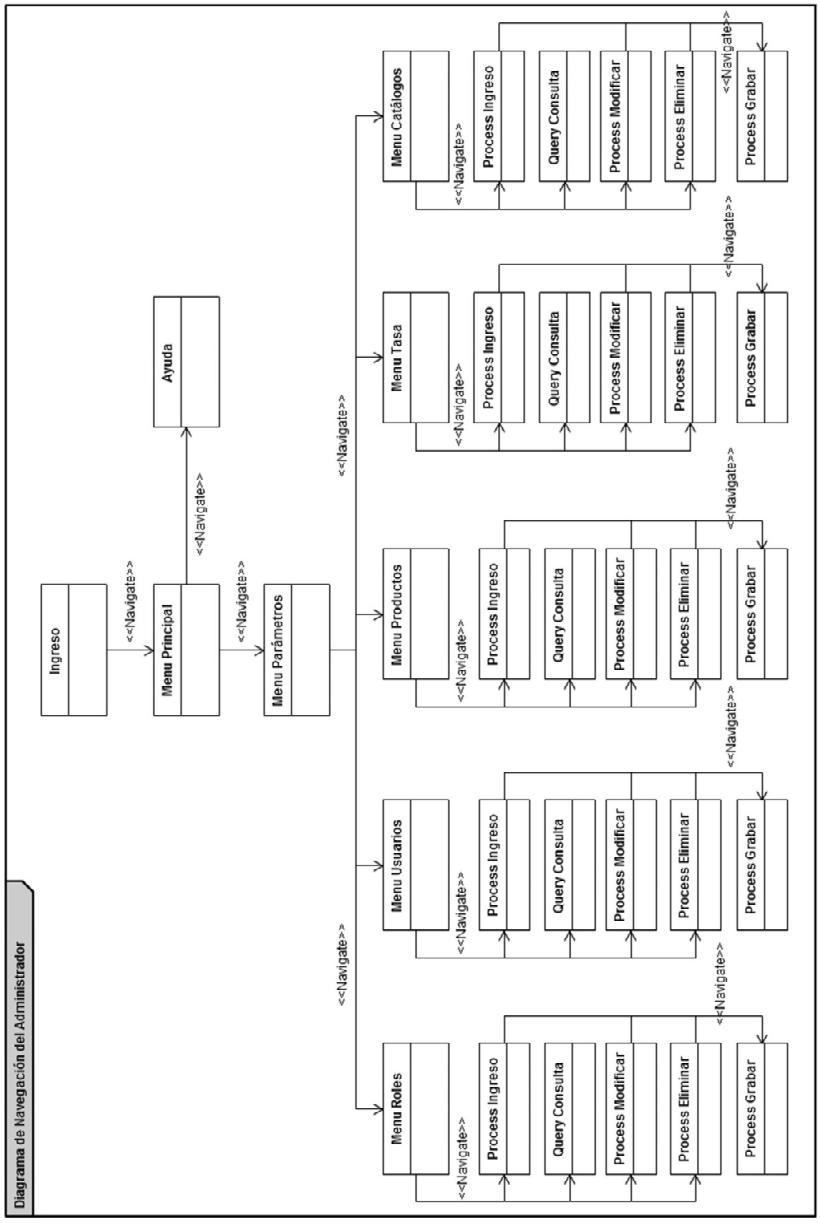

Figura 68: Diagrama de navegación del Administrador del sistema

4.4.2 Diagrama de Navegación del Cajero.

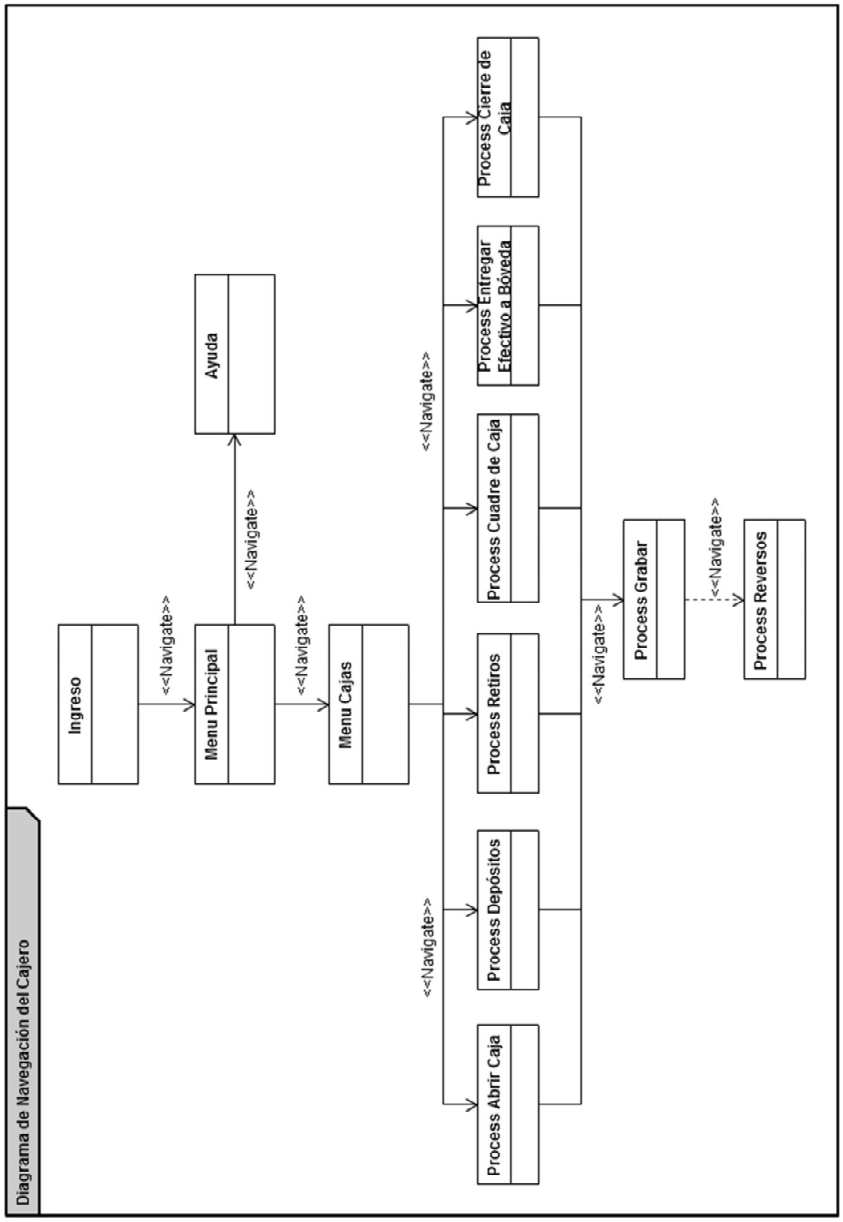

Figura 69: Diagrama de Navegación del Cajero

4.4.3 Diagrama de Navegación del Jefe de Caja.

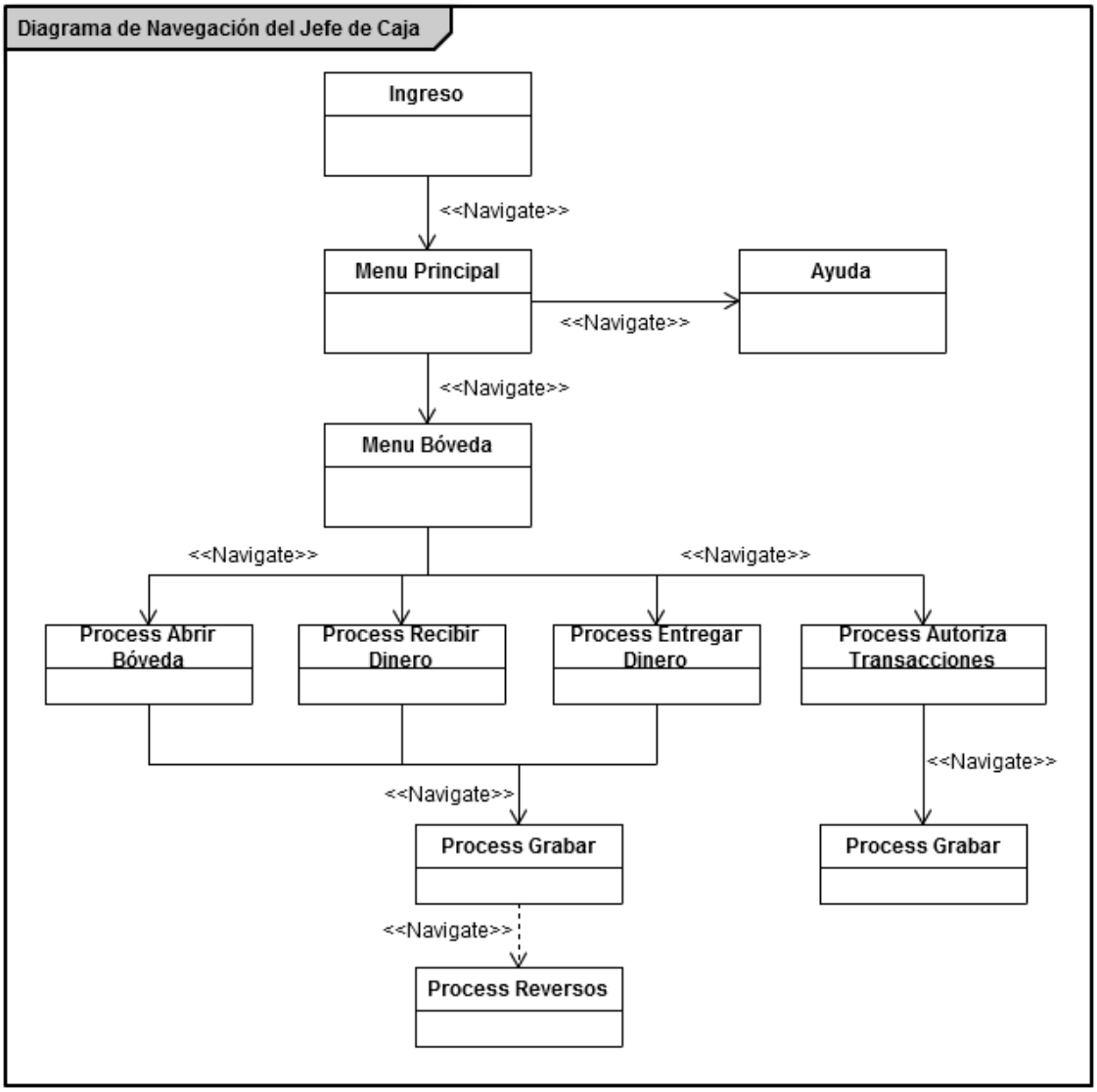

Figura 70: Diagrama de navegación del Jefe de Caja

4.4.4 Diagrama de Navegación del Ejecutivo de Cuenta.

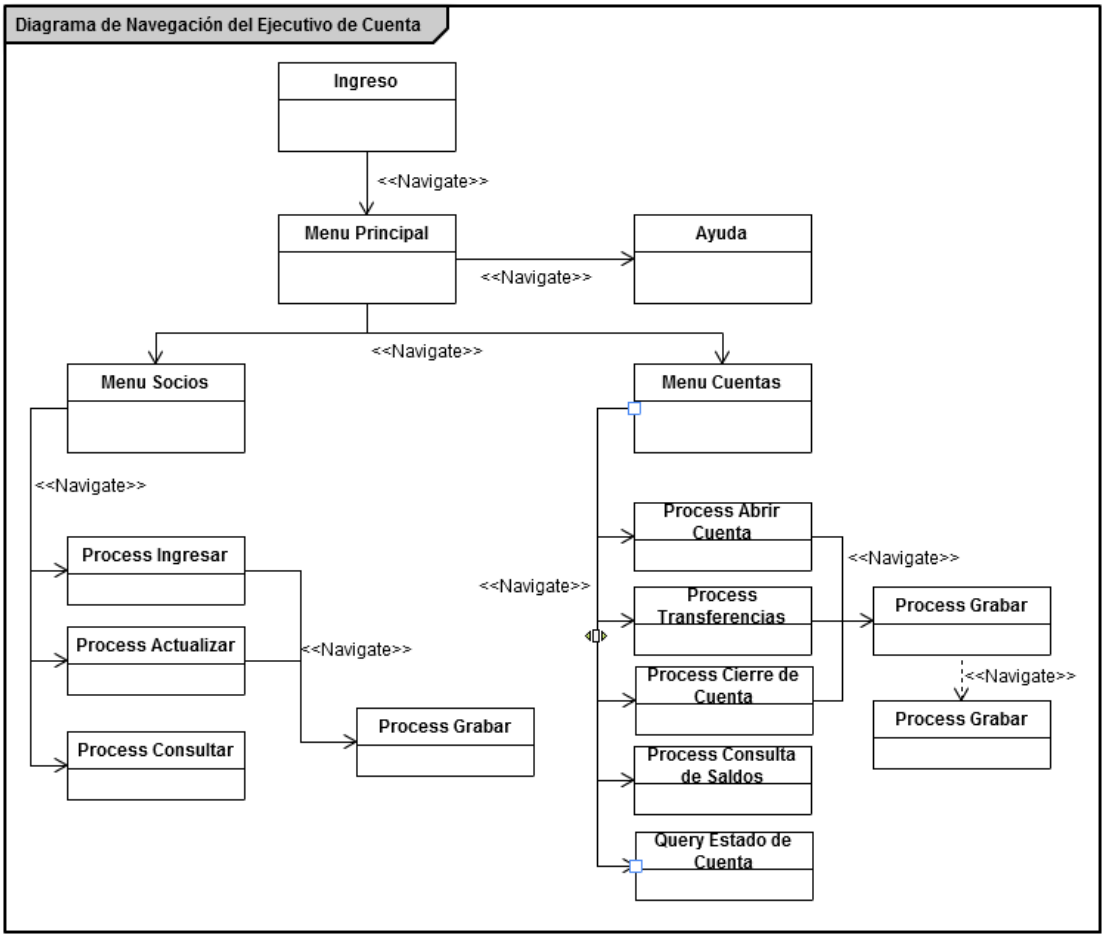

Figura 71: Diagrama de navegación del Ejecutivo de Cuenta

4.4.5 Diagrama de Navegación del Operador.

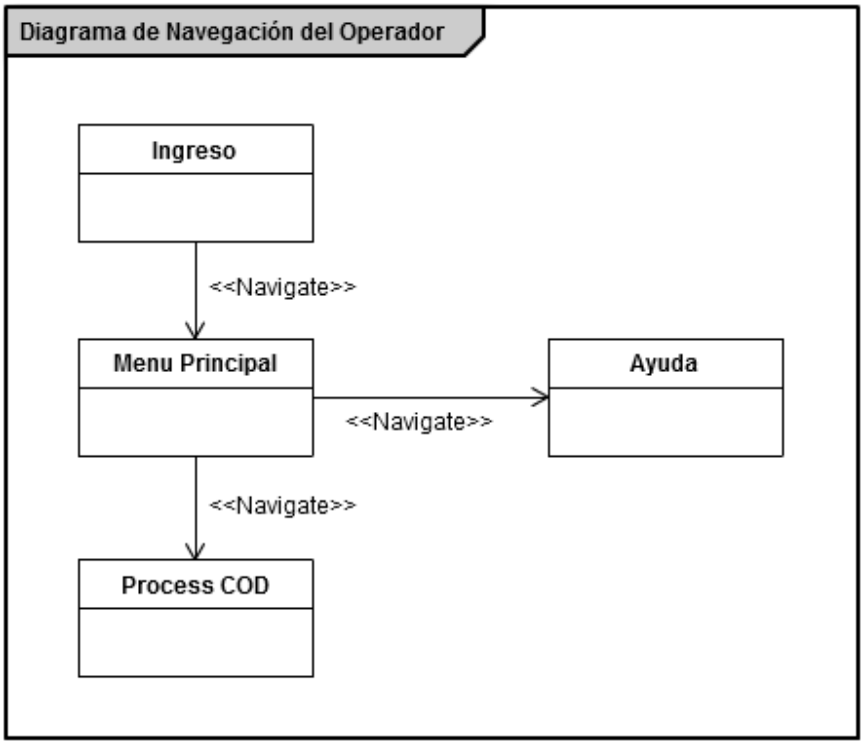

Figura 72: Diagrama de navegación del Operador del sistema

4.5 Diseño de la Interfaz

En la Figura 73, se presenta la forma en la que la interfaz de usuario está organizada, donde aprovechando las bondades de Oracle Developer Forms, se permite crear un ambiente organizado por objetos como son: una ventana principal que abarca en la parte superior, una barra que permite mostrar el nombre de la cooperativa, un barra para mostrar el menú, una barra de herramientas, y una barra con información relativa a la sesión del usuario que está conectado al sistema. En la parte inferior, muestra una barra en la que el sistema muestra mensajes informativos al usuario, mensajes de navegación y mensajes de error. La ventana principal que contiene los objetos indicados, constituye un lienzo, sobre el cual se permite el despliegue de ventanas internas, que contienen la información de la base de datos, que puede estar en forma de

bloques individuales, es decir como formularios, o bloques que contienen varios registros de una tabla. Finalmente cada registro contiene campos con la información a mostrar.

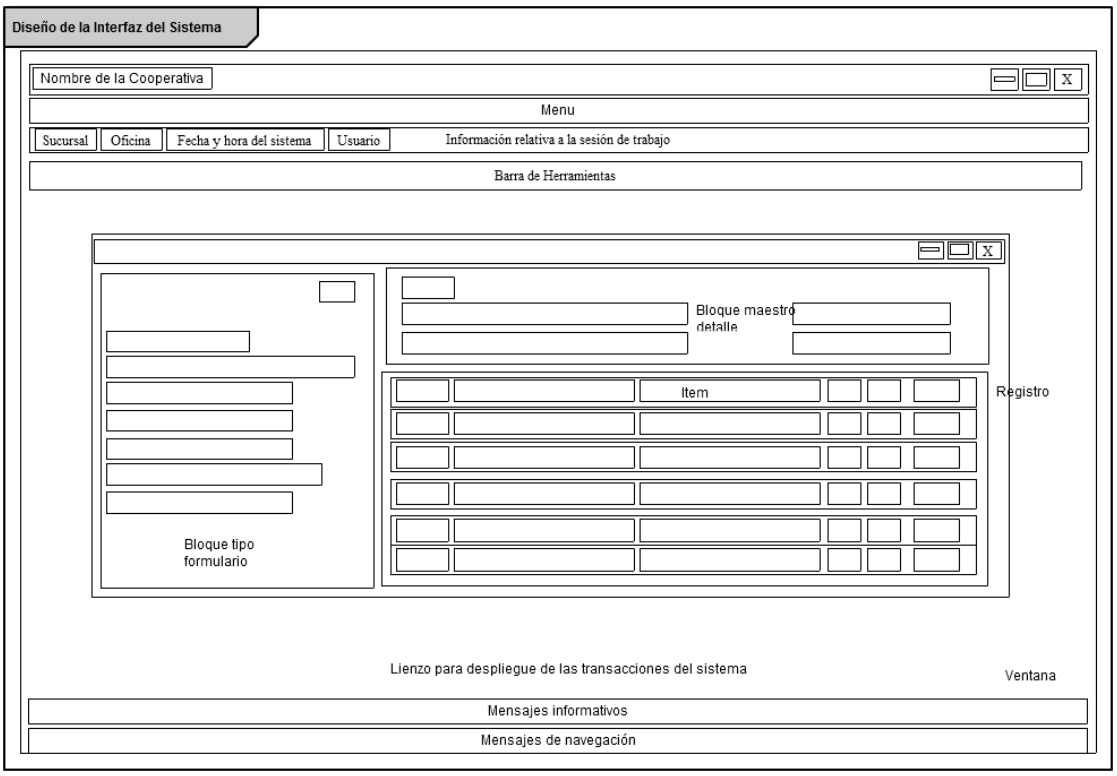

Figura 73: Diseño de la Interfaz

En la Figura 74, se muestra la pantalla principal del sistema, que al igual que un lienzo para un pintor, en esta ventana se muestran nuevas ventanas correspondientes a cada transacción, esto es de acuerdo a la funcionalidad propia que tiene Forms Developer. Cada ventana que se abre sobre este lienzo, tiene las mismas características de la ventana principal, de modo que trabaja como un estándar utilizado en el sistema, tanto en colores como en tipo de letra.

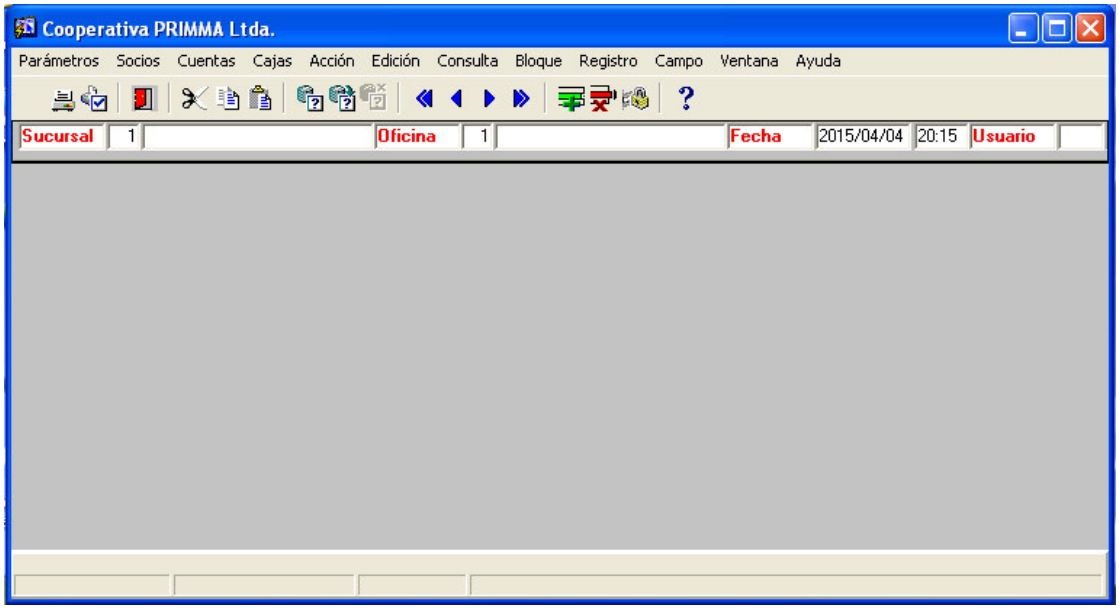

Figura 74: Pantalla principal

4.6 Pruebas del sistema

En la ejecución de pruebas del sistema, se trabaja con pruebas de caja blanca y pruebas de caja negra.

4.6.1. Pruebas de caja blanca

En vista de la magnitud del sistema, se presenta la prueba de caja blanca sobre la unidad que permite el ingreso al sistema.

Para esto, en la Figura 75, se presenta el código de la unidad a probar y en la Figura 76 se muestra el Diagrama de Flujo de la estructura de control de programación de la unidad indicada, la cual nos permite a su vez definir los caminos para la realización de esta prueba.

```
CREATE OR REPLACE PROCEDURE validar_ingreso(pusuario IN NUMBER, pclave IN VARCHAR2, perror OUT VARCHAR2) IS
  vEstado NUMBER(6);
  vHorario NUMBER(1);
  TYPE defUsuarioxRol IS RECORD(
  usuario          usuarioxrol.usuario%TYPE,
  codigorol        usuarioxrol.codigorol%TYPE,
  codigoe          usuarioxrol.codigoe%TYPE,
  clave            usuarioxrol.clave%TYPE,
  fechainicio      usuarioxrol.fechainicio%TYPE,
  fechafin         usuarioxrol.fechafin%TYPE,
  estado           usuarioxrol.estado%TYPE,
  intentospermit   usuarioxrol.Intentospermit%TYPE,
  intentosfallidos usuarioxrol.Intentosfallidos%TYPE);
  rUsuarioxRol     defUsuarioxRol;
BEGIN
  BEGIN
    SELECT usuario, codigorol, codigoe, clave, fechainicio,
           fechafin, estado, intentospermit, intentosfallidos
    INTO   rUsuarioxRol.usuario,
           rUsuarioxRol.codigorol,
           rUsuarioxRol.codigoe,
           rUsuarioxRol.clave,
           rUsuarioxRol.fechainicio,
           rUsuarioxRol.fechafin,
           rUsuarioxRol.estado,
           rUsuarioxRol.intentospermit,
           rUsuarioxRol.intentosfallidos
    FROM   usuarioxrol
    WHERE  usuario = pusuario;
  EXCEPTION
  WHEN NO_DATA_FOUND THEN
      perror := 'Código de usuario no registrado en el sistema.';
      RETURN;
  END;
    IF rUsuarioxRol.estado = 1 THEN -- ESTA ACTIVO
      IF pclave <> rUsuarioxRol.clave THEN --Clave erronea
        vEstado := rUsuarioxRol.estado;
        IF (rUsuarioxRol.intentosfallidos + 1) >= rUsuarioxRol.intentospermit THEN
          vEstado := 2; --Bloqueado
        END IF;

        UPDATE usuarioxrol
        SET    estado = vEstado,
               intentosfallidos = NVL(intentosfallidos,0) + 1
        WHERE  usuario = pusuario;

        perror := 'La clave ingresada es incorrecta. Intente nuevamente.';
      ELSE --Clave correcta
        vHorario := valida_horario(pusuario);
        IF vHorario = 1 THEN --Dentro del horario permitido
          NULL; --Usuario ingresa al sistema luego de pasar todos los filtros
        ELSE -- Fuera del horario permitido
          perror := 'El usuario está intentando ingresar fuera del horario permitido.';
        END IF;
      END IF;
    ELSE -- NO ES ACTIVO
      perror := 'El usuario no está ACTIVO.';
    END IF;
END validar_ingreso;
```

Figura 75: Código sobre el cual se ejecuta la prueba de caja blanca.

Se encuentra los siguientes caminos:

Camino 1: 1-2-3-4-5-6-20

Camino 2: 1-2-3-4-5-7-8-9-20

Camino 3: 1-2-3-4-5-7-8-10-11-12-13-14-15-20

Camino 4: 1-2-3-4-5-7-8-10-11-12-13-15-20

Camino 5: 1-2-3-4-5-7-8-10-11-16-17-18-20

Camino 6: 1-2-3-4-5-7-8-10-11-16-17-19-20

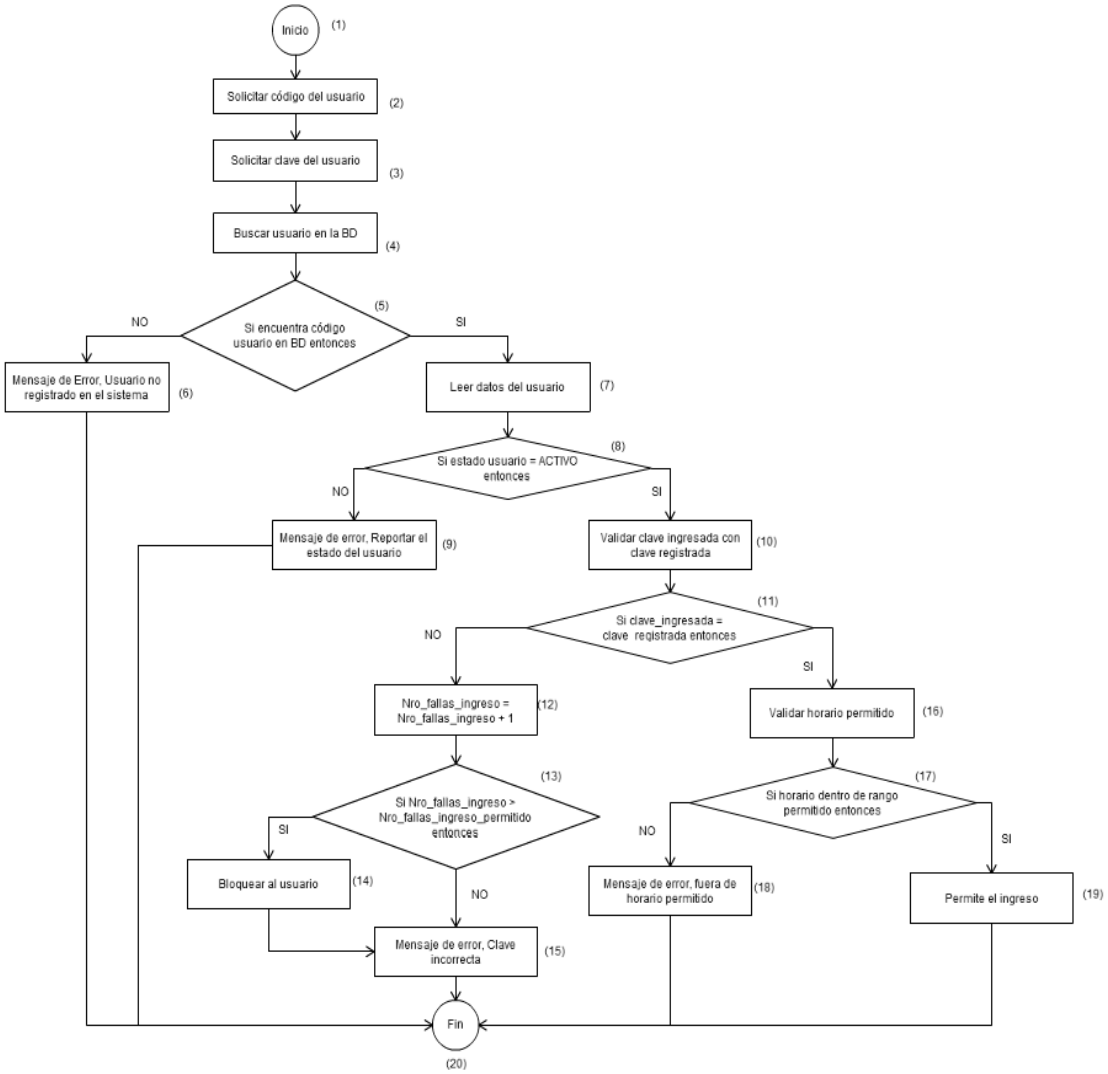

Figura 76: Diagrama de Flujo de la Estructura de Control de Programación

Para realizar la prueba de caja blanca de los caminos encontrados, se ejecuta el procedimiento almacenado, responsable de la validación del ingreso al sistema, con los parámetros adecuados para lograr que se ejecuten los mensajes de error. Para ejecutar el procedimiento almacenado, se utiliza SQL PLUS.

En el grupo de tablas que se presenta a continuación, que van desde la tabla 34 a la 39, se describe el objetivo de cada una de las pruebas a realizar, y sus respectivos resultados están en las figuras que van desde la 77 hasta la 83.

Tabla 34

Prueba camino 1

OBJETIVO		Ingresar un código de usuario que no existe en la base de datos y validar si el procedimiento, rechaza al usuario que intenta conectarse.
Datos	**Usuario**	9578
	Clave	XXXXXX
RESULTADO		El procedimiento almacenado recorre el camino esperado, y despliega a través de la interfaz un mensaje de error indicando que el usuario no está registrado.

En la Figura 77, se muestra el resultado de la prueba del camino 1, definido en la tabla 34, donde consta que el sistema realiza la validación correctamente.

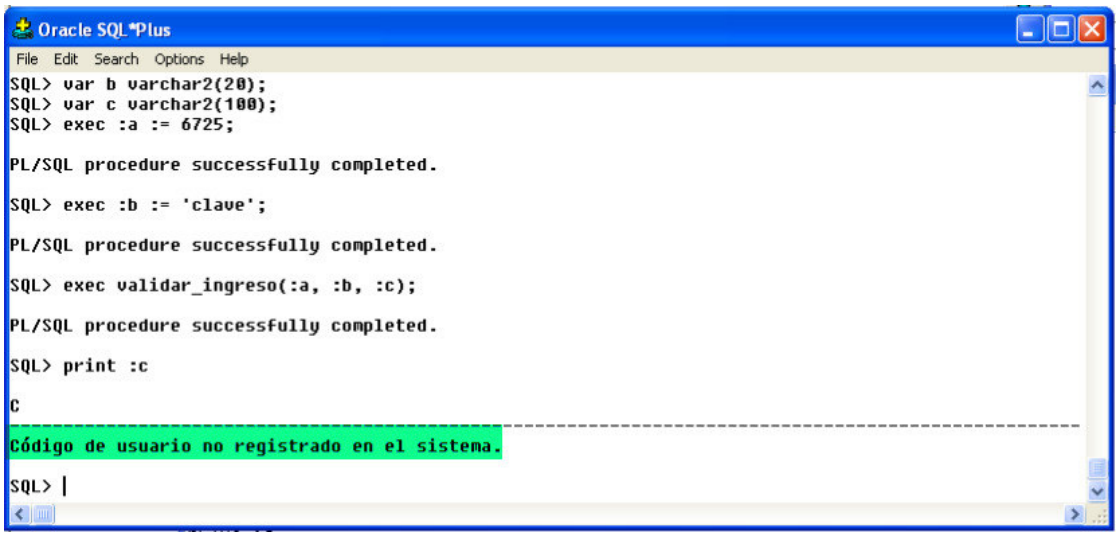

Figura 77: Resultado prueba camino 1.

Tabla 35

Prueba camino 2

OBJETIVO		Ingresar un código de usuario registrado en el sistema y que este tenga un estado diferente de ACTIVO, para comprobar si el sistema rechaza la conexión.
Datos	**Usuario**	1
	Clave	XXXXXX
RESULTADO		El procedimiento almacenado recorre el camino esperado, y muestra a través de la interfaz un mensaje de error indicando que el usuario no está activo, por tanto no puede conectarse al sistema.

En la Figura 78, se muestra el resultado de realizar la prueba del camino 2, y se aprecia que el sistema realiza la validación de una manera correcta.

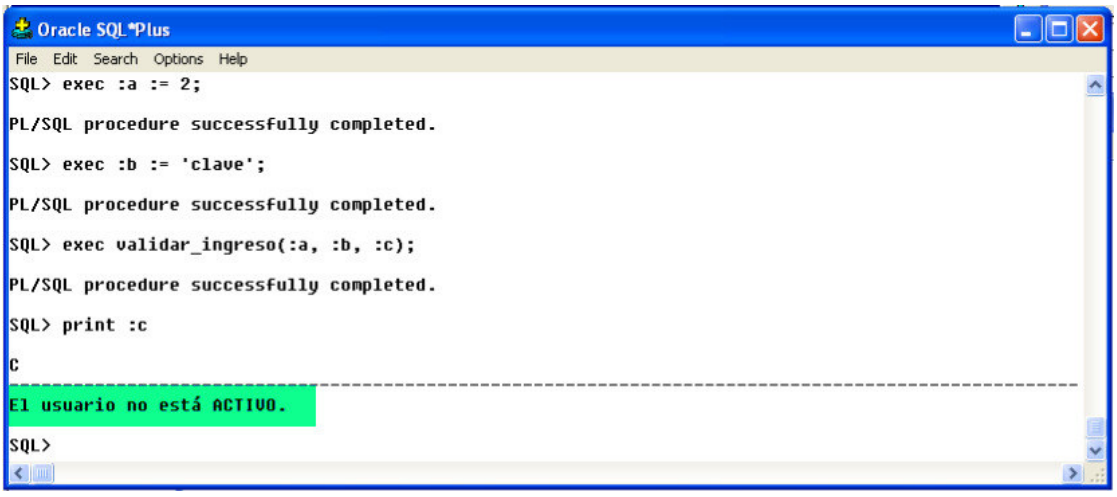

Figura 78: Resultado prueba camino 2.

Tabla 36

Prueba Camino 3

OBJETIVO		Ingresar un código de usuario registrado en la base de datos, con estado ACTIVO y con un clave diferente a la registrada en el sistema, para validar si el procedimiento, rechaza la conexión del usuario que intenta conectarse.
Datos	**Usuario**	1
	Clave	ZZZZZ
RESULTADO		El procedimiento almacenado recorre el camino esperado, y muestra a través de la interfaz un mensaje de error indicando que la clave ingresada no es correcta.

En la Figura 79 se muestra el resultado de aplicar la prueba del camino 3, obteniendo un resultado correcto.

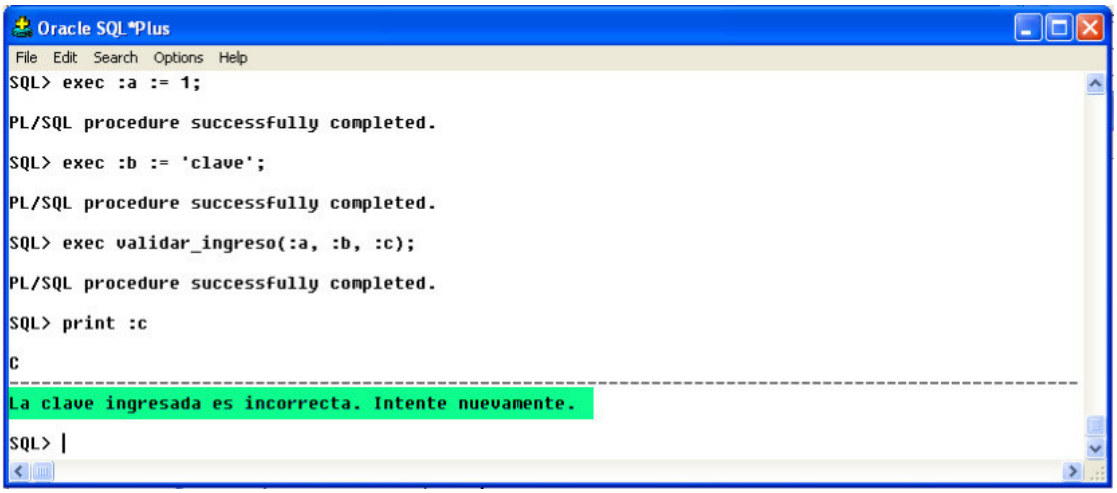

Figura 79: Resultado prueba camino 3.

Tabla 37

Prueba Camino 4

OBJETIVO		Ingresar un código de usuario registrado en el sistema, con estado ACTIVO, con un valor de clave diferente a la registrada en el sistema, y que además el número de intentos de conexión permitidos sea igual a 3 y que ya tenga dos intentos de conexión fallidos. Con esta prueba se debe validar que el usuario no pueda ingresar al sistema y que además se quede bloqueado por alcanzar el número de intentos fallidos.
Datos	**Usuario**	2
	Clave	MMMMMM
RESULTADO		El procedimiento almacenado recorre el camino esperado, por lo tanto se realiza el bloqueo del usuario y muestra a través de la interfaz un mensaje de error indicando que la clave ingresada no es correcta.

En la Figura 80, se muestra la prueba del camino 4, y en la Figura 81 se muestra el contenido del registro, luego de grabar la actualización del estado del usuario a 2 que significa bloqueado y de incrementar el número de intentos fallidos.

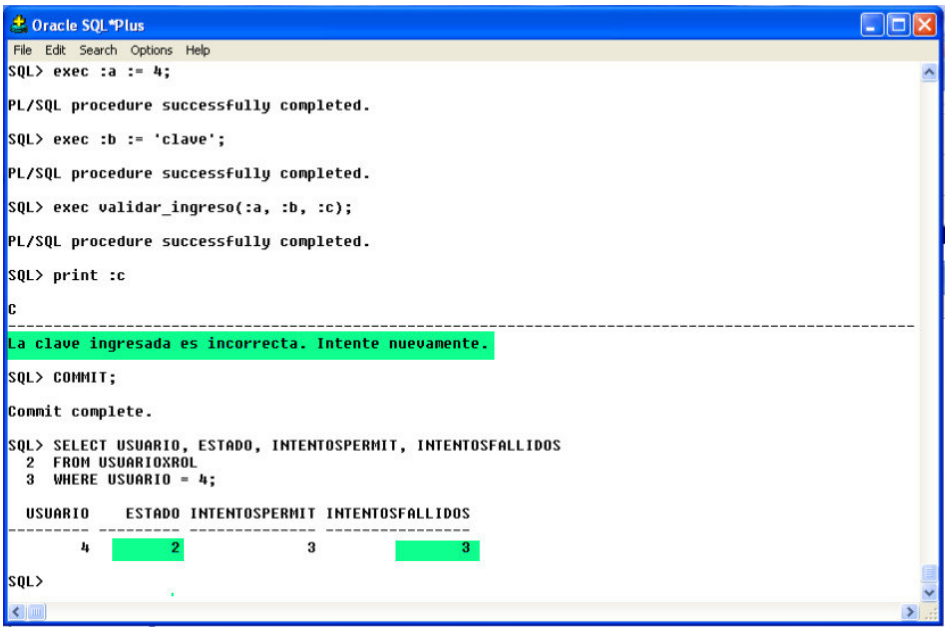

Figura 80: Resultado prueba camino 4.

	USUARIO	CODIGOROL	CODIGOE	CLAVE	FECHAINICIO	FECHAFIN	ESTADO	INTENTOSPERMIT	INTENTOSFALLIDOS
▶ 1	4	1	935	pSV6x1 ···	15/08/2013		2	3	3

Figura 81: Resultado prueba camino 4 - bloqueo usuario.

Tabla 38

Prueba Camino 5

OBJETIVO		Ingresar un código de usuario que está registrado en el sistema, con estado ACTIVO, con la clave registrada en el sistema, pero en un horario fuera del permitido, de modo que se valide si el sistema evita el ingreso de este usuario por estar fuera del horario permitido de trabajo.
Datos	**Usuario**	1
	Clave	XXXXXX
RESULTADO		El procedimiento almacenado recorre el camino esperado, y despliega a través de la interfaz un mensaje de error indicando que el usuario no puede conectarse por estar fuera de su horario permitido.

En la Figura 82, se muestra el resultado de la prueba del camino 5.

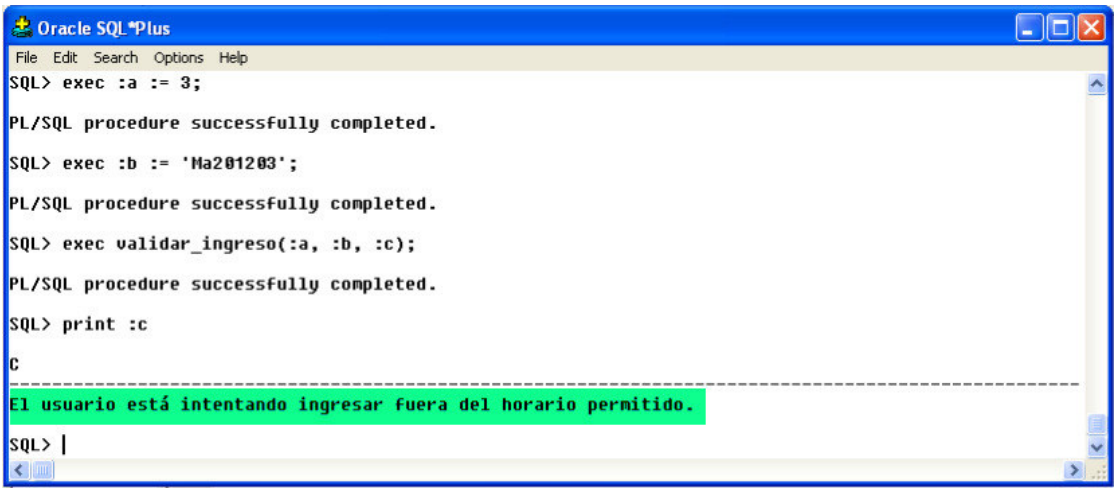

Figura 82: Resultado prueba camino 5.

Tabla 39

Prueba Camino 6

OBJETIVO		Ingresar un código de usuario que está registrado en el sistema, con estado ACTIVO, con la clave registrada en el sistema y en horas permitidas para su ingreso. Esto es para validar que el sistema le deje ingresar al sistema al haber pasado todos los filtros.
Datos	**Usuario**	6
	Clave	XXXXXX
RESULTADO		El procedimiento almacenado recorre el camino esperado, y se permite la conexión del usuario al sistema.

En la Figura 83, se muestra el resultado de la prueba del camino 6, en el que la rutina no genera mensaje de error, lo cual significa que el usuario logra conectarse al sistema exitosamente.

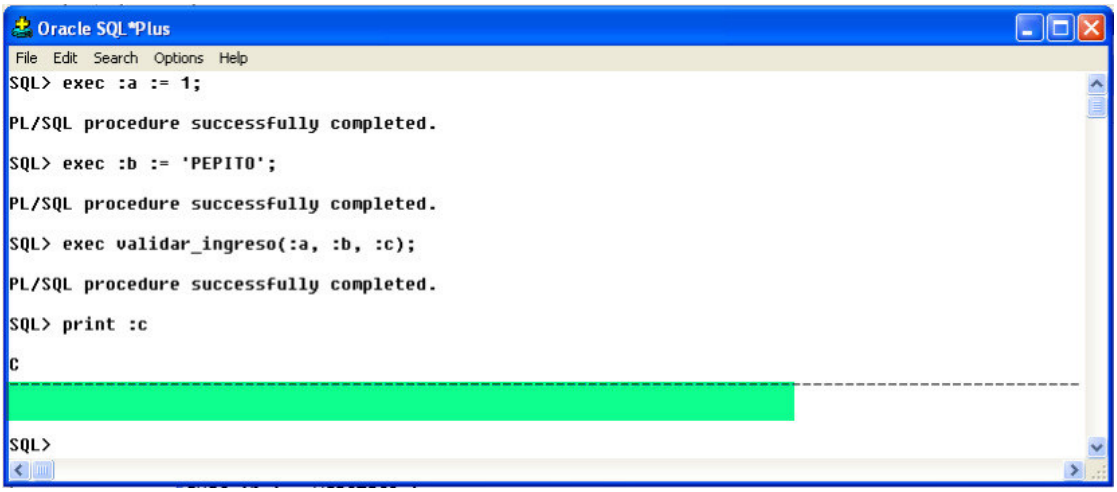

Figura 83: Resultado prueba camino 6.

4.6.2. Pruebas de caja negra

A continuación, a partir de la tabla 40 a la 44, se presenta las pruebas de caja negra, para aquellas funcionalidades más importantes del sistema, y en la parte inferior de cada una de las tablas, se muestra una figura con el respectivo resultado de cada prueba.

Tabla 40

Prueba de caja negra. Caso 1.

CASO DE PRUEBA	RESULTADO ESPERADO	RESULTADO OBTENIDO
Cada usuario tiene una clave; esta clave está almacenada en la base de datos. Una vez que el usuario intenta conectarse, debe verificar que la clave ingresada, sea igual a la clave registrada en la base de datos.	Si la clave ingresada no es igual a la clave registrada, entonces, el sistema no debe permitir el ingreso y mostrar un mensaje de error.	El sistema no permite el ingreso al sistema y muestra un mensaje, indicando que la clave no es correcta.

A continuación, en la Figura 84 se muestra el resultado de la prueba.

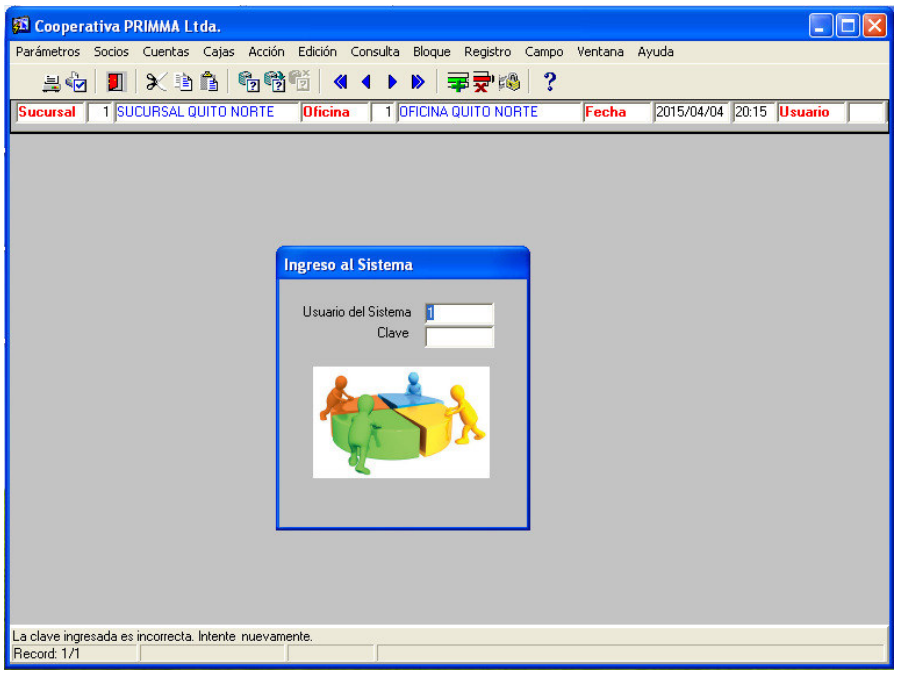

Figura 84: Resultado de la prueba de caja negra, caso 1.

Tabla 41

Prueba de caja negra. Caso 2.

CASO DE PRUEBA	RESULTADO ESPERADO	RESULTADO OBTENIDO
En el registro de usuarios y socios, es necesario validar el número de cédula de identidad de acuerdo al algoritmo de validación que aplica para Ecuador.	En primera instancia, se espera validar que el número de la cédula de identificación, tenga 10 dígitos.	El sistema valida correctamente aquellos casos donde se ingresa menos de 10 caracteres. El sistema no permite el ingreso de más de 10 caracteres.

En la Figura 85 se muestra el resultado de esta prueba.

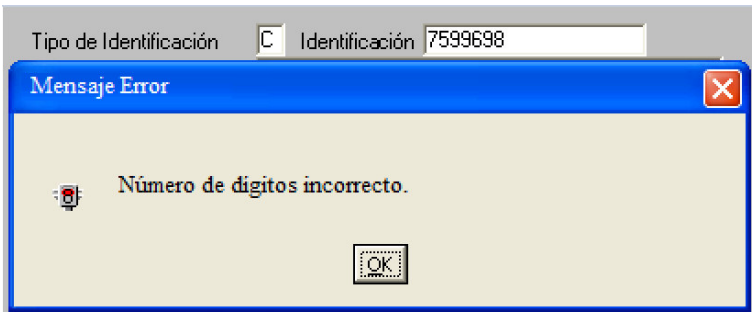

Figura 85: Resultado de la prueba de caja negra, caso 2.

Tabla 42:

Prueba de caja negra. Caso 3.

CASO DE PRUEBA	RESULTADO ESPERADO	RESULTADO OBTENIDO
Para el registro de la identificación de usuarios y socios, se debe validar que el número ingresado cumpla con el algoritmo de validación para la cédula de identidad en Ecuador.	Una vez que se ingresa el número de cédula, el sistema debe validar inmediatamente si el dígito verificador es correcto.	Se realiza varias pruebas ingresando números de cédula incorrectos y correctos, y en todos los casos la validación se realiza satisfactoriamente.

En la figura 86, se adjunta el mensaje de error que el sistema muestra al ingresar un número de cédula donde el dígito verificador es incorrecto.

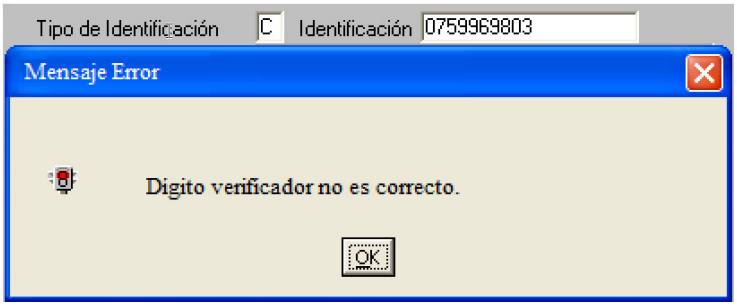

Figura 86: Prueba de caja negra, caso 3.

Tabla 43

Prueba de caja negra. Caso 4.

CASO DE PRUEBA	RESULTADO ESPERADO	RESULTADO OBTENIDO
El sistema no debe permitir el registro de un mismo socio más de una vez, sin importar el estado del socio. Esta validación se debe realizar a nivel del tipo de identificación y el número de identificación registrado en el sistema. El nombre no se utiliza para esta validación, porque pueden existir homónimos.	Si se intenta registrar nuevamente a un socio que ya fue registrado previamente, el sistema debe emitir un mensaje de error y no permitir continuar con la acción.	Cuando se intenta registrar nuevamente a un socio que ya está registrado, el sistema, muestra un mensaje de error y deja en blanco el campo de identificación, y no permite continuar hasta ingresar la identificación de un socio nuevo o al cancelar la operación, es decir salir de la transacción sin grabar.

En la Figura 87, se muestra el resultado de la prueba propuesta en la tabla 43. Donde se presenta el mensaje de error generado por el sistema.

Figura 87: Resultado de la prueba de caja negra, caso 4.

Tabla 44

Prueba de caja negra. Caso 5.

CASO DE PRUEBA	RESULTADO ESPERADO	RESULTADO OBTENIDO
En los campos donde se registra valores monetarios para realizar depósitos, retiros y transferencias, el sistema no debe permitir el ingreso de valores negativos.	En las transacciones de Depósitos, retiros y transferencias, no se debe permitir el ingreso de valores negativos. Si esto ocurre, el sistema debe emitir un mensaje de error y borrar el valor ingresado para que se corrija el error.	El sistema no permite el ingreso de valores negativos, y se emite el respectivo mensaje de error.

En la Figura 88 se muestra el mensaje de error generado por el sistema, al realizar la prueba propuesta en la Tabla 44.

Figura 88: Resultado de la prueba de caja negra, caso 5.

CAPÍTULO 5

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones:

- Al ser los casos de uso la especificación funcional, donde se establece con la cooperativa los límites de lo que se va a desarrollar y de lo que se espera del sistema, es muy importante que se tenga un buen nivel de detalle, sobre todo cuando la parte encargada de hacer el desarrollo, es externa a la cooperativa.

- Una vez obtenidos los diagramas casos de uso, se procede al desarrollo de los diagramas de secuencia, los que permiten definir fácilmente desde el punto de vista técnico como se va a desarrollar la aplicación, especialmente en los procesos que tienen alto grado de complejidad; lo cual tiene mucha importancia cuando la gente que realiza el desarrollo de la aplicación, no tiene conocimiento de cómo se maneja el negocio desde el punto de vista funcional.

- En caso de que en el equipo de trabajo, existan usuarios con experiencia previa con otros sistemas financieros informáticos, se nota que intentan hacer una réplica del sistema anterior. Por esta razón, es muy importante que el analista revise bien las propuestas del usuario, para evitar realizar sistemas engorrosos o sistemas poco funcionales.

- En un sistema financiero cooperativista, por ser su objetivo principal el manejo y administración de dinero, es muy importante que todas las transacciones realizadas sean guardadas en un registro, con la siguiente información: nombre de la persona que efectuó la transacción, la fecha y

la hora; proveyendo así de una herramienta para la ejecución de auditorías.

- Para el sistema financiero cooperativista, es muy importante que los borrados de información sean a nivel lógico, pues el borrado físico causa pérdida de información y en caso que se necesite recuperar la información borrada se debe realizar un backup (única alternativa); lo cual implica pérdida de tiempo, recursos en disco, y recursos humanos tanto en la recuperación de la información como en la revisión de la información recuperada.

- El reverso de transacciones monetarias que realiza el socio, debe generar entradas contables que representen el movimiento inverso de la transacción que se está reversando, en lugar de tan solo cambiar el estado de la transacción. Esto es porque la transacción ya pudo haber sido impresa en una libreta que describe las transacciones realizadas en la cuenta.

- La creación de nuevos productos financieros a través del uso de tablas de parámetros, que consideran todas las posibles opciones que pueden tener las cuentas a la vista, le da a la cooperativa la capacidad de crear nuevos productos financieros que se ajusten a sus necesidades, y de este modo ofrecer nuevos y atractivos productos a sus asociados.

- Con la implementación del sistema informático en la cooperativa, se logra simplificar las actividades de cada miembro, lo que le permite a la cooperativa enfocarse en la creación de nuevos productos.

5.2 Recomendaciones

- Antes de iniciar el proceso de creación de productos financieros, se recomienda realizar un análisis que identifique las características de cada producto financiero manejado por la cooperativa, considerando no dejar fuera ningún detalle que luego puede afectar al funcionamiento normal del sistema informático.

- Se recomienda manejar una política adecuada de backups, que le permita a la cooperativa recuperar información frente a fallos en: el sistema informático, base de datos o comunicaciones. La técnica de backup y recovery adoptada por la cooperativa debe ser previamente probada, para asegurarse de que la información puede ser recuperada.

- Se recomienda mantener una política adecuada de tunning a la base de datos, en la que se calendarice y ejecute periódicamente esta tarea; esto es con la finalidad de mantener un buen desempeño de las consultas a la base de datos, y de este modo garantizar una buena respuesta del sistema informático, en lo que al tiempo se refiere.

- Para garantizar un rendimiento adecuado del sistema informático, es importante que el diseño de la base de datos se lo realice con un Administrador de Base de Datos, pensando no solo en la información a ser almacenada, sino también con un enfoque a las consultas que se van a realizar en la base de datos, tanto para reportes como consultas por pantalla.

BIBLIOGRAFÍA

ALTOVA. (05 de 03 de 2015). *Diagramas de implementación UML*. Obtenido de http://www.altova.com/es/umodel/uml-deployment-diagrams.html

BOOCH, G., RUMBAUGH, J., & JACOBSON, I. (Fecha de Acceso: 15 de Marzo de 2015). *UML. El Lenguaje Unificado de Modelado*. Obtenido de http://elvex.ugr.es/decsai/java/pdf/3E-UML.pdf

CHÁVEZ, V., & OLIVARES, J. (Fecha de acceso: 16 de Marzo de 2015). *METODOLOGÍA OMT (Rumbaugh)*. Obtenido de http://www.monografias.com/trabajos13/metomt/metomt.shtml

Desconocido. (28 de 02 de 2015). *basesdedatos*. Obtenido de http://basesdatos2011.wikispaces.com/Cliente-Servidor+De+Dos+Y+Tres+Capas

ECURED. (Fecha de acceso: 20 de Marzo de 2015). *PowerDesigner*. Obtenido de http://www.ecured.cu/index.php/PowerDesigner

ERIKSSON, H.-E., & PENKER, M. (Fecha de acceso: 15 de Marzo de 2015). *EL LENGUAJE UNIFICADO DE MODELADO (UML)*. Obtenido de http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html

GARCÍA, J. (Fecha de acceso: 20 de Marzo de 2015). *DECSAI*. Obtenido de DEVELOPER: http://flanagan.ugr.es/docencia/2005-2006/2/developer/

GONZÁLEZ, J. (Enero de 2008). *¿Qué es UML?. El Lenguaje de Modelado Unificado*. Obtenido de http://www.docirs.com/uml.htm

HERNÁNDEZ, E. (Fecha de acceso: 15 de Marzo de 2015). *El Lenguaje Unificado de Modelado (UML)*. Obtenido de http://www.disca.upv.es/enheror/pdf/ActaUML.PDF

InfoMedia. (2009 de MAYO de 27). *Pruebas de Caja Blanca*. Obtenido de https://xp-dev.com/svn/jlpino-IS3/iteracion3/6_Pruebas/DocumentoDePruebasDeCajaBlanca.pdf

LAMARCA, J. (Fecha de acceso: 15 de marzo de 2015). *Lenguaje UML*. Obtenido de http://www.hipertexto.info/documentos/uml.htm

Mendoza, C. V.-U. (Fecha de acceso: 19 de Marzo de 2015). *Análisis Orientado a Objetos*. Obtenido de www.um.edu.ar/catedras/

PÉREZ, A. (06 de 09 de 2010). *Autentia*. Obtenido de Cacoo, herramienta colaborativa para hacer diagramas, incluso prototipado de pantallas o UML: http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=cacoo

PILATAXI, L. (Fecha de acceso: 20 de Marzo de 2015). *Manual de Oracle Developer*. Obtenido de http://www.monografias.com/trabajos61/manual-oracle-developer/manual-oracle-developer.shtml

Profeso Manuel Torres, Universidad de Almería. (01 de 04 de 2015). *Técnicas de prueba*. Obtenido de http://indalog.ual.es/mtorres/LP/Prueba.pdf

PUERTA, E. B. (Fecha de acceso: 20 de Marzo de 2015). *Power Designer*. Obtenido de http://elblogpuerta.blogspot.com/2012/12/power-designer.html

SIERRA, J. (12 de Mayo de 2014). *CACOO*. Obtenido de http://jcsierra04.blogspot.com/2014/05/definicion-es-una-gran-aplicacion-en.html

TicNews, R. (22 de Octubre de 2013). *Diagrama de Despliegue*. Obtenido de https://video.search.yahoo.com/video/play;_ylt=A0LEVvntowhVGiUAS4YlnIlQ;_ylu=X3oDMTB0ZjNuMHJ1BHNlYwNzYwRjb2xvA2JmMQR2dGlkA1lIUzAwM18x?p=ejemplo+de+modelos+de+despliegue&tnr=21&vid=A40FE7A1D69AD1674260A40FE7A1D69AD1674260&l=214&turl=http%3A%2F%2Fts4.mm.bin:=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3Dqa6o_OF2Sw8&sigr=11btp9dip&tt=b&tit=Diagrama+de+Despliegue&sigt=10me0ud47&back=https%3A%2F%2Fsearch.yahoo.com%2Fyhs%2Fsearch%3Fp%3Dejemplo%2Bde%2Bmodelos%2Bde%2Bdespliegue%26ei%3DUTF-8%26hsimp%3Dyhs-004%26hspar

WIKIPEDIA. (Fecha de acceso: 20 de Marzo de 2015). *ORACLE DATABASE*. Obtenido de http://es.wikipedia.org/wiki/Oracle_Database

Wikispaces. (28 de 02 de 2015). *Taller Base de Datos*. Obtenido de http://tallerbd.wikispaces.com/ARQUITECTURA+CLIENTE-SERVIDOR+DE+3+CAPAS