**Universidad de las Ciencias Informáticas**

**Facultad 3**

**Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.**

# “Desarrollo del subsistema Conciliaciones y fase 2 de los módulos Tasa de Cambio y Activo Fijo del Sistema Quarxo.”


**Autores:**

Raylith Yera Pérez

Leandro González Feal

**Tutor:**

Ing. Leonardo Vásquez Arzuaga

## DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo de tesis y se le reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma el presente a los ____ días del mes de _____ del año _____.

_____ _____

**Raylith Yera Pérez** **Leandro González Feal**

Firma del Autor Firma del Autor

_____

**Ing. Leonardo Vásquez Arzuaga**

Firma del Tutor

*"La ciencia más útil es aquella cuyo fruto es el más comunicable."*

*Leonardo Da Vinci*

Imagen: “El hombre de Vitruvio”. Autor: Leonardo Da Vinci.

**DEDICATORIA**

A mi mamá, que siempre me dio fuerzas para seguir adelante, para vencer todos los obstáculos que se pusieron en mi camino. Si hoy estoy aquí es por ti.

Ray.

A mi madre, mi guía, mi amiga, mi todo por ser tan maravillosa.

A hermana y a mis abuelos Librada y Antonio.

Leo.

**AGRADECIMIENTOS**

A mi familia y sobre todo a mi hermana por ser mi amiga. A Yanet, por aguantarme, aceptarme y apoyarme siempre; gracias por tu amistad. A Maritza por cambiarme la vida y creer en mí. Gracias a Audrey, Kirenia y Vicente por tantos buenos y malos momentos que hemos pasado juntos, por siempre estar para mí cuando yo más lo necesitaba.

A todos mis compañeros de aula. En especial a Liset, Rosalia, Yadini e Iraidis, por mantener la amistad a pesar de nuestras diferencia. Gracias a todos por los mejores 5 años de mi vida.

A los profesores que tanto me han enseñado en este tiempo y por su puesto a mi tutor que siempre ha estado ahí.

Ray.

Muchas gracias:

A mi madre, por su apoyo incondicional, su ternura, su confianza, su entrega, por darme las fuerzas necesarias, pocas serían las palabras con las cuales describir lo mucho que significas en mi vida.

A mi hermanita linda, de la cual estoy muy orgulloso.

A toda mi familia por su apoyo cuando lo he necesitado, mis tíos, mis primos que son más que eso.

A mi novia por su paciencia y su entrega, por estar siempre ahí.

A mi padre y mis hermanos por su apoyo desde lejos.

A mi tutor por todo lo que aprendí a su lado y su ayuda incondicional.

A mis amigos, los verdaderos, los que no necesitan un nombre, gracias por quererme tan incondicionalmente.

A todos los profesores del proyecto, que de una forma u otra siempre me ayudaron.

A mis compañeros de grupo , los del apto que me soportaron estos 5 años, a quienes quiero mucho y nunca voy a olvidar, gracias por los momentos de alegría, por enseñarme, por compartir esta etapa inolvidable.

A muchos de los profesores que eh tenido los que han contribuido en mi formación profesional y en la culminación de esta investigación.

A mi grupo de teatro, con los cuales pasé momentos inolvidables.

A todos, gracias.

Leo.

**RESUMEN**

En el Banco Nacional de Cuba se lleva a cabo la inserción de Quarxo, un sistema de gestión bancaria. Dentro de sus funciones de banco comercial se encuentra como principal actividad dejar constancia del cotejo de cada importe abonado o cargado por el Banco, con cada importe registrado por la entidad, de cada cuenta bancaria que esta opere, el proceso de verificación y confrontación de estas transacciones, es conocido como conciliación bancaria. Este proceso es realizado manualmente, en el cual se necesita de mucho tiempo y recursos, así como la ocurrencia de errores humanos. La actualización, cancelación, inserción en el extranjero de los activos fijos y la comprobación de la correcta actualización de la tasa de cambio también están dificultadas porque el banco no lo realiza, siendo de vital importancia llevar este control.

El presente trabajo se propone implementar un subsistema para la gestión de la conciliación bancaria y la actualización de los módulos Activo Fijo y Tasa de Cambio, que formarían parte de la versión 2.0 de Quarxo. Para la implementación fue necesario el estudio, profundización y utilización de tecnologías y herramientas entre las que se encuentran fundamentalmente: Spring como Framework núcleo, Eclipse como Entorno Integrado de Desarrollo y como lenguaje de programación Java.

Con la implementación del subsistema y la actualización de los módulos será posible viabilizar la gestión de la conciliación, el control de los activos fijo y la comprobación de la tasa de cambio en el sistema de gestión bancaria Quarxo para el Banco Nacional de Cuba.

**Palabras clave:** Activos fijos, conciliación bancaria, tasa de cambio, Quarxo

**TABLA DE CONTENIDO**

## INTRODUCCIÓN

El surgimiento de los bancos está dado a partir de la necesidad de las personas de que financiaran sus ideas y proyectos, inicialmente para realizar simples operaciones de cambio y crédito a niveles personales; pero pronto se comenzaron a desarrollar funciones más amplias, pasando a ser instituciones más complejas. La complejidad en que se desarrollan las actividades bancarias ha llevado a la necesidad de informatizar la mayor cantidad de procesos, logrando un mejor control del enorme flujo de informatización que se maneja en la sociedad moderna; todo esto dado por el cambio y la adaptación del sistema financiero al nuevo entorno competitivo.

A partir de la modernización de los sistemas bancarios en Cuba, diferentes bancos existentes en el país como el Banco Central de Cuba (BCC), Banco Popular de Ahorro (BPA), Banco de Crédito y Comercio (BANDEC) empezaron a contar con sistemas informáticos que le ayudaran en la gestión de sus procesos, con una infraestructura de comunicación entre ellos, a partir de la Sociedad para Telecomunicaciones Financieras Interbancarias Mundiales (SWIFT). El Banco Nacional de Cuba (BNC) se suma a toda esta red contando actualmente con más de 200 sucursales por todo el país, y otras pocas sucursales internacionales.

La Universidad de las Ciencias Informáticas (UCI) es una de las máximas responsable del proceso de modernización llevado a cabo en Cuba, por lo que se le encomendó la tarea de la creación y desarrollo de un sistema para el BNC, con el fin de sustituir SABIC (Sistema Automatizado para la Banca Internacional y de Comercio). Debido a que el sistema SABIC en su versión MS-DOS[1] presenta limitaciones por ser desarrollado sobre FoxPro[2] ya que no permite agregar nuevas funcionalidades, ni hacer modificaciones necesarias para el buen funcionamiento del sistema bancario. Así surgió Quarxo, el cual se encuentra ya en su segunda versión con el fin de mejorar su rendimiento y las necesidades de los trabajadores; facilitando el trabajo y la eficiencia de todas las actividades llevadas a cabo.

El BNC dentro de sus funciones de banco comercial tiene como una de sus principales actividades respaldar las transacciones financieras que el estado cubano asume para lograr obtener bienes o

[1]MS-DOS: Sistema operativo mono tarea y mono usuario para ordenadores personales.

[2]FoxPro: Lenguaje privativo propiedad de la Microsoft.

servicios, fundamentalmente para la compra de alimentos, combustible, medicamentos, así como cualquier otro recurso de importancia para el país en todos los sectores. El proceso de verificación y confrontación de las transacciones hechas entre los propios bancos es el que se conoce como conciliación bancaria, lo cual permite comparar los valores que los bancos tienen registrados en sus cuentas de ahorros o cuentas corrientes, con los valores que el banco les suministra por medio del extracto bancario que suele recibirse cada mes.

Actualmente este proceso se realiza manualmente lo que trae como consecuencia que ocurran errores humanos, así como la no coincidencia de las referencias e importes, dando lugar a que no cuadre el cierre de conciliación; además de un gran gasto de recursos y tiempo. El gasto aproximado en un año es de 36 cajas de hojas de papel continuo y 24 paquetes de papel "Bond", costando cada uno de ellos 20.8 y 4.35 cuc respectivamente, provocando un gasto total de 853.20 cuc. Además el volumen de transacciones diarias entre el BNC y otras entidades bancarias oscila entre los 1000 y 5000; estas operaciones son realizadas a través de los diferentes tipos de mensajería. Esta cantidad de información circulando conlleva a que el proceso de conciliación sea demasiado extenso, una persona puede demorarse 15 días o más identificando las diferencias entre las operaciones [5].

Además todos los procesos para gestionar los activos fijos existentes en la entidad, así como en sus sucursales extranjeras no están del todo informatizados; por lo que no se tiene un control absoluto de los mismos. También ocurre que a la hora de actualizar la tasa de cambio se producen errores dando lugar a que no se actualice correctamente, ocasionando errores en el momento que se produce la conversión de monedas para las transacciones.

Por lo que se plantea como **problema a resolver**: ¿Cómo controlar los activos fijos, la confirmación de la tasa de cambio y el estado de las cuentas del sistema de gestión bancario Quarxo?

Como **objeto de estudio:** los procesos de contabilidad y conciliación bancaria enmarcados en el **campo de acción:** informatización de los procesos de contabilidad y conciliación bancaria mediante sistemas informáticos en el Banco Nacional de Cuba.

Por lo que el **objetivo general:** es el desarrollo de componentes y funcionalidades para la gestión de activos fijos y los procesos de conciliación bancaria, así como la confirmación de la tasa de cambio para Quarxo.

Teniendo como **objetivos específicos**:

- ✓ Realizar el diseño teórico metodológico de la investigación
- ✓ Diseñar e implementar el subsistema Conciliaciones y la versión 2.0 de los módulos Tasa de Cambio y Activo Fijo del subsistema de Contabilidad de Quarxo
- ✓ Validar la propuesta de solución

**Resultados esperados:**

- ✓ Artefactos generados en el proceso de desarrollo de los módulos Tasa de Cambio y Activo Fijo del subsistema de Contabilidad y el desarrollo del subsistema de Conciliación para el sistema bancario Quarxo
- ✓ Un nuevo subsistema de Conciliación Bancaria para el sistema bancario Quarxo
- ✓ Mejoras en los módulos de Tasa de Cambio y Activo Fijo del subsistema de Contabilidad del sistema bancario Quarxo

**Estructura del documento:**

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Se realiza una investigación sobre los procesos de negocios asociados a la contabilidad y la conciliación bancaria, así como los principales conceptos a tratar sobre activo fijo y tasa de cambio. Se lleva a cabo un estudio de los sistemas existentes en el mundo que utilizan la conciliación bancaria. Además se especifican las metodologías, técnica, herramientas, lenguaje y frameworks a utilizar que conforman el ambiente de desarrollo del sistema.

CAPÍTULO 2: CARACTERISTICAS DEL SISTEMA, ANÁLISIS Y DISEÑO

Se modela todo el negocio, definiéndose los requisitos, realizando el análisis y el diseño de todo el sistema implementado; generándose los artefactos de salida del subsistema de Conciliaciones Bancarias y los módulos Activo Fijo y Tasa de Cambio del subsistema Contabilidad del sistema Quarxo.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

Este capítulo se centra en la implementación del subsistema Conciliaciones y los módulos Activo Fijo y Tasa de Cambio del subsistema Contabilidad, así como la realización de la validación de la solución propuesta mediante las diferentes pruebas.

# CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

## 1.1. Introducción

En este capítulo se abordarán los aspectos fundamentales que servirán de soporte teórico para el desarrollo de toda la investigación. Primeramente se realiza un análisis de los diferentes conceptos a tratar, se analizan algunas soluciones asociadas al tema de conciliación bancaria. Además se verá una breve descripción de las herramientas, técnicas, lenguajes, frameworks, así como el modelo de desarrollo a utilizar en la solución del problema planteado.

## 1.2. Marco conceptual

### 1.2.1. Contabilidad

Ciencia que tiene por objeto el registro de las operaciones económicas efectuadas por una persona o entidad, con el fin de conocer sus resultados y la situación de la misma [1].

Técnica en constante evolución, basada en conocimientos razonados y lógicos que tienen como objetivo fundamental, registrar y sintetizar las operaciones financieras de una entidad e interpretar los resultados [2].

En el presente trabajo se tomará la contabilidad como: disciplina o rama del conocimiento que permite controlar las operaciones de la entidad, suministrando datos útiles de actividades vinculadas con la gestión de bancos, comisiones, nomencladores, activos fijos, así como en diferentes tipos de cuenta, además de operaciones relacionadas con las transacciones contables, libro contable y tasa de cambio.

#### 1.2.1.1 Tasa de Cambio

Una tasa es un coeficiente que expresa el vínculo existente entre dos magnitudes. Cambio, por su parte, es sustituir una cosa por otra. Por lo que tasa de cambio es la cantidad de unidades de moneda nacional que se debe entregar a cambio de una unidad de moneda extranjera. La tasa de cambio puede ser real o nominal [36].

#### 1.2.1.2 Activo Fijo

Representan bienes o derechos de cierta permanencia en la empresa, que se adquieren no con la intención de venderlos, sino de utilizarlos en las operaciones normales del negocio [2].

Dentro de los activos fijos se tienen dos grupos: los tangibles y los **i**ntangibles

- ✓ Tangibles: Comprende las propiedades o bienes susceptibles de ser tocados
- ✓ Intangibles: Incluye cosas que no pueden ser tocadas materialmente

### 1.2.2. Conciliación Bancaria

La conciliación bancaria es un proceso sistemático de comparación entre los ajustes contables de una cuenta corriente realizada por el banco y la cuenta de bancos correspondientes en la contabilidad de una empresa, con una explicación de las diferencias encontradas [3].

#### 1.2.2.1. Proceso de conciliación

El proceso de conciliación permite confrontar y conciliar los valores que la empresa tiene registrados de una cuenta de ahorros o corriente con los valores que el banco suministra por medio del extracto bancario. Las empresas tienen un libro auxiliar de bancos en el cual se registran cada uno de los movimientos realizados en una cuenta bancaria, como son: el giro de cheques, notas débito, notas crédito, anulación de cheques y consignaciones. La entidad financiera donde se encuentra la respectiva cuenta almacena un registro completo de cada movimiento que el cliente (la empresa) hace en su cuenta. Al enviar el banco el extracto mensual a la entidad y no coincidir los valores con los del libro auxiliar de la misma, comienza el proceso de verificación y confrontación [4].

#### 1.2.2.2. Proceso de conciliación en el BNC

Mensualmente los bancos exteriores envían al BNC un extracto en el que se muestran todos esos movimientos que concluyen en un saldo de la cuenta el último día del respectivo mes. Por diferentes motivos el saldo del extracto bancario no coincide casi nunca; el proceso de conciliación no es más que verificar estos motivos, los cuales pueden ser:

- ✓ **Cheques pendientes o en tránsito:** cheques emitidos por la empresa y no cobrados en el Banco por el beneficiario del mismo. Por lo que están abonados en libros, pero no cargados en el Estado de Cuenta del Banco.
- ✓ **Depósitos en tránsito:** generalmente corresponden con depósitos enviados por correo a fin de mes o que por alguna causa no hayan llegado al Banco.

- ✓ **Notas de Débito:** cargos hechos por el Banco por diversos conceptos (intereses, comisiones, giros descontados, devueltos, cheques recibidos de clientes y devueltos por el Banco) que por no haberse recibido del Banco la nota de débito respectiva (generalmente por correos) no se ha abonado en los libros de la entidad.
- ✓ **Notas de Crédito:** abonos hechos por el Banco (descuento de giros, pignoraciones, pagarés) que por no haberse recibido la nota de crédito no se han cargado en los libros.
- ✓ **Errores:** puede suceder tanto en los registros de la empresa como en los del Banco ya que al registrarse cualquier operación puede colocarse una cantidad distinta.
- ✓ **Cargos o abonos incorrectos:** puede originarse por depósitos o cheques de bancos con los que la empresa lleva cuenta, los cuales por error se carguen o abonen a otro Banco distinto o que el Banco cargue o abone operaciones que corresponden a otra cuenta cliente del Banco.
- ✓ **Otras diferencias:** Algún otro tipo de diferencia que ocurre con menor frecuencia.

#### 1.2.2.3. Comunicación entre bancos

SWIFT es una sociedad cooperativa privativa a través de la cual el sector financiero lleva a cabo sus operaciones de negocios de forma rápida, segura y fiable; gracias a una red internacional de comunicaciones entre bancos y otras entidades. Más de 10 000 entidades financieras y sociedades de 212 países depositan su confianza en este sistema cada día para intercambiar millones de mensajes estandarizados. Su sede central se encuentra en Bélgica y cuenta con oficinas en los principales centros financieros y mercados en desarrollo del mundo; no posee fondos ni gestiona cuentas en nombre de los clientes, ni tampoco almacena información financiera de forma permanente.

SWIFT permite automatizar y estandarizar las transacciones financieras y de este modo, reducir tanto los gastos como el riesgo operativo y eliminar ineficiencias en sus operaciones. Esta actividad implica el intercambio seguro de los datos privados de tal forma que se garanticen su confidencialidad e integridad [4].

#### 1.2.2.4.Sistemas de conciliación bancaria

Para la captura de requisitos, así como ver la dinámica y prototipos de interfaz de usuario que proponen se estudiaron diferentes sistemas existentes en el mundo que realizan la conciliación bancaria. Tal es el

caso de sistemas como SIC.solucions, Sistema Isis Classic, Winledger, SICOB, Conciliación Express, SIAFI y en el caso de Cuba un módulo del Banco Metropolitano.

**SIC.solucions**

Esta herramienta web fue creada en Barcelona por expertos en soluciones profesionales *open source* con más de 30 años de experiencia y constante innovación. Tiene integrado un sistema de conciliación que posibilita en gran medida la mejora de la gestión de los procesos contables [6].

Este sistema de conciliación bancaria, además de la integración automática de la información la cual mantiene un registro histórico de movimientos bancarios, permite el intercambio de datos electrónicos con los bancos; tiene como objetivos principales:

- ✓ Automatizar el proceso de conciliación bancaria, reduciendo el tiempo destinado a esta tarea.
- ✓ Simplificar y agilizar el proceso de conciliación, a la vez que le confiere seguridad en la comprobación de los movimientos.

**Sistema Isis Classic**

Es un software de gestión comercial diseñado para la administración eficiente de las pequeñas y medianas empresas con el objetivo de obtener el balance, mayor y libro diario de la empresa con la posibilidad de operar con centros de costos y en distintas monedas. Sin límite de empresas permitiendo consolidarlas en forma automática [7].

En el mismo momento de la conciliación permite dar de alta a los débitos y créditos detectados en el extracto, actualizando:

- ✓ Libro banco
- ✓ Libro de IVA (Impuesto al Valor Agregado)
- ✓ Ingresos Brutos
- ✓ Contabilidad general

**Winledger**

Desarrollado en Venezuela, cumple con las normas y especificaciones descritas por las leyes de comercio establecidas en el país y ha sido implementado en empresas de diversas actividades económicas.

Es un módulo muy práctico y fácil de usar, donde conciliar sus bancos deja de ser una tarea tediosa y pasa a ser una labor sencilla. Este módulo realiza la conciliación bancaria automática entre el estado de cuenta del banco y los registros correspondientes del módulo de Contabilidad [8].

Características Generales:

- ✓ Se alimenta de la información del módulo de contabilidad Winledger
- ✓ Incluye entre sus módulos la posibilidad de hacer uso de los estados de cuenta que su empresa descarga a través de los sitios bancarios, eliminando así el procesamiento manual de los estados de cuenta
- ✓ Conciliación automática, a través de números de referencias
- ✓ En caso de divergencia en los números de referencias, el sistema permite conciliaciones manuales en forma automatizada

**SICOB**[3]

Es un software desarrollado para las áreas de tesorería, contraloría o contabilidad de las empresas que le permitirá automatizar su proceso de conciliación [9].

Funcionalidades:

1. Carga de Información
2. Conciliación Automática
3. Conciliación Manual
4. Conciliación por Rango de Ajuste
5. Conciliación con Soporte escrito

**Conciliación EXPRESS**[4]

Es un sistema de conciliación bancaria automatizado sencillo y ajustable individualmente para cada cliente, optimizando todo el proceso de conciliación. El sistema está hecho para dar a los clientes las mayores ventajas posibles y optimizar el proceso para cada una. Incluye el almacenamiento de las transacciones en tráfico para que puedan conciliarse en el mes posterior [10].

[3]SICOB:Sistema de conciliación bancaria

[4]EXPRESS: Sistema Automático de Conciliación de Cuenta en línea

**SIAFI**[5]

La aplicación se basa en la creación de tablas básicas que permitirán codificar los atributos de los principales componentes del proceso de conciliación bancaria automática y establecer vinculaciones entre codificaciones básicas. Utiliza la mensajería SWIFT, mediante la determinación de estándares, la emisión, la utilización de formas y formatos, así como el modo de la alimentación de los datos “en línea” al software [11].

**Módulo de Conciliación del Banco Metropolitano (BM)**

El uso de los Sistemas Automatizados Contables resulta común en las entidades cubanas, orientados fundamentalmente a obtener una mayor eficiencia en la gestión contable empresarial. En el sector financiero bancario, existen entidades que ya han implementado la conciliación bancaria en su versión automatizada, pero ajustadas a las necesidades específicas de cada institución; tal es el caso del Banco Metropolitano (BM) el cual presenta un sistema de conciliación.

El módulo de conciliación del BM es un software desarrollado e incluido dentro del paquete de servicios que brinda el Banco Metropolitano. El sistema facilita llevar un control detallado de las acciones operativas y administrativas relacionadas con la conciliación bancaria. Además de las operaciones comunes que se realizan durante el proceso de conciliación.

## 1.3. Metodologías, lenguaje de modelado, frameworks y herramientas utilizadas

Es importante conocer el contexto donde se utilizará un software, para una apropiada selección de las herramientas y tecnologías a utilizar influyendo considerablemente lo que el cliente desee, el tiempo de desarrollo y la calidad final del producto.

### 1.3.1. Metodologías de desarrollo

Se entiende por metodología de desarrollo una colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software, estableciendo rigurosamente las actividades, artefactos, herramientas y notaciones a utilizar.

El Centro de Informatización de la Gestión de Entidades (CEIGE) perteneciente a la facultad 3 de la UCI, se especializa en la producción de proyectos por lo general de gran magnitud; por lo que se le hizo

[5]SIAFI: Sistema Integrado de Administración Financiera

necesario crear su propio modelo de desarrollo para estandarizar las fases por las que se debe pasar y los artefactos a desarrollar. El Modelo de desarrollo de software para el CEIGE especifica las actividades de cada una de las fases del ciclo de vida de los proyectos del centro teniendo en cuenta los procesos de CMMI[6] nivel dos. Dicho modelo cuenta con dos fases: Inicio o Estudio preliminar y Desarrollo. La fase de desarrollo cuenta con 6 disciplinas las cuales son: Modelado del negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación. Cada fase del mismo detalla los objetivos, hitos y artefactos a generar en cada momento, independientemente de las herramientas o métodos que se utilicen para ello; destaca su flexibilidad y robustez [12].

**Inicio o Estudio preliminar**: Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto.

**Desarrollo:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.

### 1.3.2. Lenguaje y herramientas para el modelado

#### 1.3.2.1. Notación para la Modelación de Procesos de Negocio

Diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades, BPMN es una notación gráfica utilizada para la descripción de la lógica definiendo un Diagrama de Procesos de Negocio. Proporcionando un lenguaje común para que las partes involucradas puedan entenderse y comunicar los procesos de forma clara, completa y eficiente [13].

#### 1.3.2.2. Lenguaje Unificado de Modelado (UML) v2.0

Se denomina lenguaje de modelado de objetos al conjunto estandarizado de símbolos y las distintas combinaciones de ellos para modelar el software a lograr. El lenguaje utilizado para modelar elementos de

[6]CMMI:CapabilityMaturityModelIntegration (Integración de Modelos de Madurez de capacidades)

software es UML[7]; este permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Se caracteriza por dividir a los sistemas en una estructura estática con un comportamiento dinámico.

#### 1.3.2.3. Visual Paradigm v6.4

Se tomó Visual Paradigm como herramienta para el modelado visual. La misma facilita el trabajo durante la confección de un software, garantiza la calidad del producto final, además de proporcionar características tales como generación del código, ingeniería inversa y generación de informes y soporta el ciclo de vida de desarrollo del software completo: análisis, diseño, implementación, pruebas y despliegue [14].

### 1.3.3. Eclipse v3.5

Como IDE (entorno de desarrollo integrado) se utilizará Eclipse, contando con un bajo consumo de recursos. Es un entorno de desarrollo integrado y multiplataforma; el mismo permite la integración de diversos lenguajes sobre un mismo IDE debido a la arquitectura de *plugins* que posee. También permite introducir otras aplicaciones que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías [15].

### 1.3.4. Apache Tomcat v6.0

Apache Tomcat es el contenedor Web usado como soporte para el funcionamiento de aplicaciones desarrolladas en Java. Tomcat es una implementación libre y de código abierto de las tecnologías Servlets y JSP desarrollada bajo el proyecto Jakarta de la Fundación Apache. Es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad [5].

### 1.3.5. SQL Server 2005

Como gestor de base de datos a pesar de ser SQL Server un software privativo, fue el que se utilizó; ya que fue así impuesto por el cliente para poder acelerar el proceso de desarrollo con la utilización del núcleo de SABIC el cual incluye un grupo de tablas y procedimientos almacenados que ayudarán en la implementación del sistema.

[7]UML:UnifiedModelingLanguage (Lenguaje de modelado unificado)

SQL Server 2005 provee herramientas sólidas, reduciendo la complejidad de la creación, despliegue, administración y uso de aplicaciones analíticas y de datos empresariales en plataformas que van desde los dispositivos móviles hasta los sistemas de datos empresariales. Permite trabajar en modo cliente-servidor, donde la información y datos se alojan en el servidor y los terminales o clientes de la red sólo acceden a la información, además permite administrar información de otros servidores de datos. A través de un conjunto global de características, la interoperabilidad con sistemas existentes y la automatización de tareas rutinarias, SQL Server 2005 ofrece una solución completa de datos para empresas de todos los tamaños [16].

## 1.4. Ambiente de desarrollo

### 1.4.1. Lenguaje: Java v1.6

Java es el lenguaje de programación en el cual se desarrolló el sistema Quarxo y fue definido así en la arquitectura del proyecto. Se seleccionó el mismo atendiendo al conocimiento y preparación del equipo de desarrollo, además de la existencia de algunas librerías, las cuales facilitaron el trabajo siendo más rápido y logrando una arquitectura robusta que se adaptaba a las necesidades del cliente.

Java es un lenguaje de programación orientado a objetos el cual tiene similitud con C# y C++, robusto, poderoso y fácil de aprender. Es muy utilizado por su adaptabilidad a diferentes plataformas, además de ser flexible e integrable a frameworks de desarrollo y de código abierto. Java fue diseñado para softwares altamente confiables, establecer conexiones con servidores y crear aplicaciones tanto de red como de escritorio; aplicaciones dinámicas con acceso a bases de datos [17].

#### 1.4.1.1. JavaScript

**JavaScript** es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico; el cual permite crear funcionalidades específicas en las páginas web, es utilizado mayormente del lado del cliente [18].

#### 1.4.1.2.Lenguaje de Consulta Hibernate (HQL)

Hibernate ofrece un lenguaje de consulta de datos potente (HQL) que se parece a SQL; sin embargo, comparado con SQL, HQL es completamente orientado a objetos y comprende nociones como herencia, polimorfismo y asociación. En HQL, las consultas no son sensibles a mayúsculas, a excepción de los nombres de las clases y propiedades Java [19].

### 1.4.2. Plataforma J2EE

*Java Platform, Enterprise Edition* (Plataforma Java, Edición Empresarial), es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en lenguaje de programación Java. Las aplicaciones desarrolladas en esta plataforma tienden a ser portables, escalables, robustas y seguras. Define una arquitectura utilizando un modelo multicapa.

Es una plataforma precisa, un estándar para crear aplicaciones empresariales usando un modelo multicapas. Simplifica las aplicaciones empresariales organizándolas en componentes modulares y estandarizados; dividiendo la aplicación en diferentes niveles especializándose cada uno en una tarea en particular; permitiendo así realizar aplicaciones distribuidas complejas. Proporciona además un conjunto completo de servicios a estos componentes, y maniobra muchas de las funciones de la aplicación de forma automática, sin necesidad de una programación compleja [20].

## 1.5. Tecnologías y Frameworks

### 1.5.1. Frameworks

Un framework (marco de trabajo) no es más que un Generador de Aplicación que se relaciona directamente con un dominio específico, este no tiene funcionalidades de una aplicación específica, sino que las aplicaciones se construyen sobre él [21].

### 1.5.2. Spring

Se utilizará Spring MVC[8] v2.5 como framework núcleo de la aplicación, para atender las peticiones web simples con navegación lineal. Spring MVC brinda una jerarquía de clases “controladora” para recibir dichas peticiones, los diferentes tipos responden a diferentes momentos en la recepción de peticiones como [22]:

- ✓ *AbstractCommandController* (Control de comando abstracto)
- ✓ *SimpleFormController* (Controlador de formulario simple)
- ✓ *WizardController* (Controlador asistente)
- ✓ *MultiActionController* (Controlador de acción múltiple)

[8]MVC: Modelo Vista Controlador

### 1.5.3. Spring WebFlow[9] v1.9.7

Surge como una respuesta a la funcionalidad limitada del flujo de página ofrecida por los frameworks MVC clásicos. Los flujos Web en este framework son diseñados para ser auto-controlados, dando la posibilidad de definir reglas de navegaciones (múltiples y complejas).

En Spring WebFlow un flujo maneja la conversación (ámbito nuevo que define el vacío entre Sesión y Petición) completa, desde que inicia hasta que culmina y en este punto limpia la memoria automáticamente siempre y cuando se termine el flujo, lo cual es una mejoría ya que el usuario no tiene que preocuparse por borrar los recursos en memoria que no estén siendo utilizados.

Siempre y cuando el caso de uso que se desarrolle presente un flujo complejo y/o no lineal se debe utilizar Spring WebFlow.

Cuando existen funcionalidades con flujos complejos y estos son utilizados más de una vez por otras funcionalidades entonces se puede valorar la creación de flujos independientes para que sean reutilizados [22].

**Spring DAO y DAO genérico**

El framework Spring propone la utilización del patrón DAO (objeto de acceso a datos) para interactuar con la base de datos. Este soporte lo brinda a través de "*Spring DAO*". El patrón DAO tiene como objetivo separar la manera en que la capa de acceso a datos resuelve el manejo de los datos persistentes con la lógica de la aplicación.

Por otra parte, existe un componente nombrado "*DAO genérico*" que tiene programadas las funciones básicas para interactuar con la base de datos (Inserción, Actualización, Eliminación y Consultas por criterios) [22].

**Spring JDBC[10] y StoreProcedureDAOSupport[11]**

[9] Spring WebFlow: Es un complemento de Spring MVC que permite el manejo de aplicaciones Webs

[10] JDBC: Java DatabaseConnectivity (conectividad de base de datos java)

[11] StoreProcedureDAOSupport: Soporte de procedimiento almacenado DAO

Spring JDBC es un módulo perteneciente al framework Spring. El mismo posee librerías de clases para trabajar con base de datos a través del API[12] de Java JDBC. Unas de las características principales de este módulo es el soporte que brinda para invocar procedimientos y funciones almacenadas. Además el componente "*StoreProcedureDAOSupport"* fue desarrollado para facilitar la interacción con los procedimientos y funciones almacenadas utilizando las ventajas que brinda el módulo Spring JDBC [22].

**Spring Transaction**

Una transacción, es la manera de agrupar una o varias tareas en una única y consistente unidad de trabajo, de forma tal que no puede verse el funcionamiento de ellas por separado; sino como un todo armónico. En el caso de que falle algunas de estas tareas se deshacen las tareas ejecutadas inicialmente en esa transacción y se vuelve al estado inicial.

Spring Transaction (transacción) brinda soporte para asegurar las ejecuciones de las funcionalidades de la aplicación. Además de soportar el manejo de transacciones para varios gestores de base de datos y recursos compartidos a través del API JTA, también se pueden definir las transacciones en el código de la aplicación y declarativamente en los ficheros de declaración [22].

### 1.5.4. Hibernate v3.5

Hibernate es un framework para resolver el problema de persistencia de datos en Java, entre la aplicación y la base de datos relacional, dejando al desarrollador concentrarse en los problemas de la aplicación. Es utilizado para el trabajo con la base de datos, considerado un *ObjectRelationMapping* (Mapeo de objeto relacional, ORM). Una de las características principales es que abstrae elegantemente el trabajo con la base de datos, soportando la manipulación de los datos persistentes objetualmente, a través de ficheros que mapean clases Java contra tablas [22].

### 1.5.5. DojoToolkit v1.3

Potente framework que contiene APIs y controles para ayudar al desarrollo de aplicaciones web. Incluye un sistema de empaquetado, los efectos visuales de la interfaz de usuario, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX[13]. Proporciona además variadas

[12]API: AplicationProgramming Interface (Interfaz de programación de aplicaciones)

[13]Ajax: (Asynchronous JavaScript And XML) es una técnica de desarrollo web para crear aplicaciones interactivas.

opciones en una sola biblioteca haciendo un mejor trabajo que sustenta los nuevos y viejos navegadores, resolviendo los problemas de compatibilidad. Tiene múltiples puntos de entrada, es independiente del intérprete y unifica estándares de codificación [23].

Se seleccionó Dojo debido a la fácil integración con Spring MVC. Además se tuvo en cuenta que el proyecto SIGEP desarrolló componentes y funciones las cuales se podían reutilizar en el desarrollo del sistema Quarxo.

### 1.6. Patrones

Un patrón es una unidad de información nombrada, instructiva e intuitiva que captura la esencia de una familia exitosa de soluciones probadas a un problema recurrente dentro de un cierto contexto. El objetivo de los patrones es crear un lenguaje común a una comunidad de desarrolladores para comunicar experiencia sobre los problemas y sus soluciones. Pueden referirse a distintos niveles de abstracción, desde un proceso de desarrollo hasta la utilización eficiente de un lenguaje de programación [24].

Un buen patrón debe:

- ✓ Solucionar un problema
- ✓ Ser un concepto probado
- ✓ La solución no es obvia
- ✓ Describe participantes y relaciones entre ellos
- ✓ Tiene un alto componente humano: estética y utilidad

Los patrones de diseño de software son una solución probada para un problema general de diseño, en un contexto determinado. Encierran la experiencia de los programadores e ingenieros que han ido adquiriendo en las soluciones de problemas comunes.

#### 1.6.1. Patrones arquitectónicos

Un patrón arquitectónico determina la arquitectura global de una aplicación, proporcionando un esquema genérico probado para solucionar un problema en específico. Los mismos proveen un vocabulario común y comprensible, facilitan la construcción de software con propiedades particulares y ayudan a construir

arquitecturas heterogéneas y complejas. Con este fin se utiliza en Quarxo para la definición de su arquitectura el patrón MVC [24].

**Modelo Vista Controlador**

Este patrón propone dividir la aplicación en tres capas:

- ✓ **Modelo**: Administra el comportamiento y los datos del dominio de aplicación
- ✓ **Vista**: Maneja la visualización de la información, esto es típicamente en HTML
- ✓ **Controlador**: Controla el flujo entre la vista y el modelo. Escucha los cambios en la vista y los envía al modelo, el cual le regresa los datos a la vista formando un ciclo

Al aplicarse este modelo, la vista y el controlador se abstraen del acceso a los datos, del cual se encarga la capa de dominio [24].

### 1.6.2. Patrones de diseño

El diseño hace uso de varios patrones, los cuales proponen una solución efectiva y probada para un problema general dentro del marco del diseño de software.

#### 1.6.2.1. Gangs of Four

El grupo de *Gangs of Four* (pandillas de cuatro, GoF) clasificó los patrones en tres grandes categorías basadas en su propósito:

- ✓ Creacionales: patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- ✓ Estructurales: los patrones estructurales describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- ✓ Comportamiento: los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

**Patrón *Singleton* (semifallo):** Quizás el más sencillo de los patrones que se presentan en el catálogo de GoF, también uno de los más utilizados; a menudo es implementado por otros patrones. Patrón creacional diseñado para restringir la creación de objetos pertenecientes a una clase. Su objetivo es garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a esta instancia.

**Patrón *Facade* (fachada):** Patrón estructural que trata de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que hace de pantalla o fachada. El objetivo de este patrón es proveer un intermediario entre dos grupos de objetos, disminuyendo el número de objetos con los que trabaja un cliente y simplificando el acceso de éste a un conjunto de objetos relacionados; proporcionando una interfaz unificada de alto nivel que, representando a todo un subsistema, facilite su uso. La fachada satisface a la mayoría de los clientes, sin ocultar las funciones de menor nivel a aquellos que necesiten acceder a ellas.

**Patrón Cadena de Responsabilidad:** La cadena de responsabilidad se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición [25].

#### 1.6.2.2. Acceso a Datos (DAO)

Centraliza todo el acceso a datos en una capa independiente, aislándolo del resto de la aplicación, implementando un mecanismo de acceso requerido para trabajar con la fuente de datos; desacoplando la lógica de negocio de la lógica de acceso a datos. Sus principales beneficios son que reduce la complejidad de los objetos de negocio al abstraerlos de la implementación real de la comunicación con la fuente de datos, oculta completamente los detalles de implementación a sus clientes, además permite adaptarse a diferentes esquemas de almacenamiento. Cada DAO tiene que implementar los métodos declarados en la interfaz DAO correspondiente [26].

#### 1.6.2.3. Patrón de Presentación Vistas Compuestas

Gestiona los diferentes elementos de la vista por medio de una plantilla que hace la representación de las vistas más manejable. Tiene como propósito el crear Vistas Compuestas de varias sub-vistas de forma modular, flexible y extensible para construir vistas de páginas JSP para aplicaciones J2EE, que incluye otras páginas JSP y HTML usando la directiva *include*, utilizando componentes JavaBeans o también mediante etiquetas personalizadas [25].

#### 1.6.2.4. Patrones de asignación de responsabilidades (GRAPS)

**Patrón Experto:** Se asigna una responsabilidad al experto en la información, o sea, se definen las clases y sus funcionalidades atendiendo a la información concreta que deben manejar. De este modo se obtiene un diseño con mayor cohesión y la información se mantiene encapsulada (disminución del acoplamiento).

**Patrón Creador:** Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

**Patrón *Controller* (controlador):** Se asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas que tengan una posición intermediaria entre la interfaz de usuario y las clases donde reside la lógica de la aplicación, facilitando la centralización de actividades.

**Patrón Bajo Acoplamiento:** Propone mantener pocas dependencias entre las clases, logrando que el código sea más fácil de entender, mantener y reutilizar.

**Patrón Alta Cohesión:** Propone que cada elemento del diseño debe realizar una labor única dentro del sistema, logrando un alto grado de funcionalidad combinada con una reducida cantidad de operaciones, trayendo como ventaja que se simplifique el mantenimiento y que aumente la capacidad de reutilización [27].

### 1.6.3. Patrones de flujo de trabajo

Existen diferentes patrones definidos a la hora de modelar el negocio con notación BPMN, en la descripción de los diferentes procesos analizados se evidencian:

- ✓ **Secuencia:** El patrón de secuencia se define como una serie ordenada de actividades, con una actividad de partida después de que una actividad anterior se ha completado. Un diagrama de proceso de negocio define este patrón como una serie de actividades relacionadas con el Flujo de Secuencia. La dirección de las puntas de flecha determina el orden de la secuencia.

  Para este modelo, cuando una actividad se completa, una muestra viajará a través del flujo de secuencia a partir de esa actividad a la siguiente actividad de la secuencia. No hay ninguna condicionalidad o cualquier otro tipo de control establecido.

- ✓ **Elección Exclusiva:** Se define como un lugar en un proceso en el que el flujo se "divide" en dos o caminos alternativos más exclusivos. El patrón es exclusivo en la que sólo uno de los caminos alternativos pueden ser elegido para que el proceso continúe.

  Dado que se necesita un control adicional para crear el patrón Elección Exclusiva, BPMN utiliza una puerta de enlace. Específicamente, una compuerta exclusiva base de información se utiliza para este patrón. El saliente de secuencia de flujo de la puerta de enlace tendrá expresiones booleanas que serán evaluados para determinar la secuencia de flujo que debe utilizarse para continuar el proceso. Cuando una muestra llega a la puerta de enlace, se evalúan las expresiones (en un orden predeterminado) y para la primera expresión que se determina verdadera, se elegirá el flujo de secuencia correspondiente y la muestra continuará por ese camino [34].

### 1.7. Conclusiones parciales

En este capítulo se sentaron las bases teóricas que sustentan el proceso de desarrollo de la solución del problema planteado. Se realizó un análisis sobre la conciliación bancaria y sistemas existentes en el mundo, lo cual permitió enriquecer las funcionalidades para el mejoramiento del subsistema Conciliación en Quarxo. Además se vieron algunos conceptos relacionados con el subsistema de Contabilidad. Se caracterizaron las tecnologías, notaciones, técnicas, metodología, herramientas de desarrollo, así como los patrones que contribuyeron a la construcción del sistema informático para el cual se propone la solución.

# CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA, ANÁLISIS Y DISEŇO

## 2.1. Introducción

Este capítulo se incluye una caracterización de la arquitectura definida en el proyecto. También se aborda el modelado del negocio, teniendo los artefactos correspondientes al análisis a partir de las funcionalidades identificadas. Se especifican los diagramas que dan lugar al desarrollo del diseño, así como los artefactos del mismo.

## 2.2. Modelado del Negocio

Esta fase está destinada a comprender los procesos de negocio de la organización, para percibir cómo funciona el negocio que se desea automatizar y tener garantías de que el software desarrollado va a cumplir su propósito. El modelado de negocio se define como un proceso de representación de uno o más aspectos o elementos de un sistema, identificando las necesidades de cada área involucrada [28].

### 2.2.1. Descripción del proceso del negocio

Un proceso de negocio es una colección de actividades lógicamente relacionadas llevadas a cabo con un fin, identificar los procesos del negocio constituye el punto de partida de todo software. Para ver los restantes procesos ver Anexo 1.

Proceso: **Actualizar activo fijo**

| Nombre: Actualizar activo fijo |
|---|
| Resumen:<br><br>El subproceso es generado una vez detectado errores en un activo fijo ya registrado en el sistema. Actualmente dicho proceso no se realiza satisfactoriamente; pues lo que se hace es eliminar el activo fijo, para luego ingresarlo correctamente. Dando también problemas ya que la cancelación del activo fijo no lo elimina completamente del sistema quedando pendiente la contabilidad de los mismos. |
| Salida (s):<br><br>✓ Actualización de Activo Fijo de forma satisfactoria. |

| |
|---|
| ✓ Transacción contable de AF |
| Reglas del negocio:<br>✓ El activo fijo entrará a Quarxo luego de haberse hecho el cierre contable en el sistema Inventario.<br>✓ Se podrá actualizar el activo fijo una vez que se encuentre, por lo que tendrá que estar registrado y el estado del mismo será “Alta” |

**Tabla 1.** Descripción del proceso de negocio Actualizar activo fijo

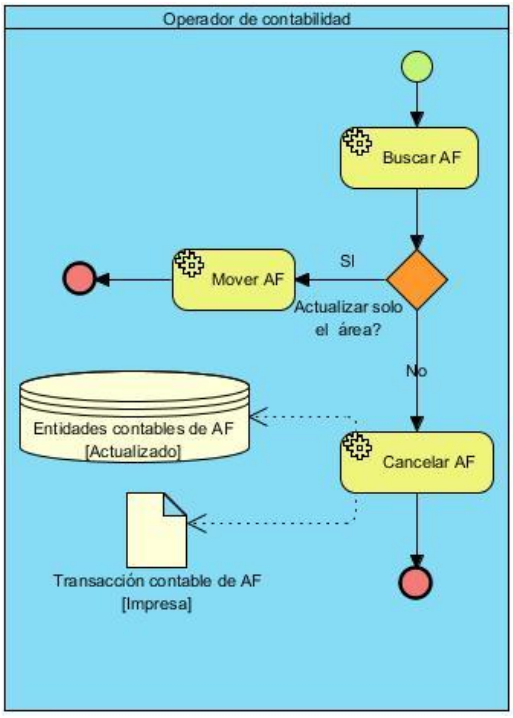


**Ilustración 1**. Diagrama del proceso: Actualizar activo fijo

### 2.2.2. Involucrados

Los involucrados son aquellos que participan en los procesos del negocio, ya sean trabajadores de la entidad financiera o cualquier persona natural o jurídica que interactúe con las acciones o procesos que se desarrollan dentro de la misma, como son:

| Involucrados | Descripción |
|---|---|
| Tesorero | Encargado de realizar la actualización diaria de la tasa de cambio, generándola desde el sistema Reuters[14] y cargándola en el sistema Quarxo |
| Operador de contabilidad | Encargado de operar con los activos fijos en el sistema Quarxo, siendo capaz de llevar a cabo toda la gestión de activos fijos tanto de la entidad misma como los del extranjero |
| Operador de BD | Encargado y único autorizado a trabajar directamente con la base de datos |
| Operador de conciliación | Encargado de realizar todo el proceso de conciliación a partir de los mensajes llegados por las diferentes vías contra el reporte de estado de “Nuestras Cuentas” que le hace llegar el Dpto. de Contabilidad, para de esta forma proceder con el desarrollo del proceso. |

**Tabla 2.** Involucrados en los diferentes procesos

### 2.2.3. Artefactos

Los artefactos no son más que los productos creados, modificados o eliminados a lo largo de los diferentes procesos; los cuales constituyen entradas y salidas importantes para la realización de los mismos, como son:

| Artefactos | Descripción |
|---|---|
| Doc. Tasa de Cambio | Tres ficheros con los datos de la tasa de cambio generada del sistema Reuters y cargar en el sistema Quarxo; pertenecientes al módulo de Tasa de Cambio |

[14]Reuters: Aplicación Web que se utiliza en la Dirección de Tesorería para cargar en tablas Excel los tipos de cambios

| | |
|---|---|
| Transacción contable de AF | Documento formal que contiene todos los datos de un activo fijo determinado, el mismo puede ser una salida del sistema o una entrada dependiendo del proceso; pertenecientes al módulo de Activo Fijo |
| Reporte de Entrada | Documento elaborado por una entidad bancaria, con el fin de recoger en este toda la información relacionada con los movimientos realizados en una cuenta desde la última conciliación |
| Txt | Documento con los reportes de las cuentas que llegan por mensajería; se genera automáticamente una vez entrado un número de cuenta en el sistema SISCOM[15] |
| Transacciones SWIFT | Mensaje del estado de cuenta que es enviado por una entidad bancaria, para dar a conocer al BNC el estado de cada una de sus cuentas en dicha; contiene todos los movimientos hechos por el BNC en dicha cuenta |
| Transacciones EC | Mensaje del estado de cuenta emitido por el departamento de conciliación con todos los movimientos hechos por el BNC en una cuenta determinada |
| Cierre Conciliación | Documento formal el cual contiene un resumen detallado del cierre de conciliación. |

**Tabla 3.** Artefactos generados por los diferentes procesos

## 2.3. Requisitos

Los requisitos funcionales definen el comportamiento interno del software, expresando la esencia del sistema. Definen las funciones que el sistema será capaz de realizar.

### 2.3.1 Técnicas para la captura de los requisitos funcionales

[15]SISCOM: Sistema informático que permite el procesamiento y enrutamiento de la mensajería SWIFT

La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. Las técnicas utilizadas fueron:

- ✓ **Análisis de soluciones existentes**: Esta técnica consiste en analizar diferentes sistemas ya desarrollados que posean características semejantes con el sistema a ser construido. Los diferentes sistemas estudiados en el caso del subsistema de Conciliación proponen funcionalidades las cuales fueron parte de la propuesta de solución: simplificar y agilizar el proceso de conciliación, conciliación automática, se permite conciliaciones manuales en forma automatizada y la carga de Información.

- ✓ **Entrevistas**: El uso de las entrevistas le permite al analista obtener conocimiento del problema y comprender los objetivos de la solución, mediante esta técnica el equipo de trabajo se acerca al problema de una forma natural. La técnica de la entrevista (no estructurada) realizada a los operadores de las diferentes áreas sirvieron para delimitar el alcance de los módulos describiendo los procesos a desarrollar, firmando en cada caso un acta de conformidad de la entrevista (ver Anexo 11)

### 2.3.2. Descripción de requisitos funcionales

Luego del análisis de los procesos y las entrevistas con los clientes se detectaron los siguientes requisitos funcionales:

| Requisitos | Descripción |
|---|---|
| RF.1 Imprimir Tasa de Cambio | Una vez actualizada la tasa de cambio se imprimirá automáticamente una confirmación de que el proceso se ha realizado correctamente. |
| RF.2 Actualizar activo fijo | Se permitirá la actualización de todos los datos del activo fijo, por posibles errores cometidos a la hora de registrarlos o por cambios hechos en alguno de ellos. |
| RF.2.1 Cancelar activo fijo | No solo eliminará un activo fijo existente con todos sus datos, |

| | |
|---|---|
| | sino que también lo elimina de las entidades contables, cancelando toda la contabilidad que el mismo posee. |
| RF.3 Registrar activo fijo del extranjero | Consiste en registrar un activo fijo con todos sus datos, existente en algunas de las sucursales que el BNC tiene en el extranjero. |
| RF.4 Buscar Partidas Pendientes | Consiste en buscar todas las partidas pendientes existentes, según un criterio de búsqueda. |
| RF.4.1 Actualizar Partida Pendiente | Permite renovar los datos de una partida pendiente ya existente en el sistema. |
| RF.4.2 Consultar Partida Pendiente | Da la posibilidad de ver una partida pendiente, sin poder realizarle cambios a la misma. |
| RF.5 Crear Cierre | Una vez que el operador realiza la conciliación, esta es guardada como un cierre. De quedarse a media la conciliación se ofrecerá la opción de guardarlo a medias. |
| RF.5.1 Buscar Cierre | Permite buscar todos los cierres existentes según el criterio de búsqueda entrado. |
| RF.5.2 Confirmar Cierre | El cierre ha de ser creado por el operador, pero tiene que ser confirmado por la persona a cargo del departamento de conciliación. |
| RF.5.3 Consultar Cierre | Muestra el cierre con todos sus datos, no da la posibilidad de realizar cambios en el mismo. |
| RF.5.4 Cargar Cierre | Permitirá cargar un cierre guardado para terminarlo de crear. |

**Tabla 4.** Requisitos funcionales

### 2.3.3. Requisitos no funcionales:

| **Tipo de requisito** |
|---|

| PC | Software | Hardware |
|---|---|---|
| **Clientes** | Máquina virtual de Java 6.0 u 20 o superior<br>Mozilla Firefox 3.6 o superior | Procesador Pentium IV o superior 2.0 GHZ o superior<br>RAM: 256 MB(recomendado 512 MB)<br>Una tarjeta de red |
| **Servidor 1** | Windows Server 2003<br>Apache Tomcat 6.0 o superior<br>Máquina virtual de Java 6.0 u 20 o superior | Procesador: Core 2 Duo 2.0 GHZ o superior<br>RAM: 4 GB<br>Disco duro: 160 GB<br>UPS: 1<br>Lector de CD: 1 |
| **Servidor 2** | Windows Server 2003<br>Microsoft SQL Server 2005 o superior | |

**Tabla 5.** Requisitos no funcionales

**Soporte**: Solo se necesita en un navegador para ejecutar la aplicación en las estaciones de trabajo, las actualizaciones solamente son en el servidor. La instalación del sistema consiste publicar la aplicación en un servidor.

**Seguridad**: La lógica de la aplicación reside en el servidor. El sistema concederá el acceso a partir de un usuario y una contraseña; teniendo acceso el usuario solo a las funcionalidades que se les definieron a partir de su área de trabajo.

**Usabilidad**: Peticiones asíncronas con la tecnología Ajax, componentes visuales de java script.

**Prestaciones en el cliente:** Los llamados Sistemas de cliente ligero, el sistema reside en el servidor por lo que las máquinas clientes no necesitan muchas prestaciones para ejecutar la aplicación.

**Interacción con dispositivos**: En la plataforma Java para interactuar con dispositivos desde la Web están los *applets*. Un componente Java que se ejecuta en el navegador y es capaz de interactuar con el cliente.

**Integridad en los datos**: Como la lógica de la aplicación reside en el servidor, todas las funcionalidades se pueden ejecutar transaccionalmente, asegurando la integridad en la información.

**Escalable y rendimiento**: Se pueden establecer Balanceo de cargas y *Clusters* del lado del servidor para soportar una gran cantidad de peticiones concurrente.

### 2.3.4. Priorización de requisitos funcionales

El Modelo de desarrollo del centro propone una tabla para la priorización de requisitos donde se plantean diferentes aspectos por los cuales los mismos son dados en Bajo, Medio o Alto:

**Criterios de Complejidad:** Para la determinación de la complejidad de los requisitos se analizan individualmente los criterios siguientes, llegando al resultado de si el requisito es de complejidad Alta, Media o Baja. La clasificación de la complejidad permite estimar el esfuerzo de implementación del requisito y contribuye a la decisión sobre la inclusión en las etapas de desarrollo del software.

**Complejidad por Interfaces:** El criterio interfaz se aplica a requisitos que presenten algún tipo de complejidad en su interacción con los siguientes elementos:

- ✓ Humanas (formularios, informes)
- ✓ Equipo (Tomógrafos, Rayos X); Ej *API, drivers*
- ✓ Programación (Programas externos necesarios para apoyar el producto); Ej. Adicionar un nuevo usuario, usa el LDAP)
- ✓ Comunicación (Protocolos de comunicación que serán utilizados); Ej. HL7

**Diferentes comportamientos:** Un mismo requisito se comporta de manera diferente ante determinadas situaciones; Ej. Admisión de un paciente, puede ser normal, o por emergencia, en el primer caso recoge más información que en el segundo.

**Formas de inicialización:** Un mismo requisito puede ser inicializado de diferentes formas, Ej. Ver detalles de una historia clínica, se inicializa cuando creas la historia clínica en una vista previa, y cuando se selecciona mostrar historia clínica.

**Consultas a fuentes de almacenamientos:** Los requisitos pueden presentar diversidad en la cantidad y complejidad de la interacción con la fuente de datos.

- ✓ Base de Datos
- ✓ Ficheros
- ✓ Otros

**Restricciones de validación:** Complejidad de todas las validaciones que lleve un requisito, tanto las validaciones en el lado del cliente, como en el servidor.

**Grado de reutilización:** Complejidad de un requisito, para poder ser reutilizado por otros.

**Lógica de negocio:** Los requisitos pueden presentar diferentes niveles de complejidad para la implementación de la lógica de negocio que contienen; Ej. Operaciones, métodos matemáticos, CRUD, etc.

| Resumen | |
|---|---|
| Cant. de requisitos con complejidad Alta | 3 |
| Cant. de requisitos con complejidad Media | 5 |
| Cant. de requisitos con complejidad Baja | 4 |

**Ilustración 2.** Evaluación de requisitos

### 2.3.5. Matriz de trazabilidad

Cuando el proyecto a desarrollar es muy grande o complejo es difícil poder saber que test ejecutados o diseñados cubren cada una de las especificaciones o requisitos del sistema. Es por este motivo que existe lo que se conoce como la matriz de trazabilidad; la cual es una herramienta que se utiliza para saber que requisitos quedan cubiertos por una prueba, la misma se desarrolla sobre el Visual Paradigm.

Para realizar la trazabilidad se definieron elementos como: requisitos, casos de prueba, modelo conceptual, reglas del negocio, descripción del proceso de negocio

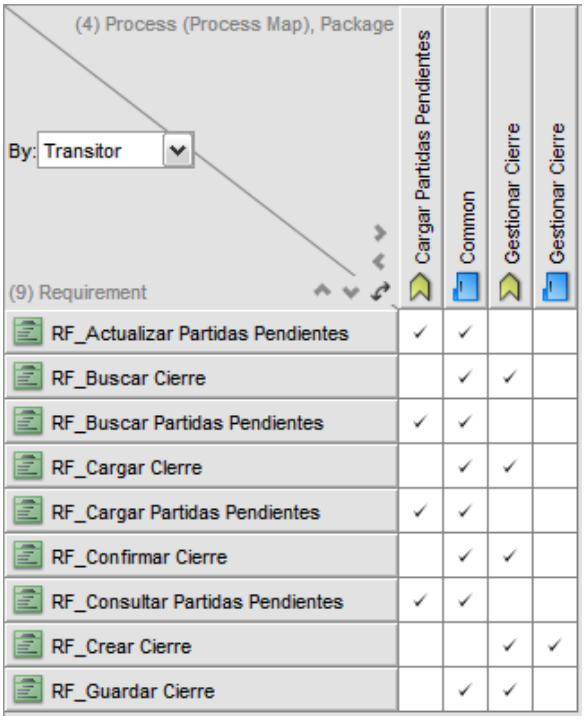


**Ilustración 3.** Diagrama de matriz de trazabilidad del subsistema Conciliación

### 2.3.6. Validación de los requisitos funcionales

La etapa de validación de los requisitos funcionales es considerada la actividad donde los clientes con ayuda de los desarrolladores, revisan los requisitos definidos para confirmar que realmente reflejan sus necesidades y que definen el producto deseado, dando solución a todos los problemas planteados. En el desarrollo de la ingeniería de requisitos aplicada al subsistema Conciliación Bancaria y los módulos Activos Fijo y Tasa de cambio del subsistema Contabilidad se utilizaron las técnicas de validación requisitos mediante prototipos y escenarios.

Las técnicas utilizadas para la validación de requisitos fueron:

- ✓ **Prototipos:** Se crearon interfaces de usuario, para una mejor visualización y aceptación por parte del cliente (ver Anexo 3)
- ✓ **Revisiones de requisitos:** Se firmo un Acta de conformidades de la entrevista (ver Anexo 11), donde el cliente daba por aprobado la descripción de cada uno de los requisitos.

- ✓ **Generación de caso de prueba:** Se generaron casos de prueba por cada uno de los requisitos funcionales (ver Anexo 8)

Los prototipos que se presentan a continuación validan las funcionalidades de Actualizar activo fijo

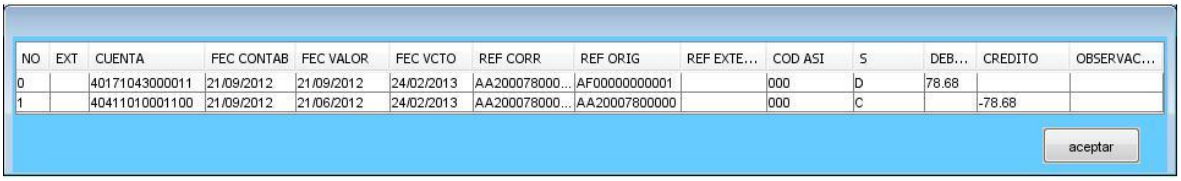


**Ilustración 4.** Interfaz de usuario: Transacción contable

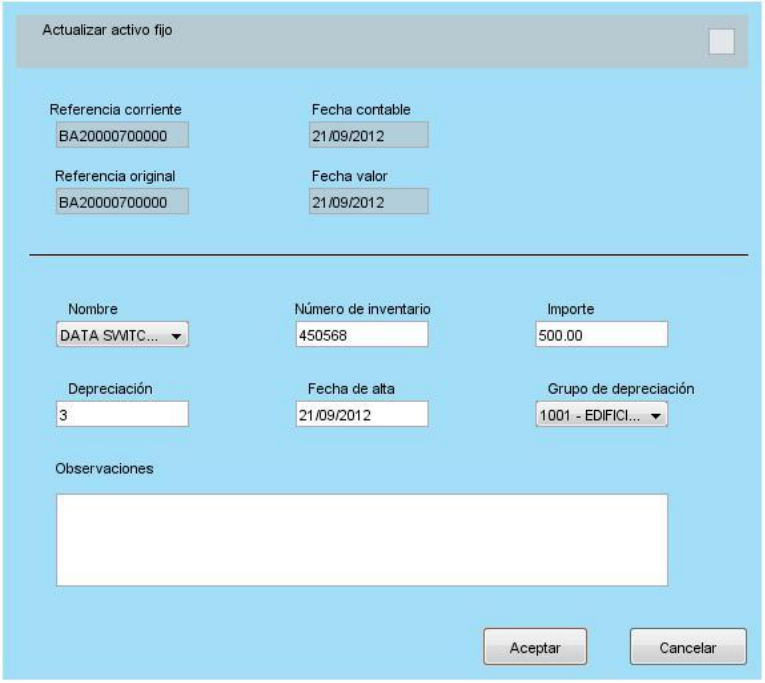


**Ilustración 5.** Interfaz de usuario: Actualizar activo fijo

## 2.4. Análisis y Diseño

Durante esta etapa es modelado el sistema para que soporte todos los requisitos antes aprobados. Esto da lugar a una arquitectura sólida y estable, convirtiéndose en un plano para la próxima fase.

### 2.4.1. Arquitectura de Quarxo

La arquitectura de software no es más que estructuras de un sistema, compuestas de elementos con propiedades visibles de forma externa y las relaciones que existen entre ellos [29].

La arquitectura de un sistema informático es el resultado de acoplar elementos arquitectónicos de forma apropiada para satisfacer los requisitos funcionales y no funcionales de un sistema. Una de las más utilizadas es la arquitectura de tres capas: presentación (interfaz de usuario), negocio y almacenamiento de información (acceso a datos).

Fue definido por los arquitectos del proyecto, la realización del sistema Quarxo utilizando dicha arquitectura; atendiendo al funcionamiento y complejidad de los procesos e incluyéndose además una capa transversal a las otras, conteniendo la misma los objetos del dominio. Lográndose flexibilidad y adaptabilidad en el sistema realizando una estructuración en subsistemas, componentes y módulos [30].

Se agruparon los Módulos y Componentes por Subsistema, cada Subsistema tiene uno o más Módulos y/o Componentes estrechamente relacionados con las funcionalidades que ejecutan. Los Módulos y/o Componentes estarán separados por diferentes capas lógicas según la naturaleza de los mismos:

- ✓ Capa de Presentación
  - Lógica de presentación
- ✓ Capa de Negocios
  - Fachada y Lógica de negocio
- ✓ Capa de Acceso a Datos
  - Desarrollo de *DAOs*. Clases para el acceso a la base de datos
- ✓ Capa de Dominio
  - Dominios o entidades persistentes

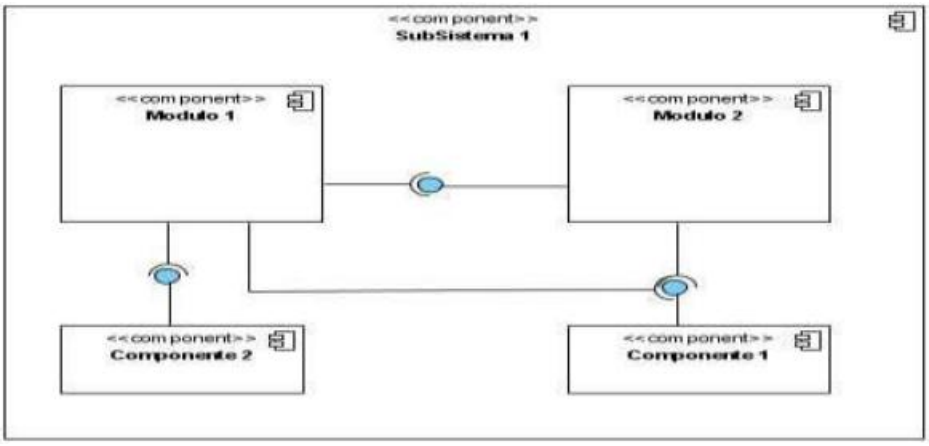


**Ilustración 6.** Arquitectura de un subsistema en Quarxo

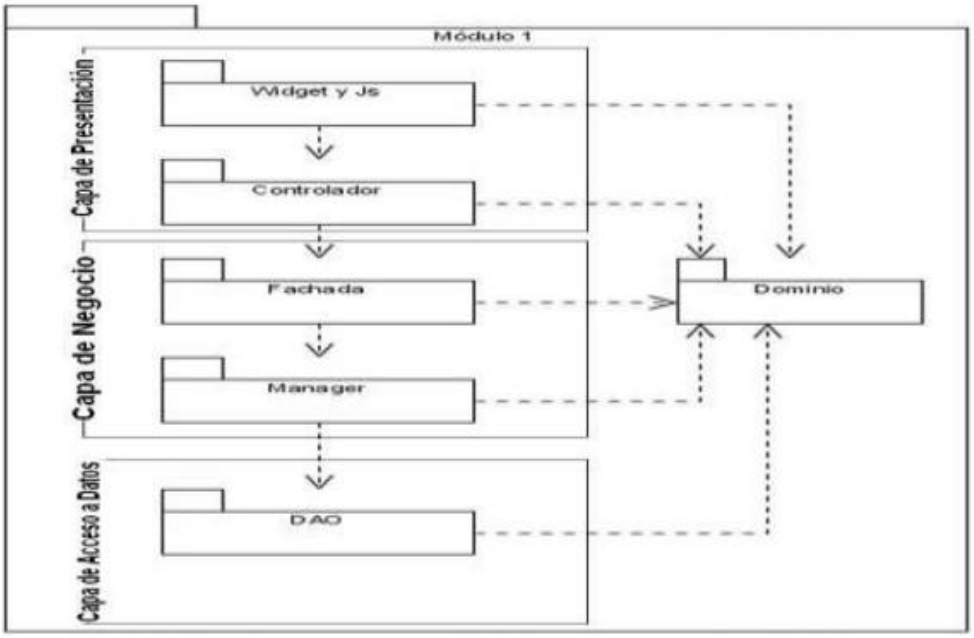


**Ilustración 7.** Arquitectura de un módulo en Quarxo

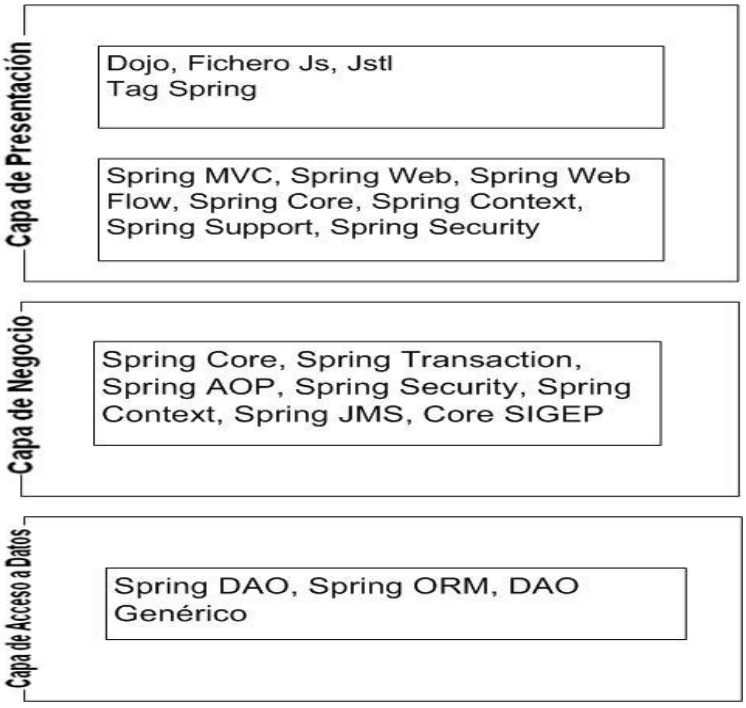


**Ilustración 8.** Relación entre capas lógicas y los frameworks

**Capa de Presentación**

Está dividida en dos partes: cliente y servidor. El servidor se encargada de recibir todos los pedidos de la interfaz de usuario, controlar el flujo de presentación del sistema y enviar las respuestas correspondientes a la interfaz de usuario; esta subcapa se relaciona con la capa de Negocios y de Dominio. El cliente se encargará del manejo de los eventos y las validaciones.

**Capa de Negocios**

Está divida en dos subcapas principales, Fachada y Manager. En la Fachada se expondrán todas las funcionalidades que la capa de presentación necesitará, encargándose de agrupar los métodos según su naturaleza. Por otra parte la subcapa Manager tiene la jerarquía de clases suficiente para implementar la lógica de negocio de la aplicación a desarrollar.

**Capa de Acceso a Datos**

En esta capa se implementarán los métodos encargados en interactuar con el gestor de Base de Datos. Permitiendo la conexión de la aplicación con la fuente de datos, así como el acceso y persistencia de la información; teniendo dependencia solo con la Capa de Dominio.

**Capa de Dominio**

En esta capa se están declaradas todas las clases que representan entidades del negocio, estas clases están presentes en todas las capas anteriormente descritas.

### 2.4.2. Modelo de datos

Un modelo de datos es una colección de conceptos bien definidos matemáticamente que ayudan a expresar las propiedades estáticas y dinámicas de una aplicación con un uso de datos intensivo. Sirve para describir la estructura de la base de datos, así como los datos, sus relaciones y las restricciones que deben cumplirse entre ellos [3] (ver Anexo 2).

Las bases de datos relacionales se normalizan con el fin de evitar la redundancia de los datos, los problemas de actualización de los datos en las tablas y proteger la integridad de los datos. Decir que una base de datos está normalizada es decir que todas las tablas de la base de datos también lo están. En el caso de la base de datos de SABIC utilizada por el sistema Quarxo, no se puede decir que está normalizada. Sin embargo las tablas de base de datos que dan solución a la problemática planteada si se encuentran en la tercera forma normal. Para que una tabla se encuentre en la tercera forma normal debe de cumplir antes la primera y segunda [35]:

Una tabla está en Primera Forma Normal si:

- ✓ Todos los atributos son atómicos. Un atributo es atómico si los elementos del dominio son indivisibles, mínimos
- ✓ La tabla contiene una llave primaria única y la misma no contiene atributos nulos
- ✓ No debe existir variación en el número de columnas
- ✓ Los Campos no llave deben identificarse por la llave (Dependencia Funcional)

- ✓ Debe Existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados
- ✓ Una tabla no puede tener múltiples valores en cada columna
- ✓ Los datos son atómicos (a cada valor de X le pertenece un valor de Z y viceversa)

Segunda Forma Normal (2FN):

- ✓ No exista dependencia funcional

Tercera Forma Normal (3FN):

- ✓ No existe ninguna dependencia funcional transitiva entre los atributos que no son clave.

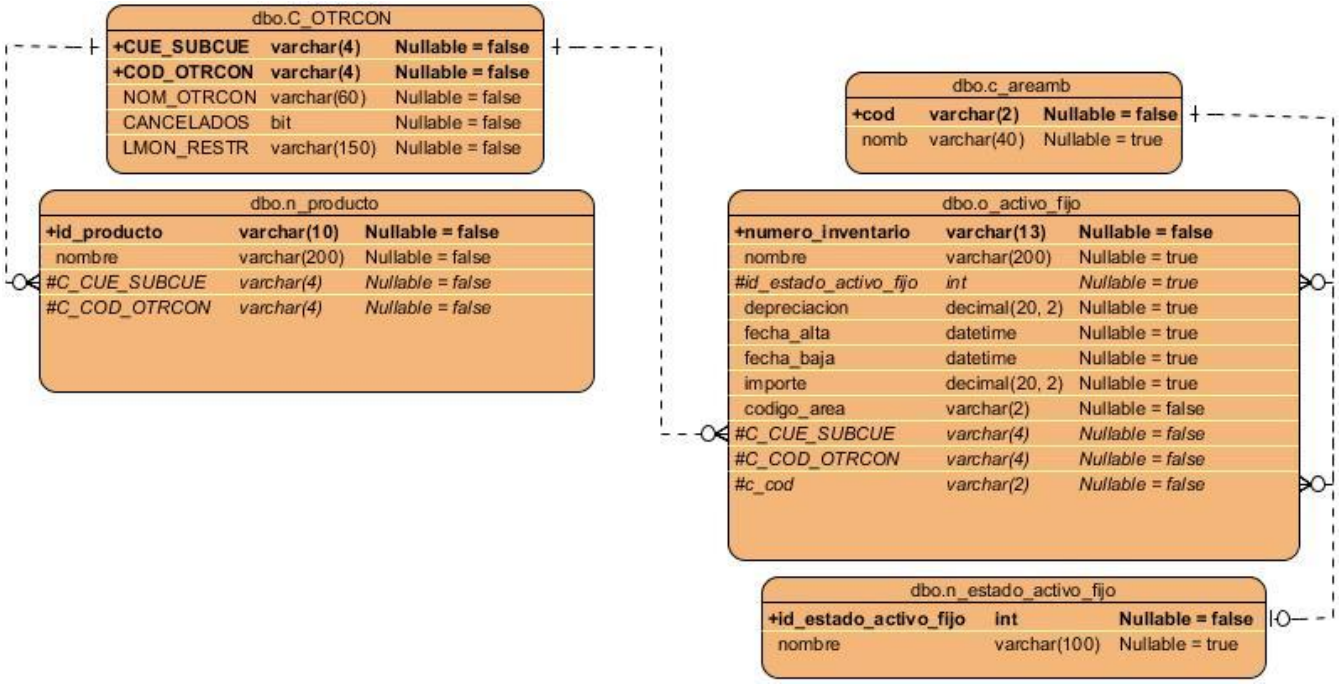


**Ilustración 9.** Modelo de Datos del módulo Activo Fijo

### 2.4.3. Diagrama de paquetes

Los paquetes permiten manejar los módulos en partes manejables mediante la agrupación de clases y otros paquetes, un diagrama de paquetes permite dividir el sistema organizándolo (ver Anexo 4).

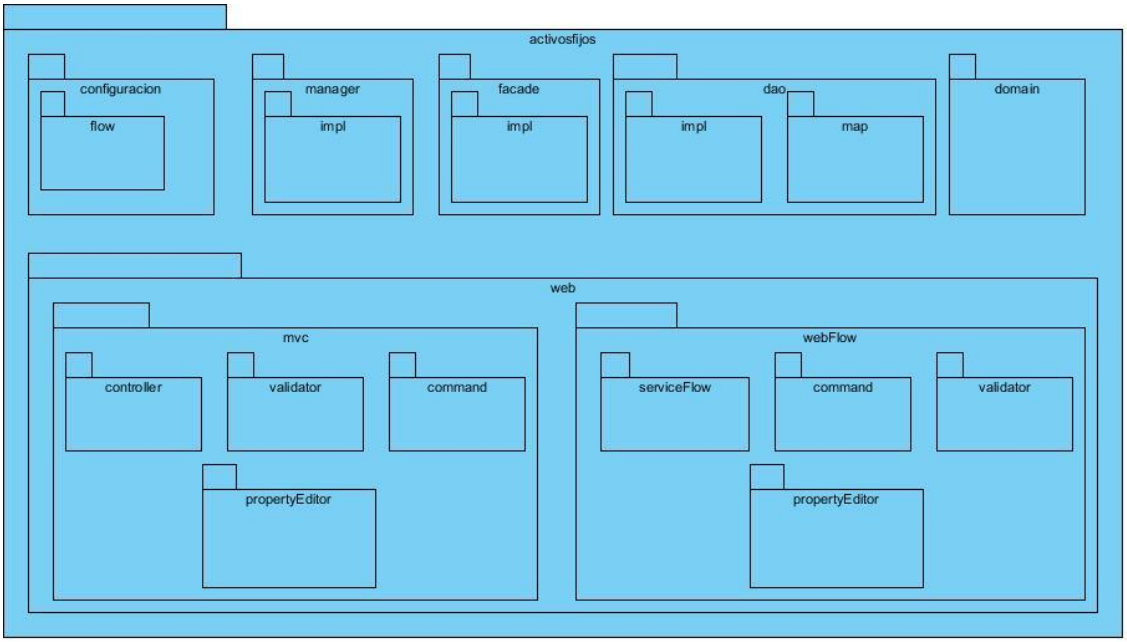


**Ilustración 10.** Módulo Activo Fijo

### 2.4.4. Diagrama de Clases del Diseño

El Diagrama de Clase es el diagrama principal para el análisis y diseño de un sistema. El mismo se elabora para tener en cuenta los detalles concretos de la implementación del sistema, describiendo gráficamente las especificaciones de las clases y de las interfaces de una aplicación; representa las definiciones de las entidades del software conteniendo de manera general las clases, asociaciones, métodos, dependencias y navegabilidad (ver Anexo 5).

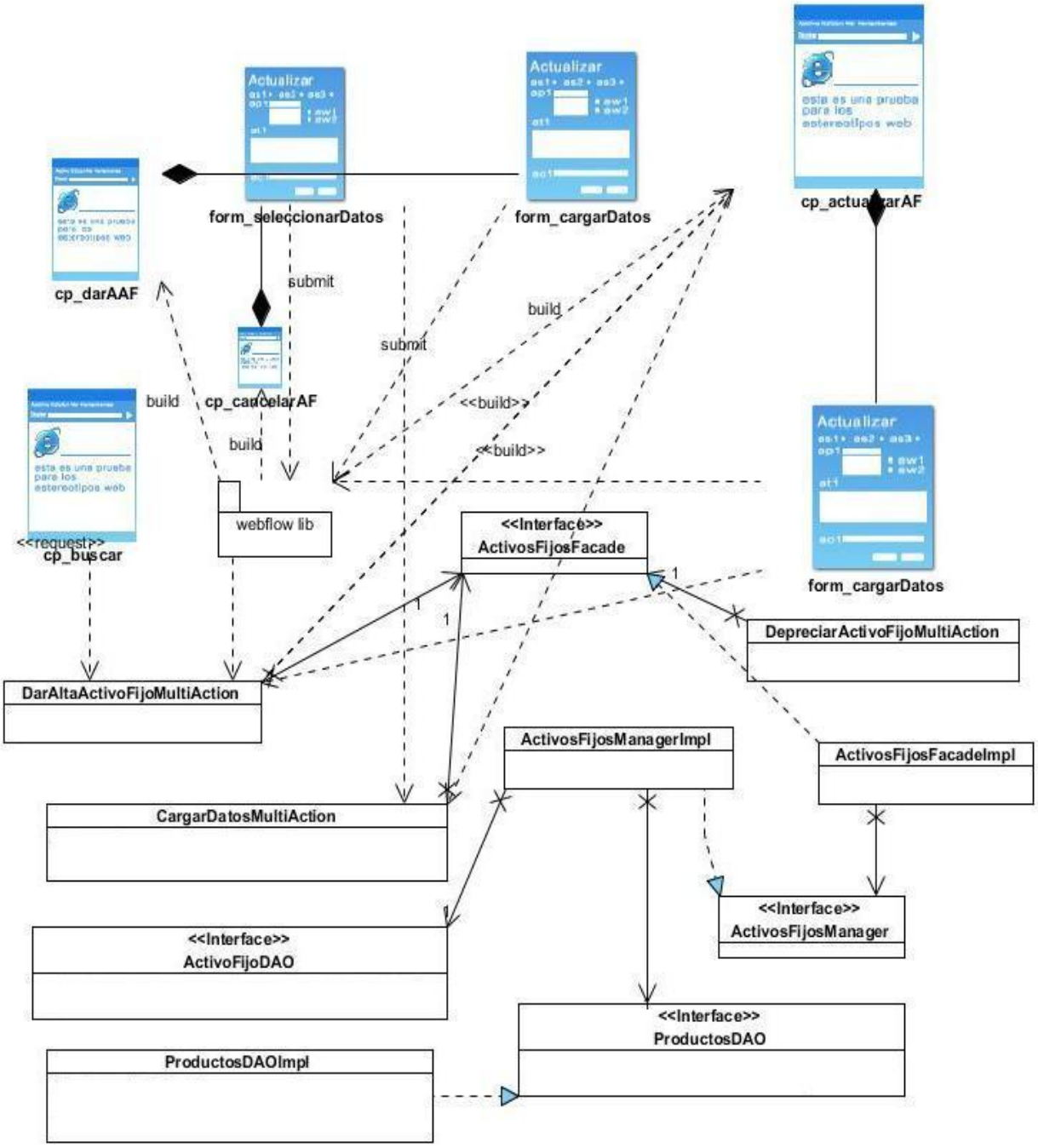


**Ilustración 11.** Actualizar activo fijo

| **CargarDatosMultiAction** | |
|---|---|
| **Atributos** | **Descripción** |
| activosFijosFacade :ActivosFijosFacade | -Fachada del activo fijo. |
| **Métodos** | **Descripción** |
| +listarAreas(request, response) : void | Listas las áreas. |
| +cargarCodigoGrupoDepreciar(request, response) : void | Carga el código del grupo a despreciar. |
| +fechaContableSistema(request, response) : void | Devuelve la fecha Contable del Sistema. |
| +listarNombres(request, response) : void | Lista los nombres de los activos fijos. |
| +cargarCodigosConceptoTasas(request, response) : void | Carga los códigos de los conceptos tasa de cambio. |
| +datosFacturaEntradaAlmacen(request, response) : void | Carga los datos de la factura de entrada al almacén. |
| +listarAsientosCarterasDarAlta(request ,response) : void | Lista los asientos de cartera para dar altas a los activos fijos. |
| +verificarActivoFijo(request, response) : void | Verifica los activos fijos. |
| DarAltaActivoFijoMultiAction | |
| **Atributos** | **Descripción** |
| activosFijosFacade :ActivosFijosFacade | -Fachada de activos fijos |
| **Métodos** | **Descripción** |
| +inicializarFlujo(context) : Event | Inicializa el flujo principal |
| +construirAsientos(context) : Event | Construye los asientos |
| +primeraValidacion(context) : Event | Realiza la primera validación de los datos |
| +segundaValidacion(context) : Event | Realiza la segunda validación de los datos |
| -formarMensajes(segundaV : String) : String | Crea el Mensaje |
| +contabilizar(context) : Event | Contabiliza |
| +validacionGeneral(context) : Event | Realiza una validación general de los |

| | |
|---|---|
| | datos |
| DepreciarActivoFijoMultiAction | |
| **Atributos** | **Descripción** |
| activosFijosFacade :ActivosFijosFacade | -Fachada de activos fijos |
| **Métodos** | **Descripción** |
| +DepreciarActivoFijoMultiAction() | Deprecia el activo fijo |
| +salvar(context) : Event | Guarda los cambios |
| +construirAsientos(context) : Event | Construye los asientos |
| +primeraValidacion(context) : Event | Realiza la primera validación de los datos |
| +segundaValidacion(context) : Event | Realiza la segunda validación de los datos |
| -formarMensajes(segundaV : String) : String | Crea el Mensaje |
| +contabilizar(context) : Event | Contabiliza |
| +vistaPrevia(context) : Event | Devuelve la vista previa |
| +imprimirAutomatico(context) : Event | Imprime el reporte |
| ActivosFijosManagerImpl | |
| **Atributos** | **Descripción** |
| -contabilizarFacade : ContabilizarFacade<br>-globalFacade : GlobalFacade<br>-nomencladorContabilizacionFacade : NomencladorContabilizacionFacade<br>-activoFijoDAO : ActivoFijoDAO<br>-productosDAO : ProductosDAO<br>-tasaActivoFijoDAOTasaActivoFijoDAO<br>-areaActivoFijoDAO : AreaActivoFijoDAO | -Fachada de contabilizar<br>-Fachada global<br>-Fachada de nomenclador<br>-Dao de activo fijo<br>-Dao de productos.<br>-Dao de tasa de activo fijo<br>-Dao de área de activo fijo |
| **Métodos** | **Descripción** |

| | |
|---|---|
| +construirAsientosEntradaAlmacen(refExternaFactura : String, refOriginalChequeCUC : String, refOriginalChequeCUP : String, refCorriente : String, fechaContable : java.util.Date, observacion : String) : Asiento | Construye los asientos de entradas al almacén. |
| +construirAsientosDarBajaActivoFijo(numeroInventario : String, fechaContable : java.util.Date, refCorriente : String, observacion : String, operador : String) : Asiento | Construye los asientos de dar baja al activo fijo. |
| -calcularDepreciaicon(tasa : TasaActivoFijo, importe : BigDecimal) : BigDecimal | Calcula la depreciación. |
| +contabilizar(asientos :.Asiento) : NError | Contabiliza |
| +primeraValidacion(asientos : Asiento) : int | Realiza la primera validación de los datos |
| +segundaValidacion(asientos : Asiento) : String | Realiza la segunda validación de los datos |
| -obtenerAsientoActivoFijo(refOriginal : String) : Asiento | Obtiene el asiento de activo fijo según la referencia original. |
| +listarProductos() : Producto | Lista los productos |
| +listarTasaDepreciacionActivoFijos(codGrupoDepreciar : String) : TasaActivoFijo | Lista la tasa de depreciación del activo fijo. |

**Tabla 6.** Descripción de las clases

### 2.4.5. Diagrama de Secuencia

El diagrama de secuencia modela la forma en que los objetos se comunican entre sí. Muestra los objetos participando en la iteración y la secuencia de mensajes intercambiados.

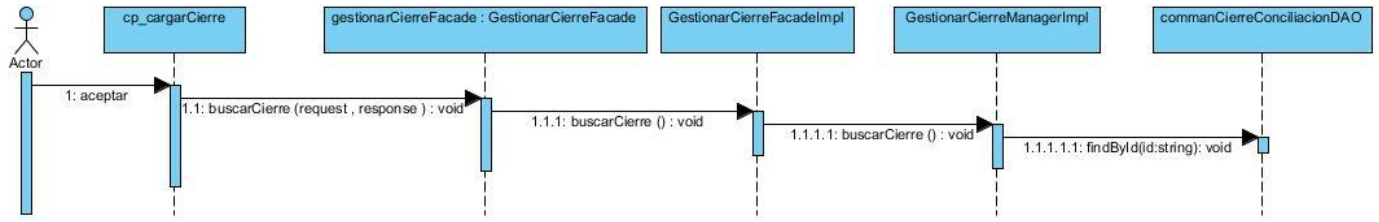


**Ilustración 12.** Cargar cierre

### 2.4.6. Patrones del diseño

A continuación se especifica el uso de patrones utilizado en la solución de la problemática:

- ✓ **Patrón de Acceso a Datos (DAO):** Se aplica con la definición de las interfaces DAO que disminuyen la complejidad de los objetos del dominio, al librarlos de la responsabilidad de manejar la implementación de sus fuentes de datos

- ✓ **Patrones GRASP**

  ✓ **Controlador:** Las clases controladoras de los módulos, constituyen un ejemplo de la aplicación de este patrón, las mismas tendrán la responsabilidad de escuchar y responder a las peticiones realizadas por la capa de presentación y de comunicarse con la capa de negocio. Por ejemplo en la interfaz **GestionarCierreMultiActionController**

  ✓ **Experto:** Este patrón fue utilizado, en el diseño, en la definición de las clases de acuerdo con las funcionalidades que deben realizar a partir de la información que manejan

  ✓ **Alta cohesión:** Este patrón se manifiesta en el diseño del componente de manera general, donde se agruparon las clases en dependencia de los requerimientos a los que se les debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional

  ✓ **Bajo acoplamiento:** Este patrón se manifiesta con la definición de interfaces e implementaciones, como puede ser la interfaz **GestionarCierreFacade** y su implementación **GestionarCierreFacadeImpl** permitiendo que clases como **GestionarCierreMultiActionController** se relacionen únicamente con ellas para realizar

sus operaciones, reduciendo el impacto de cambios posteriores en el negocio del sistema, logrando que el código sea más fácil de entender, mantener y reutilizar

- **Patrones GoF**

  ✓**Fachada:** La utilización de este patrón se evidencia en la definición de la interfaz **GestionarCierreFacade** responsable de la comunicación entre la capa de presentación y el grupo de clases e interfaces más complejas que se encargan de la lógica de negocio y el acceso a datos

- **Patrón Modelo Vista Controlador (MVC):** Se evidencia su aplicación en la arquitectura dividida en tres capas, utilizada en la solución. La capa de negocio, con la inclusión de las clases **GestionarCierreFacade** y **GestionarCierreManager,** y de acceso a datos, que contiene las interfaces **DAO** (ej: **PartidasMensajeriaDAO**) por cada funcionalidad y sus implementaciones

#### 2.4.6.1 Patrones de flujo de trabajo

En los diferentes procesos modelados con la utilización BPMN, se ven la utilización de los patrones empleados; tal es el caso del proceso Cargar tasa de cambio:

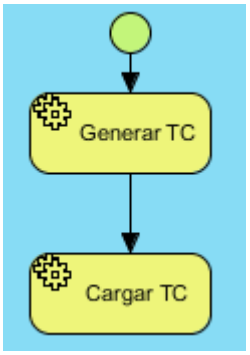


**Ilustración 13.** Patrón de Secuencia

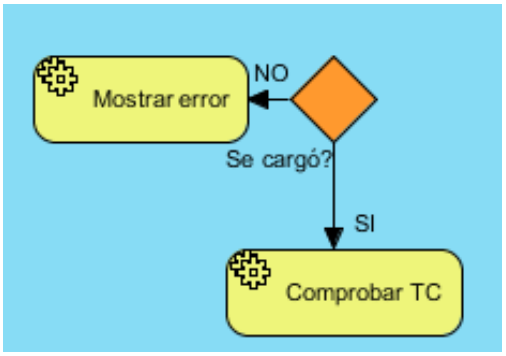


**Ilustración 14.** Patrón Elección Exclusiva

### 2.4.7. Validación del Diseño

Una métrica es un instrumento que permite evaluar el software al inicio del proceso, que cuantifica además un criterio y persigue comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel de proyecto. La aplicación de métricas al diseño de un producto de software constituye un elemento fundamental a la hora de evaluar la calidad del mismo.

Con el fin de desarrollar un diseño robusto y sencillo se realizó la validación del mismo utilizando las métricas Tamaño operacional de clase (TOC) y Relaciones entre clases (RC).

| nro | nombre de la clase | cant metodos |
|---|---|---|
| 1 | ActivosFijosManager | 56 |
| 2 | ActualizarActivoFijoMultiAction | 10 |
| 3 | CancelarActivoFijoMultiAction | 7 |
| 4 | DarAltaActivoFijoMultiAction | 9 |
| 5 | MoverActivoFijoMultiAction | 7 |
| 6 | ActivoFijoDAO | 6 |
| 7 | CargarDatosMultiAction | 32 |
| 8 | GestionarPartidasManager | 3 |
| 9 | ActualizarPartidaPendienteSimpleFormController | 5 |
| 10 | ConsultarPartidaPendienteAbstracComandController | 3 |
| 11 | PartidasPendientesMultiActionController | 6 |
| 12 | PartidaPendienteValidator | 2 |
| 13 | ConsultarCuentaConciliacionAbstracComandController | 2 |
| 14 | GestionarCuentaConciliacionMultiActionController | 10 |
| 15 | GestionarCierreMultiActionController | 25 |
| 16 | GestionarCierreManager | 19 |
| 17 | CommonManager | 12 |
| 18 | GestionarCuentaConciliacionManager | 11 |
| 19 | CierreConciliacionDAO | 1 |
| 20 | FechasCargaDAO | 1 |
| 21 | BuzonDAO | 1 |
| 22 | GruposDAO | 1 |
| 23 | PartidasECDAO | 1 |
| 24 | PartidasMensajeriaDAO | 1 |
| | Suma Total | 231 |
| | Promedio Total | 9,625 |

Activo Fijo

Conciliaciones

**Ilustración 15.** Clases y cantidad de métodos

| Atributos de calidad | Descripción |
|---|---|
| **Responsabilidad** | Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta |
| **Complejidad del mantenimiento** | Nivel de esfuerzo necesario para sustentar, mejorar o corregir el diseño de *software* propuesto. Puede influir significativamente en los costes y la planificación del proyecto |
| **Complejidad de implementación** | Grado de dificultad que tiene implementar un diseño de clases determinado. |
| **Reutilización** | Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de *software*. |
| **Acoplamiento** | Dependencia o interconexión de una clase o estructura de clase respecto a otra |
| **Cantidad de pruebas:** | Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto diseñado. |

**Tabla 7.** Atributos que posibilita medir las métricas TOC y RC

| Atributo | Categoría | Criterio |
|---|---|---|
| **Responsabilidad** | Baja | < =PO |
| | Media | Entre PO y 2* PO |
| | Alta | > 2* PO |
| **Complejidad implementación** | Baja | < =PO |
| | Media | Entre PO y 2* PO |
| | Alta | > 2* PO |
| **Reutilización** | Baja | > 2* PO |
| | Media | Entre PO y 2* PO |
| | Alta | <= PO |

**Tabla 8.** Atributos de calidad que afecta esta prueba

| Atributo de calidad | Modo en que lo afecta |
|---|---|
| Responsabilidad | Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase. |
| Complejidad de implementación | Un aumento del TOC implica un aumento de la complejidad de implementación de la clase. |
| Reutilización | Un aumento del TOC implica una disminución del grado de reutilización de la clase. |

**Tabla 9.** Modo en que se afectan los atributos de calidad

### 2.4.7.1. Métrica Tamaño Operacional de Clases (TOC)

Las métricas orientadas a tamaño para una clase Orientada a Objetos (OO) se centran en el cálculo de operaciones para una clase individual, y promedian los valores para el sistema OO en su totalidad. El tamaño general de una clase se determinó empleando la medida: número total de operaciones (tanto operadores heredadas como privadas de la instancia) que están encapsuladas dentro de la clase.

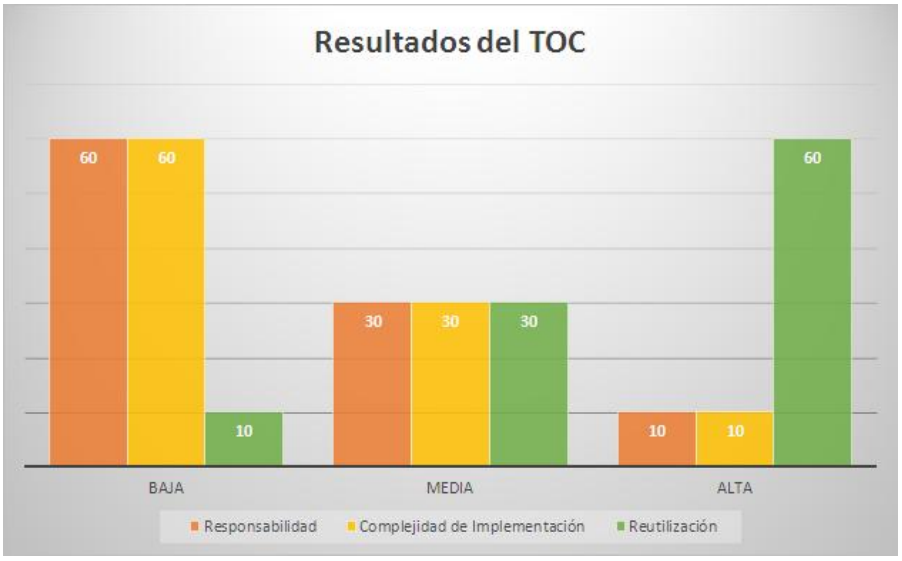


**Ilustración 16.** Resultados de la métrica TOC

### 2.4.7.2. Relaciones entre clases (RC)

Para la evaluación de las clases fueron utilizados umbrales para el acoplamiento, complejidad de mantenimiento, la reutilización y cantidad de pruebas.

| Atributos de calidad | Categoría | Criterio |
|---|---|---|
| **Acoplamiento** | Ninguno | 0 |
| | Bajo | 1 |
| | Medio | 2 |
| | Alto | >2 |
| **Complejidad de Mantenimiento** | Baja | <= PO |
| | Media | Entre PO y 2*PO |
| | Alta | > 2*PO |
| | Alta | > 2*PO |

| Atributos de calidad | Categoría | Criterio |
|---|---|---|
| **Reutilización** | Baja | >2* PO |
| | Media | Entre PO y 2*PO |
| | Alta | <= PO |
| **Cantidad de Pruebas** | Baja | <= PO |
| | Media | Entre PO y 2*PO |
| | Alta | > 2*PO |

**Tabla 10.** Umbrales utilizados

Esta métrica está dada por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad.

| **Atributo de calidad** | **Modo en que lo afecta** |
|---|---|
| **Acoplamiento** | Un aumento del RC implica un aumento del acoplamiento de la clase. |
| **Complejidad de mantenimiento** | Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase. |
| **Reutilización** | Un aumento del RC implica una disminución en el grado de reutilización de la clase. |
| **Cantidad de pruebas** | Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase. |

**Tabla 11.** Modo en que se afectan los atributos de calidad

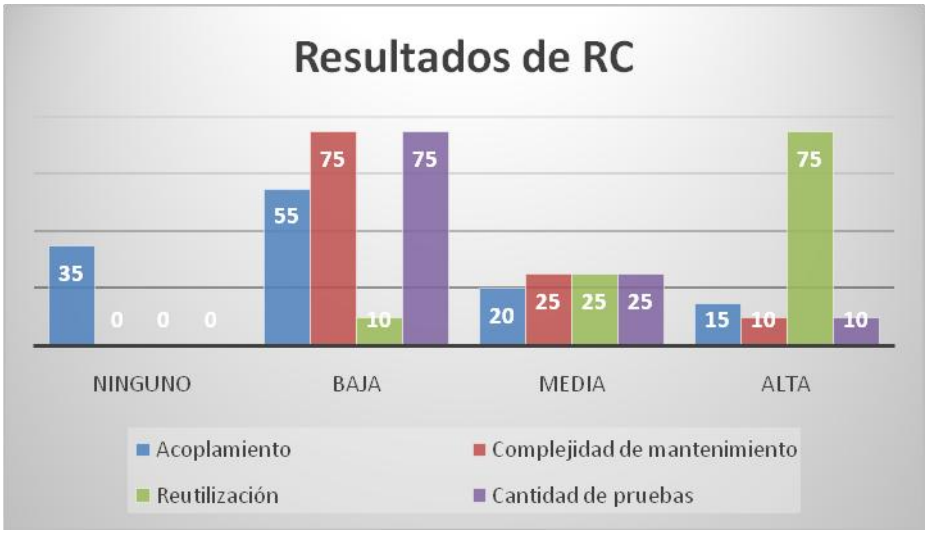


**Ilustración 17.** Resultados de la métrica RC

Luego de aplicarse la métrica de diseño RC y obtenidos los resultados de la evaluación del instrumento de medición de la métrica, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 75 % de las clases empleadas poseen menos de 3 dependencias de otras clases lo que lleva a evaluaciones positivas de los atributos de calidad involucrados (bajo acoplamiento con 55 %, baja complejidad de mantenimiento con un 75 %, baja cantidad de pruebas con 75 % y alta reutilización con un 75 %). Favoreciendo de esta manera la reutilización de las clases así como la modificación e implantación del diseño.

### 2.4.7.3. Matriz de inferencia de indicadores de calidad

La matriz de inferencia permite evaluar positiva o negativamente los resultados obtenidos de las relaciones atributo/métrica en una escala numérica, donde el valor 1 representa un resultado “positivo” y el valor 0 “negativo”. Si la métrica no influye en el atributo de calidad la relación es considerada como nula y es representada con un guion simple (-). Al promediar por cada atributo las relaciones no nulas, se obtiene un resultado general que al ubicarse en el rango de evaluación (valor <=0.4: “Mala”, valor >0.4 y valor<0.7: “Regular”, valor>=0.7: “Buena”) permite arribar a conclusiones sobre la calidad del diseño propuesto. Teniendo en cuenta que los resultados arrojados con la aplicación de las métricas fueron positivos para todos los atributos, la matriz de la inferencia queda elaborada de la siguiente manera:

| **Atributos\Métricas** | **TOC** | **RC** | **Promedio** |
|---|---|---|---|
| Responsabilidad | 1 | (-) | 1 |
| Complejidad de implementación | 1 | (-) | 1 |
| Reutilización | 1 | 1 | 1 |
| Acoplamiento | (-) | 1 | 1 |
| Complejidad de mantenimiento | (-) | 1 | 1 |
| Cantidad de prueba | (-) | 1 | 1 |

**Tabla 12.** Matriz de inferencia

El promedio general es 1 y teniendo en cuenta el rango de evaluación se concluye que el subsistema Conciliaciones y el Módulo Activo Fijo presentan un diseño con buena calidad.

### 2.5. Conclusiones Parciales

El conocimiento de los procesos del negocio permite el buen desenvolvimiento en el modelado de negocio que a su vez es el punto de partida para el reconocimiento de los procesos necesarios para la automatización del negocio, así como su mejor manejo. La obtención de los artefactos permitió una buena organización para el entendimiento de los diferentes procesos. La combinación de técnicas en la especificación y validación de los requisitos funcionales, permitió la comunicación fluida con el cliente hasta lograr un entendimiento común, lo que dio la posibilidad de recopilar información necesaria para el reconocimiento de las funcionalidades que se necesitan para el futuro sistema. Además permitió minimizar los problemas en la mala gestión de los requisitos funcionales. A su vez mediante la validación de los requisitos se logró verificar el cumplimiento de los objetivos trazados desde los inicios, desarrollando un buen análisis y diseño de los procesos a desarrollar.

# CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

## 3.1. Introducción

El último capítulo del presente trabajo está enfocado a darle cumplimiento a las disciplinas de implementación, pruebas internas y de liberación de la fase de Desarrollo según el Modelo de desarrollo del CEIGE. Se muestran el diagrama de componentes, al igual que los estándares de codificación, junto a las diferentes pruebas realizadas a la aplicación.

## 3.2 Implementación

Tomando como punto de partida los resultados arrojados por el análisis y diseño, el objetivo de la implementación es crear la arquitectura y el sistema como un todo. La importancia de esta disciplina se debe a que se obtiene un sistema ejecutable, siendo esto uno de los principales objetivos en el desarrollo de software.

### 3.2.1. Estándares de codificación

Los estándares de codificación se definen para lograr estandarización en la programación del software, facilitando la calidad del desarrollo del software. Estos se basan en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. La generalización de aspectos tan simples como el trato de las mayúsculas, ayuda a eliminar conflictos de funcionalidades implementadas con nombres iguales y guían de forma clara el proceso de desarrollo [29].

- ✓ **PascalCasing**: Se utiliza en el nombramiento de las clases, definiendo que deben comenzar con letra mayúscula, en caso de estar compuesta por varias palabras todas deberán comenzar con mayúscula
- ✓ **CamelCasing**: Los métodos y variables que se encuentran en las diferentes clases deben comenzar con minúscula la primera palabra, las restantes con mayúscula

### 3.2.2. Convenciones de nomenclatura

A partir de las notaciones definidas se establece una nomenclatura para la codificación de las clases del sistema de gestión bancario Quarxo, según la capa de la arquitectura a la que pertenezcan dichas clases; facilitando su ubicación y mantenimiento del código.

- **Capa de Presentación**

Las clases que pertenecen a los sub-paquetes mvc y webflow del paquete web tienen la siguiente nomenclatura:

Paquete *controller*:

- ✓ [Nombre de la funcionalidad a la que responde] + [Nombre del controlador de Spring MVC que se hereda]
- ✓ Ejemplo: **GestionarCierreMultiActionController**

Las clases contenidas en el resto de los sub-paquetes tienen la siguiente nomenclatura:

- ✓ [Nombre de la funcionalidad a la que responde] + [Nombre del paquete]
- ✓ Ejemplo: **CargarDatosMultiAction**

- **Capa de Negocio**

A las clases contenidas en los paquetes manager (gerente) y facade y sus sub-paquetes impl se les define la siguiente nomenclatura:

Para las interfaces:

- ✓ [Nombre del módulo] + [Nombre del paquete]
- ✓ Ejemplo: **ActivosFijosManager**

Para sub-paquetes impl:

- ✓ [Nombre del módulo] + [Nombre del paquete] + Impl
- ✓ Ejemplo: **GestionarCierreManagerImpl**

- **Capa de Acceso a Datos**

Las clases que están contenidas por el paquete *dao* y su sub-paquete *impl* presentan la siguiente nomenclatura:

Paquete *dao*:

- ✓ [Nombre de la entidad] + DAO
- ✓ Ejemplo: **ActivoFijoDAO**

Sub-paquete *impl*:

- ✓ [Nombre de la entidad] + DAOImpl
- ✓ Ejemplo: **PartidasECDAOImpl**

### 3.2.3. Diagrama de Componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Permite además visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Los componentes representan todos los tipos de elementos del software que entran en la fabricación de aplicaciones informáticas [30].

El Diagrama de componentes que se muestra a continuación se ha elaborado de forma tal que muestre las relaciones existentes entre el subsistema Conciliación y el resto de componentes dentro del sistema.

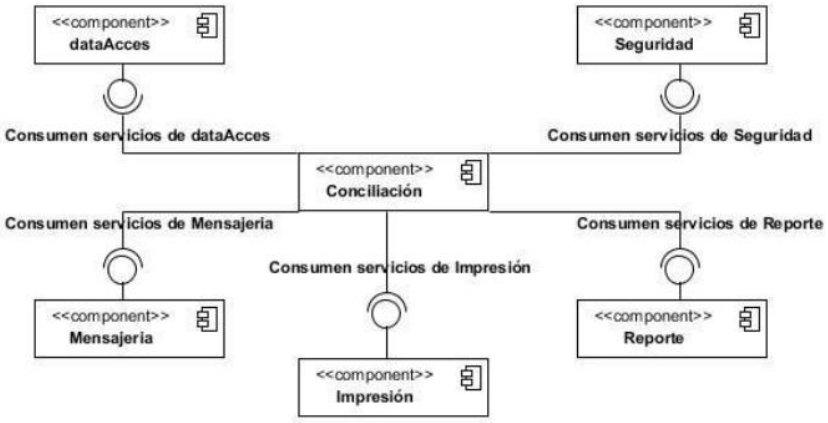


**Ilustración 18.** Modelo de componentes

A continuación se explican de manera general cada uno de los componentes relacionados en el modelo de componentes

***dataAccess*:** Este componente tiene la función de ofrecer todos los elementos necesarios para llevar a cabo la conexión a la base de datos, por lo que todos los subsistemas dependen de él para poder realizar las funcionalidades que realizan.

**Seguridad:** es el componente encargado de mantener la seguridad en todo el sistema a través de la comprobación de las peticiones y sus permisos en dependencia del usuario que la realice, de aquí que sea necesaria la comunicación con el mismo para garantizar la seguridad del subsistema Conciliación.

**Mensajería:** El subsistema Conciliación realiza operaciones que por su naturaleza requieren de la notificación mediante mensajería hacia y de los bancos implicados en las operaciones bancarias por lo que utilizan este componente que es el encargado de conceder las clases necesarias para el envío y recepción de mensajes SWIFT.

**Reporte:** Este componente tiene la función de ofrecer todos los elementos necesarios para llevar a cabo la ejecución de reportes en el cual se recibe toda la información necesaria de los estados de cuentas.

**Impresión:** Este es un componente que tiene como función brindar las clases y funcionalidades necesarias mediante las cuales es posible imprimir determinada información y de forma automática las operaciones que se realicen.

### 3.2.4. Diagrama de Despliegue

Los Diagramas de Despliegue muestran las relaciones físicas de los distintos nodos[16] que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo: dispositivos, procesadores, memorias [31].

[16]Nodos: Elementos físicos que existen en tiempo de ejecución y representan un recurso computacional, que generalmente tiene alguna memoria y capacidad de procesamiento

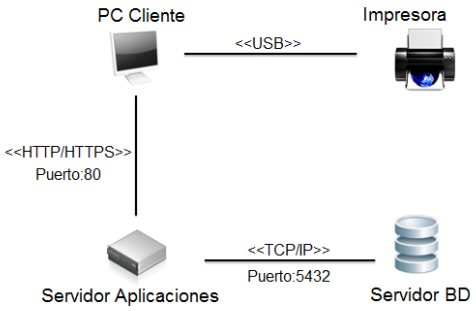


**Ilustración 19.**Diagrama de despliegue del sistema Quarxo

## 3.3 Pruebas

Conjunto de técnicas experimentales para la búsqueda de fallos en los programas, que determinan en cierto grado la calidad de un producto [32].

Se aplicaron los métodos de prueba de Caja Blanca para analizar la lógica interna del programa y Caja Negra con el objetivo de verificar que las funciones son operativas a través de la interfaz del software, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, manteniendo la integridad de la información externa.

### 3.3.1 Pruebas Internas

Durante esta fase se desarrollan las pruebas del grupo de calidad del centro verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple [32].

#### 3.3.1.1 Pruebas de Caja Blanca

Para las pruebas de Caja Blanca se utilizó el framework JUnit, el cual brinda un conjunto de librerías que se integran fácilmente al IDE de desarrollo seleccionado: Eclipse. JUnit permite la realización de las pruebas a los métodos de las clases implementadas (ver Anexo 10). Para hacer uso de este framework se definieron los siguientes pasos:

- ✓ Crear un caso de prueba por cada clase implementada
- ✓ Configurar en el caso de prueba, los ficheros que permiten comunicar las clases de las capas de presentación y lógica de negocio

- ✓ Definir los métodos a probar dentro de cada caso de prueba, incluyendo los parámetros de entrada
- ✓ Realizar pruebas a los métodos [33]

A continuación se muestran las imágenes que corresponden a la realización de las pruebas de Caja Blanca aplicando el framework *JUnit* al método: *crearCierre,* el cual se localiza en la clase controladora *GestionarCierreMultiActionController* dentro del subsistema Conciliaciones.

Primeramente se crea un objeto de la clase donde se encuentra el método a probar y a continuación se da paso a crear tres *mocks*, debido a que el método recibe como parámetros dos objetos de tipo *HttpServlet(request y response)*. Esto lo posibilita la librería *EasyMock* que trabaja en la creación de un *proxy* dinámico para dicho simulacro, el cual es controlado a través de un objeto de tipo *MockControl.*

```
public class GestionarCierreMultiActionControllerTest extends TestCase {

    GestionarCierreMultiActionController gestionarCierreMultiActionController;

    private GestionarCierreMultiActionController cierreMultiAcction;

    private MockControl controlHttpServlet;
    private HttpServletRequest mockhttpServletRequest;

    private MockControl controlHttpResponse;
    private HttpServletResponse mockhttpServletResponse;

    private MockControl controlHttpSession;
    private HttpSession mockHttpSession;
```

**Ilustración 20.** Declaración de las variables para la prueba

En el método *setUp,* de la propia clase, se inicializan las variables antes de cada prueba. Debido a la arquitectura del subsistema se debe crear un objeto por cada capa que haga énfasis en el método a probar. Estos pasos se van a configurar en el fichero*reporte-context.xml,* el cual va a ser el encargado de declarar y mapear los objetos creados por cada nivel de la aplicación.

Se inicializa cada *mock* creado para simular los objetos del método en cuestión. En este caso son tres*mock*: *mockHttpServletRequest, mockHttpServletResponse y mockHttpSession*.

```
protected void tearDown()
{
controlHttpServletRequest.verify();
}
```

**Ilustración 21.** Verificación de cada prueba.

El método *tearDown* es el encargado de las tareas a realizar en cada test. En este caso solo va a verificar los objetos que son pasados por el *request*.

Durante la realización de la prueba al método, se preparan los parámetros de pruebas necesarios al método *crearCierre*. Dichos parámetros serán pasados por el *mockHttpServletRequest*(ver ilustración 18). Posteriormente se realiza la condición de prueba a través del método *assertTrue* el cual devuelve verdadero si se emitió el reporte de manera satisfactoria.

```
public void crearCierreTest() throws Exception{
    mockhttpServletRequest.getParameter("id");
    controlHttpServlet.setReturnValue("06");

    mockhttpServletRequest.getParameter("parametros");

    Map map=new HashMap<String, String>();
    List<Map> listMap=(List<Map>)new LinkedList<Map>();

    controlHttpServlet.setReturnValue(listMap);
    controlHttpServlet.replay();

    assertTrue(cierreMultiAcction.crearCierre(mockhttpServletRequest,mockhttpServletResponse ));
}
```

**Ilustración 22.** Parámetros utilizados por el método a probar

El resultado de la prueba realizada a todos los métodos de la clase *GestionarCierreMultiActionController* fue satisfactorio.

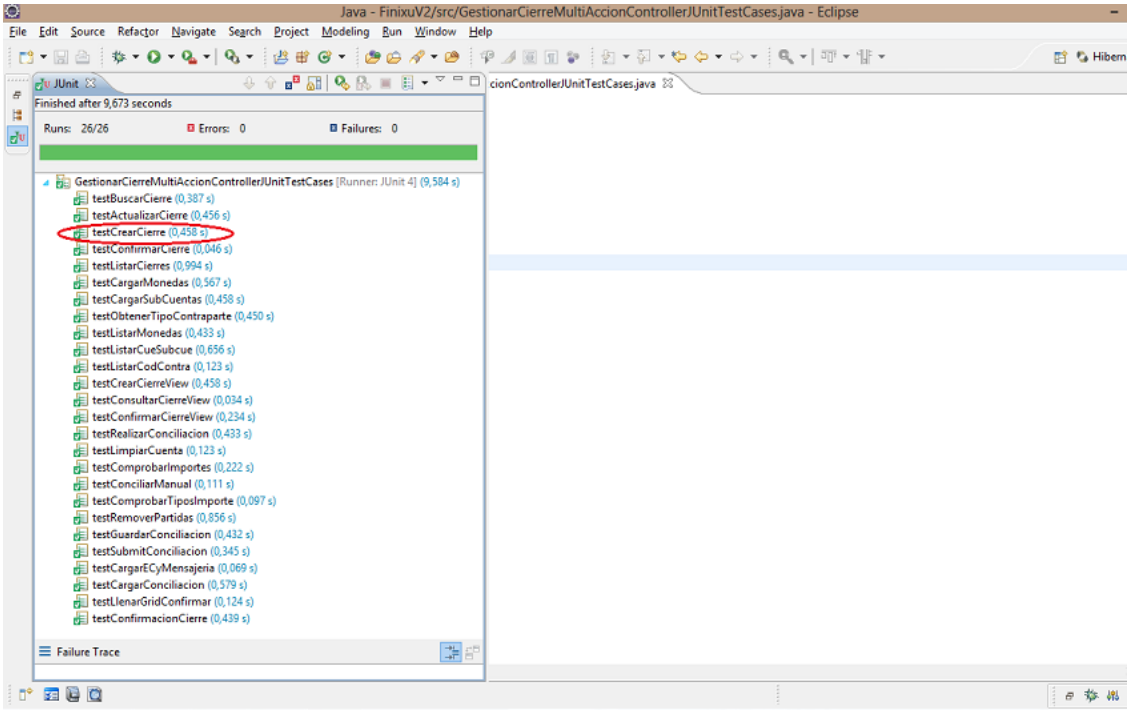


**Ilustración 23.** Resultado de la prueba realizada

### 3.3.2 Pruebas de Liberación

Se aplican pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación [12].

En esta disciplina de la metodología de desarrollo utilizada, se realizan actividades como la entrega de todos los artefactos, así como la solicitud de pruebas internas donde se presenta una plantilla la cual muestra los tipos de datos y juego de datos a entrar en el sistema, mostrando el resultado del mismo (ver Anexo 8). Al realizar las pruebas exploratorias se ven las no conformidades encontradas (ver Anexo 9), se emitieron tres iteraciones; arreglándose en cada caso los errores encontrados. Una vez arregladas todas las no conformidades se emite el acta de confirmación (ver Anexo 6).

#### 3.3.2.1 Pruebas de Caja Negra

Este tipo de prueba se centra en lo que se espera del software, es decir, los casos de prueba pretenden demostrar que las funciones del sistema son operativas, que los valores de entradas se aceptan

adecuadamente y que se produce una salida correcta, sin preocuparse de lo que pueda estar haciendo el software internamente, es decir, todas las pruebas se realizan sobre la interfaz de usuario [5].

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de Caja Blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de Caja Blanca.

Con este tipo de pruebas se intenta encontrar:

- ✓ Funciones incorrectas o ausentes
- ✓ Errores de interfaz
- ✓ Errores en estructuras de datos o en accesos a las bases de datos externas
- ✓ Errores de rendimiento
- ✓ Errores de inicialización y terminación

Las pruebas que se hacen en el departamento de calidad son las de partición de equivalencia. La partición equivalente es un método de prueba de caja negra, que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores, que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico.

Para la evaluación del componente X se realizaron X iteraciones, identificando un total de X No Conformidades las cuales fueron resueltas satisfactoriamente. Se realizaron pruebas exploratorias, funcionales y de regresión que validaron el correcto funcionamiento del componente, lo cual queda avalado por el Acta de Liberación emitida por el profesional responsable de las pruebas en el departamento.

El subsistema de Conciliaciones Bancarias, así como los módulos pertenecientes a la Contabilidad una vez probados y liberados por parte del grupo de calidad de CEIGE, se encuentra listo para entrar a las Pruebas de Integración y Aceptación, esta última debe llevarse a cabo en el BNC, donde deberá ser aprobado por parte del cliente. De manera general se espera que dicho subsistema se encuentre en

completa capacidad para su explotación como parte del sistema Quarxo una vez concluido todos los procesos de pruebas.

### 3.3.3.Validación de las variables de la investigación

El perfeccionamiento de las operaciones con el desarrollo de la solución, lo determina la agilidad y facilidad con que puedan ser realizados los procesos de activo fijo, conciliación bancaria, así como la confirmación de la tasa de cambio en el sistema Quarxo, ocasionando una reducción de tiempo y complejidad en la ejecución. A continuación la solución se somete a pruebas para comprobar que en efecto si se agilizan y facilitan las tareas antes mencionados.

Se toma como caso de estudio el proceso de actualizar activo fijo. La realización de esta funcionalidad en Quarxo v1 no era posible, por lo que se debía eliminar completamente el activo fijo y luego entrarlo correctamente, haciéndose la operación engorrosa para el usuario; además que para eliminar un activo fijo debía hacerse por el sistema y luego por la general para que quedara cancelado completamente de la contabilidad. Mediante las nuevas funcionalidades implementadas, el usuario puede actualizar el activo fijo sin necesidad de eliminarlo, pero en caso de que desee hacerlo ya el sistema permite cancelarlo correctamente, proporcionando rapidez y agilidad en el proceso y evitando la ocurrencia de errores.

## 3.4. Conclusiones Parciales

Como resultado de la disciplina de implementación se realizan los diagramas de componente y despliegue, dando una versión de las relaciones existentes entre los módulos perteneciente a la solución con relación a los componentes del sistema Quarxo. Además se realizan diferentes pruebas para la solución planteada, así como la validación de las variables de la investigación.

## CONCLUSIONES GENERALES

La investigación desarrollada y los resultados obtenidos permiten plantear las siguientes conclusiones:

- ✓ En el estudio de los sistemas que realizan la conciliación bancaria, se descartó la utilización de todos ellos por no corresponderse con la solución que se necesita
- ✓ El proceso de desarrollo de software fue guiado por el Modelo de Desarrollo de CEIGE, cumpliendo con los elementos que propone y logrando una efectiva ingeniería de software.
- ✓ Las tecnologías seleccionadas para el desarrollo de la solución están acordes a los requerimientos del cliente, a las políticas del centro y a los estándares internacionales
- ✓ Se obtuvo el subsistema Conciliaciones y los módulos de Activo Fijo y Tasa de Cambio del sistema Quarxo, los cuales se probaron mediante pruebas de Caja Blanca y Caja Negra, los mismos fueron validados utilizando pruebas de aceptación. Fue desarrollado con tecnologías libres, y provisto de elementos de seguridad, estándares y un entorno amigable, gracias a los framework utilizados

Por lo antes expuesto, se considera que la base teórica, la selección de las tecnologías para construir los diferentes módulos y su implementación, son los principales aportes de este trabajo de diploma. La solución desarrollada contribuye al sistema de gestión bancaria para el BNC, permitiendo disminuir los tiempos de respuesta y ahorrar los recursos materiales de la institución.

**RECOMENDACIONES**

Continuar los estudios del tema con el objetivo de encontrar nuevas funcionalidades para futuras versiones del sistema.

Adaptar el subsistema obtenido para que pueda ser utilizado en un futuro por otras entidades bancarias que incluyan la conciliación bancaria.

**BIBLIOGRAFÍA**


1. CORTÉS., J. Contabilidad General. Barcelona, Editorial: Mentensó., 1932. 2 p.
2. **Maldonado**. Estudio de la contabilidad general. http://bibliodoc.uci.cu/pdf/reg02900.pdf
3. **Rodríguez, Lic. Romelia.** Conciliación Bancaria. Guayana (Bolivar): s.n., 2007
4. **Edgar Naranjo Fuentes, Yelena Reyes Hidalgo.** Trabajo de Diploma: Análisis y diseño del subsistema de Conciliaciones Bancarias de QUARXO. 2011.
5. **Yakelín Parra Batista, Leonardo Vázquez Arzuaga.** Trabajo de Diploma: Desarrollo del subsistema Conciliaciones Bancarias del sistema QUARXO. 2012.
6. **SIC solucions**.SIC solucions. [Online] SICSA, 1982. [Cited: Junio 1, 2011.] http://www.sicsa.com/
7. **Sistema Isis Classic**.www.sistemaisis.com/software-de-gestion-de-bancos.htm
8. **Winledger**.www.winledger.com/conciliacion.php
9. **Sicob**.www.sicob.com.mx
10. **Conciliación Bancaria** automatizada.Conciliación Bancaria automatizada. [Online] Sol rojo Sistemas, S.A., 2003. [Cited: Junio 1, 2011.] http://www.conciliacionexpress.com/index.php
11. **Portal SIAFI**. Porta SIAFI. [Online] [Cited: Junio 1, 2011.] http://chorti.sefin.gob.hn/siafi/beneficiarios.html
12. **William González Obregón.** CEIGE-Modelo de desarrollo de Software v1.0. [Octubre 29,2012]
13. **Camy, Lázaro Issi.** JavaScript. s.l. : Anaya Multimedia, 2002
14. **Visual Paradigm.** [En línea] [Citado el: 15 de marzo de 2011.] http://www.visual-paradigm.com.
15. **JesusM.Gonzalez, JoaquinSeoane.** Introducción al software libre. http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html

16. **Microsoft SQL Server 2005**. [En línea] [Citado el: 5 de abril de 2011.] http://www.microsoft.com/sqlserver/2005/en/us/product-information.aspx

17. **Pablo Castells**[Marzo 6, 2012] www.eps.uam.es/~castells/docencia/poo/2-java-esp.pdf

18. **Harmon, James E.** Using the Dojo JavaScript Library to Build Ajax Applications. 2009.

19. **Craing Walls, Ryan Breidenbach.** Spring in Action. 2005.

20. **Manuel A. Borroto Santana.** Trabajo de Diploma: Diseño e implementación de los subsistemas Créditos y Depósitos del sistema Quarxo. 2011.

21. Popularidad mundial de los frameworks PHP [septiembre 20, 2012] www.symfony.es/.../popularidad-mundial-de-los-frameworks-php/

22. **Iglesias, Adolfo Miguel. Doc.** Arquitectura del sistema de modernización bancaria (Finixu). La Habana: s.n., Noviembre 2008

23. **Vera Santana, Diana Cristina, Lara Vásquez, Rubén Alejandro y Echeverría Briones, Pedro Fabricio**. Implementación de un portal web para la automatización del proceso de consultorías de mentores GOLD de la Región Latinoamericana del IEEE (R9), utilizando arquitectura Java 2 Enterprise Edition - J2EE y tecnología AJAX. Guayaquil, Ecuador : s.n., 2009

24. **Alain Sánchez Ledón, Ismaury Pérez Figueroa.** Trabajo de diploma: Análisis y diseño del módulo emisión de carta de crédito del subsistema comercio exterior del proyecto modernización del sistema bancario cubano. 2009.

25. **Barroso y Bello.** Trabajo de diploma: Diseño e Implementación del Subsistema Cartas de Créditos Del Proyecto SAGEB. 2010

26. **Torrijos, Ricard Lou.** Programación en Castellano. Catálogo de Patrones de Diseño J2EE. [En línea][Citado el: 15 de marzo de 2011.] http://www.programacion.com/articulo/catalogo_de_patrones_de_diseno_j2ee__y_ii:_capas_de_negocio_y_de_integracion_243/8

27. **Zukowski, John.** Programación Java 2 J2SE 1.4. s.l.: Anaya Multimedia, 2003.

28. **Slideshare.** http://www.slideshare.net/jmontilva/qu-es-el-modelado-de-negocios [febrero, 2011].

29. **Informatizate**.http://www.informatizate.net/articulos/dime_como_programas_y_te_dire_quien_eres_23082004.htmfecha.

30. **I.Jacobson, G. Booch, J. Rumbaugh.**El Proceso Unificado de Desarrollo de Software. Madrid: Perarson Educación, 2000

31. Diagrama de despliegue virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc?

32. **Dr. Macario Polo Usaola, Departamento de Informática.** Mantenimiento Avanzado de Sistemas de Información. Pruebas del software. Paseo de la Universidad, Cuidad Real: s.n

33. **Tahchiev**, P. y otros, *Junit in action*. 2010: Manning Publications Co

34. **Stephen A. White**, IBM Corp. Process Modeling Notations and Workflow Patterns. United States

35. Normalización de Base de datos y Técnicas de diseño. http://www.lsi.upc.edu/~bcasas/docencia/pfc/NormalitzacioBD.pdf

36. Definición de Tasa de cambio. http://www.tasas.us/interes/inflacion/crecimiento/definicion_de_tasa_de_cambio/

## GLOSARIO DE TÉRMINOS

**Extracto bancario:** Indica todas las actividades o movimientos hechos sobre una cuenta durante un determinado plazo de tiempo.

**Trazabilidad:** Es la habilidad de seguir la evolución de los requisitos durante el ciclo de vida del proyecto.

**Transacciones financieras:** operaciones financieras que se realizan sobre las cuentas.

**Débito:** Acción de debitar una cuenta, según la naturaleza de la cuenta si es acreedora significa disminuir.

**Crédito:** Acción de acreditar una cuenta, significa aumentar la cuenta.

**Estado de Cuentas:** Es la verificación y resumen, de los distintos créditos y los distintos deudores de una cuenta en específico.

**Libro de Banco:** Donde se anotan cronológicamente los movimientos de las cuentas bancarias, tanto los depósitos, abonos, giros, cargos etc., esto se hace detallando el movimiento de cada transacción que se realiza, y a fin de mes se realiza la conciliación bancaria, donde ves si la información del banco es igual a la que tu llevas en tus registros, esto se hace para evitar fugas de dinero por errores de los bancos y llevar claramente el movimiento de tus fondos.

**Opening:** Tipo de cambio que se utiliza en Banco Nacional de Cuba para desarrollar el Inicio del día contable.

**Closing:** Tipo de cambio que se utiliza en Banco Nacional de Cuba para desarrollar el Cierre del día contable.

**Asiento:** Denominaremos asientos a las anotaciones realizadas con la finalidad de reflejar un hecho o una operación contable. El asiento puede ser Simple (una cuenta deudora y una cuenta acreedora) o Compuesta (dos o más cuentas deudoras o acreedoras).

**Servlet:** Pequeño programa que se ejecuta dentro del entorno de un Servidor Web, en el servidor.

**JSP:** Es una tecnología Java para crear contenido dinámico para web en forma de documentos HTML oXML.

**HTML:** Lenguaje para la elaboración de páginas web, es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

**XML:** Es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium permitiendo definir la gramática de lenguajes específicos. XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades.

**Plugins:** Un plug-in es un módulo de hardware o software que adiciona una característica o un servicio específico a un sistema existente.

**IDE:** Un entorno de desarrollo integrado constituye un programa informático compuesto por un conjunto de herramientas para la programación.

**SIGEP:** Es una aplicación web que apoya el seguimiento y control de proyectos. Permite de manera ágil conocer el estado de un proyecto que ha sido o se está desarrollando. El sistema posee las siguientes funcionalidades: Controla y evalúa la gestión de los proyectos, permite medir el tiempo de duración de un proyecto, apoya el proceso de administración de proyectos, en cuanto al control y seguimiento de las tareas determinadas en el plan de trabajo o cronograma del proyecto.

**Applet:** Es un programa que puede incrustarse en un documento HTML. Cuando un navegador carga una página web que contiene un applet, este se descarga en el navegador web y comienza a ejecutarse. Esto permite crear programas que cualquier usuario puede ejecutar con tan solo cargar la página web en su navegador.