Universidad de las Ciencias Informáticas

Facultad 3

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

"Desarrollo del subsistema Conciliaciones Bancarias del sistema QUARXO"

Autores:

Yakelín Parra Batista

Leonardo Vázquez Arzuaga

Tutores:

Ing. Yadira Calimano Meneses

Ing. Rafael Bello Lara

Co-Tutora:

Ing. Lissett Díaz Mesa

La Habana, Julio 2012

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 _____.

_____	_____
Yakelín Parra Batista	Leonardo Vázquez Arzuaga
Firma del Autor	Firma del Autor
_____	_____
Ing. Yadira Calimano Meneses	Ing. Rafael Bello Lara
Firma del Tutor	Firma del Tutor

DATOS DE CONTACTO

Yadira Calimano Meneses

Graduada de Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).

Correo: ycalimano@uci.cu

Rafael Bello Lara

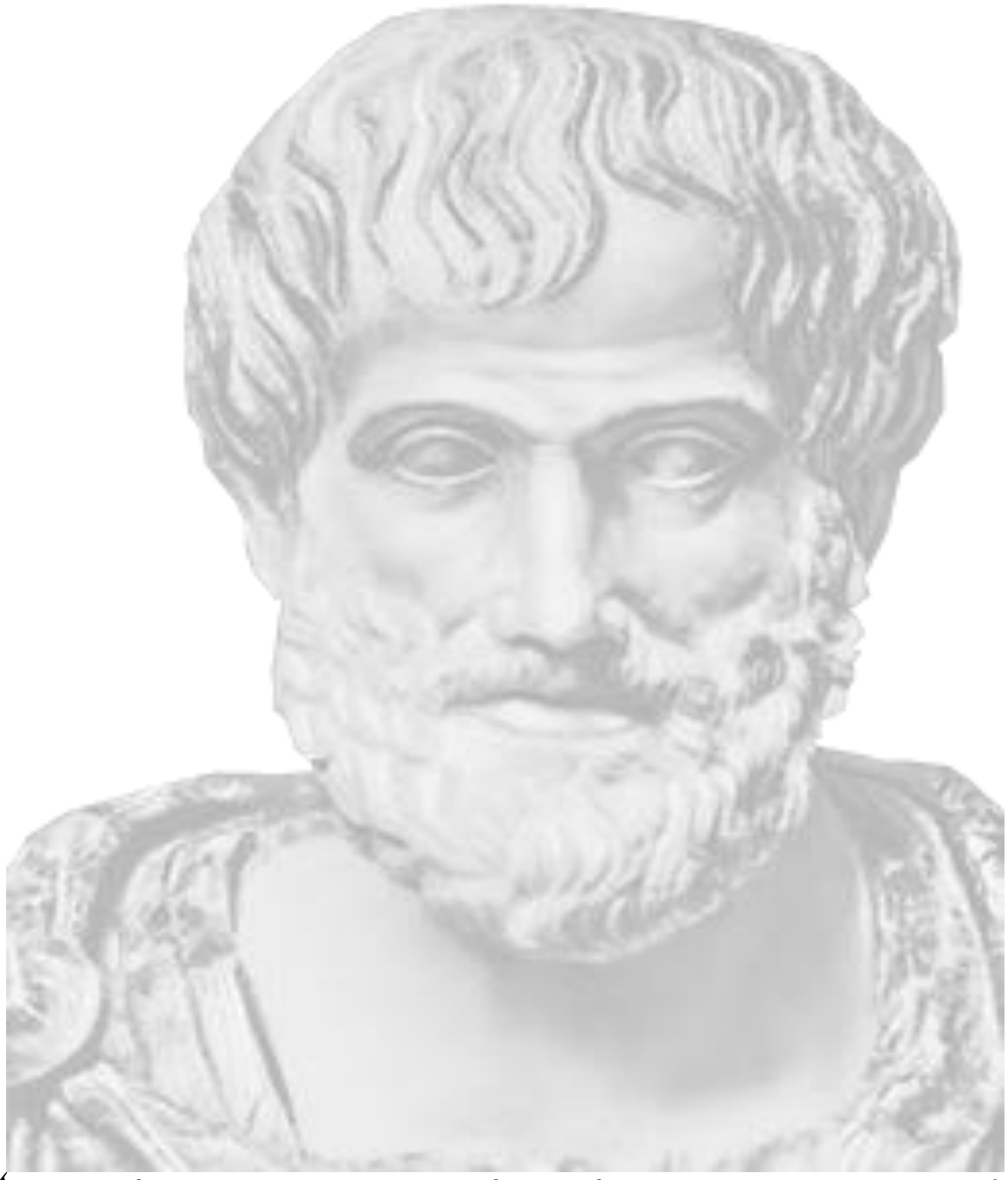
Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).

Correo: rbello@uci.cu

Lissett Díaz Mesa

Graduada de Ingeniera en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI).

Correo: ldiazm@uci.cu



"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica."
Aristóteles (384 AC-322 AC) Filósofo griego.

DEDICATORIA

A las tres personas más importantes en mi vida; mi mamá, mi abuela Reina y mi abuelo Alcibíades.

A mis tíos preferidos Belquis, Nelson y Joseito.

Yakelín

A las personas más importantes en mi vida, mi mamá Silvia, y mi papá Dagoberto, gracias por todo, sin ustedes y sin su amor nada de esto hubiera sido posible.

A la reina y dueña de mi corazón, gracias por apoyarme tanto en todo este tiempo y por estar ahí siempre que lo necesite.

Leonardo

AGRADECIMIENTOS

A mi mamá por ser la madre que es y por su fuerza inspiradora.

A mis otros padres; mis abuelos Reina y Alcibíades a quienes debo lo que soy.

A mis tíos preferidos que han estado presentes toda mi vida, que tanto me quieren y me han apoyado.

A mis tíos Beto y Raciel por su cariño y preocupación.

A mi prima Madelin por ser mi mano derecha.

A mi primo Edilberto por ayudarme tanto.

A mi padrastro Ramón por todo lo que ha hecho por mí, mil gracias.

A mi familia en la Habana a quienes voy a extrañar mucho, gracias por su inmenso cariño y apoyo incondicional.

A toda mi bella familia en general por su amor eterno, gracias.

A los viejos amigos que siempre están ahí para mí a pesar de la distancia y en especial a Caridad que tanto me ha ayudado y la quiero mucho.

A los nuevos amigos que han sido geniales a lo largo de la carrera en especial Arlethy, Libet y Puig que han sido mi látigo y mis cómplices.

A mi compañero de tesis Leonardo que realizó un trabajo excelente e hizo posible todo esto.

A todos los que de una manera u otra nos ayudaron; los tutores, los muchachos del proyecto, profesores etc.

A la Revolución y a Fidel.

Yakelín

A mis padres, por ser el pilar en el cual me apoyé para llegar a este momento cumbre de mi carrera y sin los cuales nunca hubiera llegado este momento tan importante en mi vida. Gracias.

A mi novia por ayudarme siempre, por apoyarme y por estar ahí en cada uno de los momentos que la necesité y que fueron muchos, gracias amor.

A mi familia, que siempre me han apoyado y exhortado para que salga adelante y que llevo a cada uno de ellos en mi corazón.

A mi hermana Daliana que aunque no sepa lo que hago siempre está pendiente de todos mis logros.

A mis tías Celina, Graciela, Mariana, Bertha, y a mis primas Lilian, Yulia, Mayelin, Celeste, Yesenia y la China, a mis infinitos primos y primas, a mis queridos amigos del barrio.

A mis hermanos Andrés y Rodrigo, que aunque no de sangre si de corazón, jamás me olvidare de ellos.

A mi compañera de tesis Yakelín, gracias te doy por todo, sin tu ayuda y esfuerzo esto no sería posible.

A los tutores que nos apoyaron y nos brindaron su ayuda y experiencia.

A mis especiales compañeros de grupo tanto de este último como del anterior.

A todos aquellos que de una forma u otra han estado presente en mi vida, y con los cuales he compartidos todo tipo de momentos. A todas las personas que brindaron su ayuda.

Leonardo

RESUMEN

Como parte de la modernización del sistema bancario cubano la Universidad de las Ciencias Informáticas (UCI) tiene la tarea de desarrollar un software para el Banco Nacional de Cuba (BNC) que sea capaz de gestionar todas las transacciones que se realizan diariamente en dicha entidad.

El Departamento de Conciliación Bancaria del BNC es el área encargada de controlar todas estas transacciones mediante la realización del proceso de Conciliación de Cuenta, que actualmente se realiza de forma manual. Debido a su gran importancia se necesita una herramienta capaz de realizar este proceso. Por las razones expuestas anteriormente el objetivo de la presente investigación es desarrollar un subsistema Conciliaciones Bancarias independiente pero que se adhiera al Sistema de Gestión Bancaria QUARXO.

Para una mejor comprensión de los procesos de conciliación se realizó un estudio del arte acerca de algunas soluciones existentes a nivel nacional e internacional. En el proceso de construcción del subsistema se utilizará RUP como metodología de desarrollo, como lenguaje de modelado UML y como herramienta CASE Visual Paradigm.

Palabras claves: Conciliación Bancaria, Estado de Cuenta, Libro de Banco, SAGEB, QUARXO.

TABLA DE CONTENIDO

INTRODUCCIÓN

El perfeccionamiento y evolución de los sistemas informáticos es uno de los grandes retos del presente siglo; el cual exige mejoras significativas para apoyar la informatización de los procesos que se gestionan en las diferentes entidades, dentro de las cuales se destacan las instituciones bancarias con la adopción de tecnologías y herramientas de punta.

El sector bancario actual está inmerso en una economía globalizada y un entorno de alta competitividad, enfrentándose a constantes desafíos, no sólo en el aspecto comercial de su negocio, sino también en relación a la innovación operativa de su gestión interna, que le permita sustentar el crecimiento eficiente de su negocio, así como retener y aumentar el valor de sus clientes.

En este contexto, las Tecnologías de la Información (TI) son una de las herramientas esenciales para que las entidades bancarias puedan enfrentar dichos desafíos, con una sólida gestión no sólo de sus procesos, transacciones e información interna, sino también con un profundo conocimiento de los clientes actuales y potenciales del mercado.

Los bancos son instituciones financieras con la misión de administrar fondos, apoyándose en el conjunto de productos y servicios que brindan a fin de garantizar su rentabilidad y crecimiento, la productividad del trabajo, la facilidad en los flujos de pagos, la reducción de costos y el alcance de una buena calidad para mejorar la satisfacción de los clientes.

Como parte de la modernización del sistema bancario cubano la UCI tiene la tarea de desarrollar un software para el Banco Nacional de Cuba (BNC). El mismo es uno de los bancos más importantes del país. Dentro de sus funciones de Banco Comercial se encuentra como principal actividad la de respaldar las transacciones financieras que asume el Estado Cubano para la compra de alimentos, combustible, medicamentos, entre otras actividades de vital importancia para la economía del país. El proceso de verificación y confrontación de estas transacciones, es el que se conoce como Conciliación Bancaria, el cual permite revisar y confrontar los valores que tiene registrado una empresa, en una cuenta de ahorro o corriente, con los valores que le suministra el banco a través del extracto bancario.

Mensualmente, el banco envía a la empresa un extracto en el que se muestran todos esos movimientos que concluyen en un saldo de la cuenta el último día del respectivo mes. En ocasiones, el saldo del extracto bancario no es igual al saldo que la empresa tiene registrado en sus libros auxiliares, por lo que

se hace necesario identificar las causas y diferencias por las que esos valores no coinciden. Actualmente en el Banco Nacional de Cuba este proceso se realiza de forma manual a partir de las revisiones de los mensajes SWIFT (Sistema Internacional de Transacciones Financieras, por sus siglas en inglés) que llegan por el sistema de mensajería interbancaria SACIM (Sistema de Archivo, Captación e Impresión de Mensajes SWIFT) contra el reporte de Conciliación de Cuenta que se emite por el Sistema de Gestión Bancaria QUARXO.

Por lo anteriormente expuesto se identificó como **problema a resolver**: ¿Cuál es la especificación de los artefactos necesarios para la informatización de los requisitos identificados en el proceso de Conciliación Bancaria en el Banco Nacional de Cuba?

Por tanto el **objeto de estudio** se enfocará hacia el proceso de conciliación en las entidades bancarias y como **campo de acción;** proceso de Conciliación Bancaria mediante sistemas informáticos.

A partir del problema anteriormente planteado se determina como **objetivo general** desarrollar el subsistema Conciliaciones Bancarias para QUARXO que agilice el proceso de gestión y toma de decisiones en las operaciones de conciliación realizadas en el Banco Nacional de Cuba.

Teniendo como **objetivos específicos**:

- Realizar el marco teórico de la investigación.
- Desarrollar el subsistema Conciliaciones Bancarias de QUARXO.
- Validar el subsistema implementado.

Se plantea como **idea a defender** que: Con el desarrollo de un subsistema Conciliaciones Bancarias para el Sistema de Gestión Bancaria QUARXO se agilizará el proceso de conciliación y toma de decisiones de las operaciones que se realizan en el Banco Nacional de Cuba.

Dando cumplimiento a los objetivos específicos y teniendo en cuenta la tesis de Análisis y Diseño que antecede al presente trabajo se trazaron las siguientes **tareas de investigación**:

- Caracterización de las tecnologías y herramientas a utilizar en el diseño e implementación del subsistema Conciliaciones Bancarias.
- Caracterización de los procesos relacionados con la Conciliación Bancaria en las entidades financieras.

- Caracterización de los sistemas automatizados nacionales e internacionales que soportan la realización de dichos procesos.
- Realización de mejoras en el análisis y diseño de la solución.
- Implementación de la solución.
- Realización de pruebas unitarias.

Para adquirir la comprensión y claridad de los contenidos de la investigación realizada se estructuró el documento de la siguiente manera:

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

En este capítulo se realiza una profunda investigación sobre los procesos de negocios asociados a la Conciliación Bancaria en entidades financieras bancarias. Se realiza un análisis del estado del arte a nivel internacional y nacional acerca de los procesos relacionados con la contabilidad, los sistemas contables y la Conciliación Bancaria. Se fundamenta sobre las metodologías, técnicas y herramientas a utilizar, entre otros aspectos de vital importancia que contribuyen en gran medida al desarrollo de la solución que se propone en el presente trabajo.

CAPÍTULO 2: ANÁLISIS Y DISEÑO.

En este capítulo se realiza una caracterización del sistema; identificando trabajadores, involucrados y artefactos dentro de los procesos del negocio. Se definen, especifican y validan los requerimientos funcionales del subsistema Conciliaciones Bancarias para el Sistema de Gestión Bancaria QUARXO. Se muestran los artefactos generados en el análisis y diseño de la propuesta solución, se analizan los casos de uso críticos, definiendo las clases del análisis para cada uno, así como la organización de los módulos mediante un diagrama de paquetes. Además en este capítulo se realiza el diseño de la solución donde se esbozan todas las clases para cada uno de los objetos determinados y se propone un modelo de datos que sustente las clases desarrolladas.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA.

Este capítulo se centra principalmente en la implementación del subsistema Conciliaciones Bancarias, mostrando y explicando en cada caso la interacción entre componentes del sistema a través de

diagramas, descripción de un subconjunto de clases y funcionalidades. Para finalizar el capítulo se realiza la validación a partir de las pruebas de unidad (Estructurales y Funcionales), brindando una explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y las pruebas de caja negra que se realizaron a la interfaz del mismo, en aras de localizar errores que imposibiliten el cumplimiento de los requisitos funcionales y las perspectivas del cliente.

Para cada capítulo se ofrecen sus conclusiones parciales y al final del documento se exponen las conclusiones generales, las recomendaciones propuestas, la bibliografía, los anexos y el glosario de términos.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

El presente capítulo en su primera parte realiza el estado del arte acerca de la Conciliación Bancaria, partiendo de la definición de conceptos asociados a la Conciliación de Bancos. Se incluye además, un análisis sobre algunas soluciones existentes a nivel nacional e internacional asociadas al tema. En su segunda parte se realiza un estudio relacionado con las herramientas, técnicas, lenguajes y notación que se propone para el desarrollo de la solución, con el fin de lograr de forma satisfactoria los objetivos propuestos.

1.2. Contabilidad

"La Contabilidad es la Ciencia que proporciona información de hechos económicos, financieros y sociales suscitados en una empresa; con el apoyo de técnicas para registrar, clasificar y resumir de manera significativa y en términos de dinero, “transacciones y eventos”, de forma continua, ordenada y sistemática, de tal manera que se obtenga información oportuna y veraz, sobre la marcha o desenvolvimiento de la empresa u organización con relación a sus metas y objetivos trazados, con el objeto de conocer el movimiento de las riquezas y sus resultados ". [9].

1.3. Operaciones inter-bancarias

Se denominan operaciones inter-bancarias a aquellas operaciones de crédito o débito practicadas entre bancos de manera profesional, como eslabón de una serie de operaciones activas y pasivas similares.

Como motivo de estas operaciones cada banco está en la obligación de enviar a la persona o institución que tiene una cuenta corriente abierta en su entidad, una relación detallada del movimiento de esa cuenta en el mes y el saldo al final del mismo. En ocasiones este saldo no coincide con el saldo que la cuenta "Banco" refleja en “Nuestros Libros” por lo que se requiere hacer una Conciliación Bancaria cada mes para determinar las causas de las diferencias existentes y conseguir el saldo correcto. [10].

1.4. Conciliación Bancaria

La Conciliación Bancaria es la encargada de comparar las anotaciones que figuran en el Estado de Cuenta Bancario con el “Libro de Banco” de nuestra empresa, a los efectos de determinar el origen de las diferencias. El proceso de conciliación, consiste en identificar la igualdad entre las anotaciones contables y

las constancias que surgen de los resúmenes bancarios, efectuando el comparar mediante un básico ejercicio de control, basado en la oposición de intereses entre nuestra empresa y la institución bancaria. La Conciliación Bancaria no es un registro contable, es una herramienta de control. [1].

En otras palabras; es el proceso que 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 registra cada uno de los movimientos realizados en una cuenta bancaria, como son el giro de cheques, consignaciones, notas débito, notas crédito y anulación de cheque. La entidad financiera donde se encuentra la respectiva cuenta, por su parte lleva un registro completo de cada movimiento que el cliente (la empresa), hace en su cuenta.

1.4.1. Proceso de Conciliación Bancaria en el BNC

Mensualmente, el banco envía a la empresa 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 lo general, el saldo del extracto bancario no coincide con el saldo que la empresa tiene en sus libros auxiliares, por lo que es preciso identificar las diferencias y las causas por las que esos valores no coinciden, las cuales se presentan a continuación:

- 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 cualquier 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 nos cargue o abone en nuestra cuenta operaciones que corresponden a otro cliente del Banco.
- Otras diferencias: Algún otro tipo de diferencia que ocurre con menor frecuencia.

El volumen de transacciones diarias entre el BNC y otras entidades bancarias oscila entre los 1000 y 5000, lo que conlleva a que el proceso de conciliación sea demasiado extenso. Una persona puede demorarse 15 días o más identificando las diferencias entre las operaciones, trayendo consigo que el banco se vea obligado a pagar intereses por sobregiro de sus cuentas. En muchas ocasiones se cometen errores, que atentan contra la operatividad diaria del banco, provocando un sobreesfuerzo de sus trabajadores y gasto de recursos materiales.

1.4.1.1. Conciliación de Cuenta

El sistema de registro contable, en su fase final, permite la elaboración de la información financiera, información que debe ser veraz y confiable.

Al menos, son dos los documentos informativos, el Balance General y el Estado de Resultados de Operación, los que deben ser formulados por medio de la información particular de cada cuenta contable que los forman. Cada una de ellas debe estar respaldada por su saldo respectivo y un desglose o descripción de su contenido en un documento auxiliar llamado “relación analítica” debidamente comprobada.

Cada saldo de cada cuenta debe ser conciliado con la información de cada relación analítica. Si en la comparación surgen diferencias, se deben buscar las causas que las originaron y emitir los documentos correctivos, para registrar los cambios donde corresponda y obtener la igualdad de los saldos. A esto se le llama Conciliación de Cuenta. La Conciliación Bancaria es un caso especial de Conciliación de Cuenta. [10].

1.4.1.2. Conciliación entre bancos

Durante la conciliación entre bancos se comparan dos movimientos financieros, de cuentas recíprocas, y de saldos contrarios, haciendo que en un momento dado, dichos saldos sean iguales aplicando las diferencias que se hayan encontrado. Se realiza con el objetivo de presentar en la información financiera el saldo correcto del efectivo disponible, en una fecha determinada, identificando las diferencias entre las cuentas corriendo los asientos de ajuste en libros, en base a la información que arroje la conciliación.

Las dos clases más comunes de conciliación son: individual y la conjunta, las cuales requieren para su elaboración del extracto de cuenta corriente, el reporte del Libro Auxiliar de Bancos y la conciliación anterior.

Conciliación individual: consiste en colocar de acuerdo el saldo del Libro Auxiliar de Bancos con el saldo mostrado en el extracto bancario, mediante los ajustes contables que sean necesarios para actualizar el saldo del libro auxiliar.

El método consiste en observar las operaciones que realizó el banco y las operaciones que registró la empresa durante el ejercicio y otras que habían sido contabilizadas en el anterior período que se observan en la conciliación anterior, para llevar las diferencias en la Conciliación Bancaria, en notas de contabilidad, al comprobante de contabilidad de ajustes y registrarlas en el Libro de Bancos y otros libros principales.

Conciliación conjunta: como su nombre lo indica lleva a la par los dos saldos y en cada saldo se ajusta de acuerdo a las operaciones que le son propias, o porque de todas maneras se realizarán en el ejercicio económico siguiente, poniendo de acuerdo los dos saldos: el bancario y el libro auxiliar. [10].

Existen dos formas de realizar la conciliación:

- Conciliación aritmética: consiste en conectar de forma numérica los saldos utilizando las diferencias encontradas. Partiendo del saldo de la empresa para llegar al saldo del banco o viceversa.

- Conciliación contable: la conciliación aritmética se realiza sin importar a quien le corresponde corregir dichas diferencias en sus respectivos registros de contabilidad. Para esto se utiliza la conciliación contable, que es un documento formal donde la información presenta la distribución de las partidas no correspondidas, tanto de la empresa como del banco. Haciendo las anotaciones convenientes que permitan elaborar las correcciones del saldo en libros de la empresa y los avisos para que el banco haga lo propio, en su caso.

1.4.1.3. Medios de comunicación entre bancos (SWIFT)

Dentro de los medios de comunicación entre bancos para realizar el proceso de Conciliación Bancaria se encuentra la mensajería SWIFT (System Worldwide Internacional Financial Transactions).

SWIFT es un sistema computacional a nivel mundial de comunicaciones, con sede en Bruselas, que permite a los bancos de distintos países intercambiar información relacionada con las operaciones que le son propias. [10].

1.4.2. Sistemas de Conciliación Bancaria

En muchas ocasiones el proceso de comparación de las operaciones durante la Conciliación Bancaria se hace demasiado extenso, trayendo consigo errores que atentan contra la operatividad diaria de las instituciones bancarias, es por ello que se necesita una pronta informatización de dicho proceso para que la gestión del mismo en las diferentes entidades se realice de forma satisfactoria.

Actualmente muchas instituciones distribuidas por todo el mundo, entre la amplia gama de procesos informatizados tienen la conciliación como uno de ellos. Tal es el caso de los sistemas SIC.solucions, Sistema Isis Classic, Winledger, SICOB, PA CONCILIACIÓN BANCARIA, Conciliación Express y SIAFI.

SIC.solucions

Esta herramienta web creada en Barcelona tiene integrado un Sistema de Conciliación Bancaria que posibilita en gran medida la mejora de la gestión de sus procesos contables. [2].

Este sistema de Conciliación Bancaria que automatiza SIC.solucions 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.

Añade además la integración automática de la información bancaria la cual mantiene un registro histórico de movimientos bancarios. Permite el intercambio de datos electrónicos con los bancos.

Sistema Isis Classic

Es un software de gestión comercial diseñado para la administración eficiente de las pequeñas y medianas empresas. [3].

Concilia contra el extracto bancario. Sin tener que ingresar el mismo, controla todas las partidas ingresadas por el sistema ordenándolas de la misma manera que se encuentran en el extracto.

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

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. [4].

Características Generales:

1. Se alimenta de la información del módulo de contabilidad Winledger.

2. 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.

3. Conciliación automática, a través de números de referencias.

4. En caso de divergencia en los números de referencias, el sistema permite conciliaciones manuales en forma automatizada.

SICOB (Sistema de Conciliación Bancaria)

Es un software desarrollado para las áreas de Tesorería, Contraloría o Contabilidad de las empresas que le permitirá automatizar su proceso de Conciliación. [5].

Funcionalidad 1: Carga de Información.

Funcionalidad 2: Conciliación Automática.

Funcionalidad 3: Conciliación Manual.

Funcionalidad 4: Conciliación por Rango de Ajuste.

Funcionalidad 5: Conciliación con Soporte escrito.

PA CONCILIACIÓN BANCARIA

Es un sistema de conciliación que automatiza los procesos de recogida de información bancaria y contable y realiza la conciliación evaluando la información disponible para lograr unos altísimos porcentajes de conciliación de una forma totalmente segura. [6].

Para conseguir unos porcentajes de conciliación del 100%. El sistema se apoya en cuatro tipos de conciliaciones:

- Conciliaciones automáticas.
- Conciliaciones propuestas.
- Conciliaciones multiapunte.
- Conciliaciones manuales.

Conciliación EXPRESS (Sistema Automático de Conciliación de Cuenta en línea)

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. [7].

SIAFI (Sistema Integrado de Administración Financiera)

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 “online” al software. [8].

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.

Módulo de Conciliación del BM

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.

De los sistemas mencionados anteriormente ninguno satisface completamente las necesidades del BNC; debido a que estos sistemas son software privativo y sería más costoso adquirir alguno de ellos que hacer uno nuevo, además de que no se ajustan a las necesidades del BNC. En el caso particular del Módulo de Conciliación del BM destacar que es un software desarrollado sobre FOXPRO que es una herramienta privativa de Microsoft y no tiene vida propia ya que está estrechamente ligado al SABIC (Sistema Automatizado para la Banca Internacional de Comercio).

Por lo anteriormente expuesto surge la necesidad de un nuevo subsistema Conciliaciones Bancarias mediante el cual se agilizará la gestión de las transacciones que diariamente se realizan entre el BNC y el resto de las entidades bancarias además de reducir notablemente los errores.

1.5. Metodología, lenguaje de modelado y herramientas utilizadas

En las empresas de desarrollo de software la definición de las tecnologías en la actualidad es la base primordial para la ejecución de cualquier sistema, originando la utilización de metodologías, técnicas y lenguajes que rijan y soporten el proceso de desarrollo de software en correspondencia con los intereses del cliente.

Debido a esto es muy importante conocer las herramientas y tecnologías definidas por el grupo de arquitectura del proyecto SAGEB[1], ya que estas influyen considerablemente en el desarrollo y la calidad del producto, a continuación se realiza una breve caracterización de las mismas, mostrando así los

[1]SAGEB: Sistema Automatizado de Gestión Bancaria.

beneficios que aportan en el trabajo de análisis, diseño e implementación del subsistema Conciliaciones Bancarias.

1.5.1. Metodología de desarrollo de software

Algunas de las metodologías pesadas más empleadas en el mundo de la producción de software son: RUP (Rational Unified Process) y MSF (Microsoft Solution Framework). Por su parte la metodología ágil más utilizada es: eXtreme Programming (XP). [15].

De las metodologías estudiadas se escogió RUP, la cual es una de las más utilizadas actualmente en el mundo para el desarrollo de software, debido a que RUP propone flujos de trabajo en los que se definen las secuencias de actividades, quienes las deben desarrollar y los artefactos a generar. Además la utilización de RUP como metodología de desarrollo, no sólo garantiza la aplicación de buenas prácticas recomendadas, probadas y una arquitectura configurable, sino que se tendrá unificado todo el equipo de desarrollo de software.

El ciclo de vida de RUP se caracteriza por:

- Dirigido por casos de uso: Los CU (casos de uso) reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.
- Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

1.5.2 Lenguajes y herramientas para el modelado

Existen varios software para el modelado UML (Unified Modeling Language), son las denominadas herramientas CASE[2]. Entre estas herramientas se encuentran; Enterprise Architect, Visual Paradigm y Rational Rose.

[2] Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador)

Visual Paradigm: Constituye una herramienta para facilitar el trabajo durante la confección de un software así como garantizar la calidad del producto final, fue creada para el ciclo completo de desarrollo del software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación, también proporciona características tales como generación del código, ingeniería inversa y generación de informes. [10].

Se escoge como herramienta de modelado el Visual Paradigm para UML 6.4 debido fundamentalmente a que es multiplataforma (incluyendo Linux) y soporta modelado tanto de todas las fases como del modelo de datos. Soporta además ingeniería directa e inversa y todos los diagramas UML, entidad-relación y el ciclo de vida del desarrollo de software completo: análisis, diseño, implementación, pruebas y despliegue.

1.5.3. IDE de desarrollo (Eclipse Galileo 3.5)

Entre los principales IDE's de desarrollo para el lenguaje Java se encuentran Eclipse, Netbeans e IntelliJ Idea. [16].

Todas estas herramientas son muy potentes y facilitan en gran medida el desarrollo con el lenguaje Java. La utilización de IntelliJ Idea fue descartada desde el inicio por su condición de software privativo. Entre el Netbeans y el Eclipse se analizó la experiencia de los desarrolladores con los mismos, comprobándose que en su mayoría usaban Eclipse debido a las prestaciones de las estaciones de trabajo y al bajo consumo de recursos que evidencia este frente al Netbeans. Además constituye un armazón sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la instalación de los plugins adecuados. La arquitectura de plugins[3] de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías. [13].

1.5.4. Contenedor Web (Apache Tomcat 1.6)

Como soporte para el funcionamiento de las aplicaciones desarrolladas en Java existen varios entornos de ejecución como Jboss Server, GlassFish y Apache Tomcat, este último no es considerado como un

[3] Un plugin es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande

servidor de aplicaciones, sino como un contenedor de servlets, pero es capaz de soportar la mayoría de las aplicaciones desarrolladas en dicha tecnología. [18].

JBoss Application Server (JBoss AS) es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado actualmente. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto. [19].

GlassFish es un servidor de aplicaciones que implementa la plataforma JavaEE, soporta las últimas versiones de tecnologías como: JSP, JSF, servlets, arquitectura Java para Enlaces XML y muchas otras tecnologías. [20].

Apache Tomcat está publicado bajo la licencia del software de Apache, es un contenedor de servlets que implementa las especificaciones de JavaServer Page (JSP). Es desarrollado en un entorno abierto. Tomcat puede funcionar como servidor web por sí mismo. Es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. [10].

Teniendo en cuenta las características expuestas, se selecciona Apache Tomcat como servidor de aplicaciones. Debido a que todos los servidores analizados tienen mucha similitud en lo referente a sus funcionalidades, se utilizó la experiencia del equipo de desarrollo y los bajos recursos que este demanda como factores fundamentales en la toma de esta decisión.

1.5.5. Control de versiones (Subversión)

Para el control de versiones existen distintos software que se encargan de realizar estas actividades, entre los más usados internacionalmente se encuentran: SVN (Subversión), CVS[4] y Git.

SVN es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es un software libre bajo la licencia Apache y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversión es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente; en cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

Existen dos razones fundamentales para el uso de esta herramienta:

[4]CVS: Concurrent Versions System (Sistema de versiones concurrentes)

- Gestiona las modificaciones durante el desarrollo.
- Permite que varias personas trabajen sobre los mismos ficheros.

CVS es un software de código abierto que permite gestionar los cambios realizados sobre códigos fuentes de cualquier archivo en varias plataformas (C, C++, Java, etc.). Pero tiene como desventajas que no ofrece facilidades para hacer referencia a cambios en múltiples ficheros y no permite renombrar o copiar ficheros dentro del sistema de control por lo que puede ser muy doloroso reorganizar el código después de iniciar el proyecto. [21].

Git: es un sistema de control de versiones distribuidas, es decir no hay una base de código centralizada para extraer el código. Esto lo hace diferente de otros sistemas de control de versiones, como CVS y SVN que usan un control centralizado de versiones. [22].

Según las características particulares de cada herramienta se seleccionó Subversión para el control de versiones debido a que era la herramienta más sólida que existía en el momento de la selección, además se contaba con el apoyo de personal con conocimiento y experiencia en la administración de dicha herramienta.

1.5.6. Gestor de Bases de Datos (SQL Server 2005)

Este programa provee herramientas sólidas para el trabajo con base de datos, 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. [13].

Aunque este gestor de Base de datos es un software privativo su selección fue impuesta por el cliente, debido a que para acelerar el desarrollo se utilizará el núcleo del SABIC el cual incluye un grupo de tablas y procedimientos almacenados que ayudarán en la implementación del sistema.

1.6. Ambiente de desarrollo

El ambiente de desarrollo no es más que el conjunto de herramientas, frameworks, tecnologías utilizadas en el desarrollo del software. A continuación se presentaran una breve descripción de la propuesta presentada para utilizar en el proyecto SAGEB donde se analizan patrones de diseño, herramientas, lenguajes, plataformas y frameworks utilizados durante la construcción del subsistema Conciliaciones Bancarias. [10].

1.6.1. Plataforma J2EE[5]

J2EE es una plataforma precisa, un estándar para el desarrollo de aplicaciones empresariales multicapa. Simplifica las aplicaciones empresariales organizándolas en componentes modulares y estandarizados, 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. [28].

1.6.2. Lenguaje Java

Entre los lenguajes de programación del lado del servidor más usados en el desarrollo web se encuentra PHP, JAVA y C#. PHP es un lenguaje gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. [23].

En cuanto a C# es un lenguaje simple, potente, seguro y orientado a objetos, es desarrollado y estandarizado por Microsoft como parte de la plataforma .NET; la principal desventaja de este lenguaje es que es dependiente de la plataforma a menos que se utilice MonoDevelop[6] para poder hacer de C# un lenguaje multiplataforma. [24].

Java es un lenguaje de programación orientado a objetos, robusto y fácil de aprender, permite la codificación de la propuesta de solución de una manera rápida y flexible, al integrarse con los diferentes frameworks de desarrollo, permite programar páginas web dinámicas con accesos a bases de datos utilizando XML con cualquier tipo de conexión de red entre cualquier sistema. [13].

Se seleccionó Java primeramente por todas las características que presenta el lenguaje, atendiendo al conocimiento y preparación del equipo de desarrollo con que se contaba al iniciar el proyecto, basándose

[5]Java Platform Enterprise Edition: es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en Lenguaje de programación Java.
[6]MonoDevelop: es un entorno de desarrollo integrado, libre y gratuito, diseñado primordialmente para poder utilizar C# en distintas distribuciones de Linux y Mac.

además en los resultados obtenidos por el proyecto SIGEP (Sistema de Gestión Penitenciaria), el cual usó este lenguaje y desarrolló varias librerías para el mismo con el objetivo de llevar a cabo un desarrollo mucho más rápido logrando montar una arquitectura robusta que se adaptaba a las necesidades del sistema QUARXO.

1.7. Tecnologías y Frameworks

1.7.1. Framework

Los frameworks son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados. Un framework no tiene funcionalidades de una aplicación específica, sino que las aplicaciones se construyen sobre ellos. [10].

1.7.2. Spring Framework

Del lado del servidor existen muchas tecnologías que facilitan el desarrollo de aplicaciones, entre las más populares se encuentran: Struts, JSF y Spring. [25].

El framework Struts brinda soporte para el desarrollo de aplicaciones web bajo el patrón MVC para la plataforma JEE. Struts permite reducir el tiempo de desarrollo ya que implementa un solo controlador (ActionServlet) que evalúa las peticiones del usuario mediante un archivo configurable (struts-config.xml).

El framework JSF es un estándar para aplicaciones web basadas en Java, facilita la construcción de aplicaciones siguiendo el patrón MVC, modelo de componentes orientado a objetos, desarrolladores de componentes, desarrolladores de lógica de aplicación, uso de simples clases java como controladores, fácil incorporación de potencialidades AJAX, posee un conjunto prefabricado de componentes de interfaz de usuario y modelo de programación orientado a eventos. [26].

Spring MVC es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java. Por su diseño el framework ofrece mucha libertad a los desarrolladores en Java y soluciones muy bien documentadas y fáciles de usar. Está diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además, intenta mantener un mínimo acoplamiento entre la aplicación y el propio framework de forma que podría ser desvinculada de él sin demasiada dificultad. [13].

Entre las tecnologías anteriormente descritas se seleccionó Spring MVC como herramienta de desarrollo debido a que este es ideal para aplicaciones de gran tamaño y posee una comunidad de gran prestigio a

nivel internacional (http://www.springsource.org/) que facilita una buena retroalimentación. Otra razón considerada fue su uso en la UCI en proyectos exitosos, un ejemplo de ello es el proyecto SIGEP del cual se tomó la arquitectura base de QUARXO.

1.7.3. Hibernate

Los frameworks ORM[7] se utilizan para el trabajo con el acceso a datos de una aplicación, entre ellos se encuentran IBATIS, Spring JDBC e Hibernate que son soportados por la plataforma JEE.

IBATIS toma los mejores atributos e ideas de las tecnologías ORM y Data Mapper (Mapeador de datos) encontrando un equilibrio entre ellas, convirtiéndose así en una solución híbrida. Algunas de las ideas que toma son: el soporte para procedimientos almacenados, SQL dinámico, SQL en línea y el Mapeo Objeto Relacional.

Spring JDBC es un módulo perteneciente al framework Spring. El mismo posee algunas librerías de clases para trabajar con base de datos a través del API de Java JDBC. Unas de las características principales de este módulo es el soporte que brinda para invocar procedimientos y funciones almacenadas.

Hibernate es un framework objeto/relacional y un generador de sentencias SQL[8]. Permite y hace más viable el mapeo de atributos entre una base de datos relacional y el modelo de objetos de una aplicación, mediante archivos declarativos XML haciendo más fácil esta relación. Hibernate tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales y la consulta de estas bases de datos para obtener objetos. [13].

Fue seleccionado Hibernate principalmente por la sólida integración con el framework Spring facilitando el trabajo ya que este libera en gran medida al programador de mantener el control sobre las sesiones. Una de las principales clases que nos brinda el framework para su integración es “HibernateTemplate” la cual brinda un gran número de métodos para el trabajo con los datos.

1.7.4. Dojo Toolkit

En el año 2007 los framework Javascript más usados eran Prototype y Dojo.

[7] ORM: Object Relational Mapping(Mapeo objeto-relacional)
[8] Structured Query Language (Lenguaje de consulta estructurado): es un lenguaje declarativo de acceso a bases de datos

Prototype es un framework escrito en JavaScript que implementa las técnicas AJAX y su potencial es aprovechado al máximo cuando se desarrolla con Ruby On Rails. Prototype simplifica parte del trabajo cuando se pretende desarrollar páginas altamente interactivas. [27].

Dojo Toolkit es un 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. Proporciona además variadas opciones en una sola biblioteca haciendo un mejor trabajo que sustenta los nuevos y viejos Browsers, resolviendo los problemas de compatibilidad entre los navegadores. Tiene múltiples puntos de entrada, es independiente del intérprete y unifica estándares de codificación. [13].

Tanto Prototype como Dojo brindan funcionalidades similares pero se seleccionó Dojo debido a la fácil integración con Spring MVC. Además se tuvo en cuenta que el proyecto SIGEP desarrolló una gran cantidad de componentes y funciones las cuales se podían reutilizar en el desarrollo del sistema QUARXO.

1.8. ER/Estudio 8.0

ER/Estudio es una herramienta de modelado de datos, fácil de usar y multinivel. Se enfoca específicamente en el diseño y construcción de bases de datos a nivel físico y lógico. Brinda fuertes capacidades de diseño lógico, sincronización bidireccional de los diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes. Provee a los desarrolladores de una documentación basada en HTML[9], así como un repositorio para el modelado.

1.9. 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 profundo análisis sobre la Conciliación Bancaria y algunos conceptos de importancia para la mejor compresión de la investigación. Se hizo un estudio de los sistemas de Conciliación Bancaria existentes en Cuba y en el mundo, donde la dificultad principal para su utilización resultó ser el alto costo de sus licencias. Se caracterizaron las tecnologías y herramientas que contribuyeron a la construcción del subsistema para el cual se propone la solución.

[9]HyperText Markup Language (Lenguaje de Marcas de Hipertexto): es el lenguaje de marcado predominante para la construcción de páginas web.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

2.1. Introducción

En el presente capítulo se modelan los artefactos correspondientes al análisis de la propuesta solución para el subsistema Conciliaciones Bancarias. Se especifican las clases del análisis, se define y esboza el diagrama de paquetes con sus relaciones, para el posterior desarrollo de las clases del diseño. Además se modelan los artefactos correspondientes al diseño de la propuesta solución.

2.2. Mejoras a los procesos del negocio

De manera general el proceso de Conciliación Bancaria se realizará por parte del sistema, dando la opción al usuario de hacerla manual en caso que existan partidas pendientes después de la ejecución del proceso. Partiendo de los procesos de negocios identificados que se obtuvieron previamente en el trabajo de diploma que antecede a este se determinaron nuevos procesos a petición del cliente ya que de esta forma se optimizarán los mismos. En la referencia 10 se podrán encontrar los procesos de negocios definidos anteriormente.

Procesos de negocio que el sistema permitirá gestionar actualmente:

- Los mensajes MT-950 que llegan a través de la red SWIFT se almacenarán de forma periódica facilitando al Conciliador la posibilidad de gestionarlos.
- Los estados de las cuentas a conciliar generadas por la institución se almacenarán periódicamente y se le dará al Conciliador la posibilidad de gestionarlos sin afectar los movimientos realizados por las áreas operativas.
- El sistema guardará las partidas conciliadas y se permitirá al Conciliador gestionar su información.
- El Conciliador podrá detener y continuar el proceso de conciliación cada vez que lo determine necesario.
- Luego de la conciliación del sistema se muestran las partidas pendientes.
- El Conciliador puede realizar la conciliación de forma manual con las partidas pendientes resultantes.
- Luego de terminada la conciliación el Conciliador puedo ver el cierre de partidas pendientes si así lo desea.

2.3. Definición de Requerimientos Funcionales

Partiendo de un resultado de la Elicitación de Requisitos que se obtuvo previamente en el trabajo de diploma que antecede a este donde se obtuvieron los requerimientos funcionales que debe cumplir el sistema para la gestión de la Conciliación Bancaria en el Banco Nacional de Cuba es necesario un cambio en los mismos a petición del cliente. En la referencia 10 se podrán encontrar los requerimientos funcionales definidos anteriormente.

Requerimientos Funcionales Actuales:

RF.1 Realizar Conciliación de Cuenta.

RF.1.1 Cargar Datos de Cuenta.

RF.1.2 Realizar Conciliación del Sistema.

RF.1.3 Mostrar Partidas Pendientes.

RF.1.4 Realizar Conciliación Manual.

RF. 1.5 Mostrar Conciliación Realizada.

RF.1.6 Cancelar Conciliación.

RF.1.7 Guardar Conciliación.

RF.1.8 Generar Cierre de Conciliación.

RF.1.9 Imprimir Cierre de Conciliación.

RF.2 Gestionar Partidas.

RF.2.1 Mostrar Partidas Conciliadas.

RF.2.2 Mostrar Partidas Pendientes.

RF.2.3 Imprimir Partidas Conciliadas.

RF.2.4 Imprimir Partidas Pendientes.

2.3.1. Definición de actores y casos de uso del sistema

El diagrama de casos de uso consiste en la relación existente entre actores y casos de uso. Los cuales sirven para detallar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios u otros sistemas. Los actores por su parte son una entidad externa al sistema que se modela y que puede interactuar con él. [10].

A continuación los actores definidos para el subsistema Conciliaciones Bancarias.

Actor	Descripción
Conciliador	Usuario del sistema encargado de gestionar los mensajes MT-950 y el Reporte del Estado de "Nuestras Cuentas", realizar además los procesos descritos anteriormente.
Sistema SWIFT	Sistema encargado de recibir los mensajes de Estado de Cuenta MT-950.

Tabla 1. Definición de los actores del sistema.

Los casos de uso son las funciones que proporciona un sistema para añadir valor a sus usuarios. Estos se han adoptado casi universalmente para la captura de requisitos de sistemas de software, sin embargo son más que simplemente una herramienta para la captura de requisitos; sino que dirigen todo el proceso de software. [11].

Partiendo de las descripciones definidas y los prototipos de interfaz de los casos de uso del subsistema Conciliaciones Bancarias propuestos en el trabajo de diploma que antecede a este se muestran a continuación los cambios sufridos a los mismos por decisión del cliente para una mejora y optimización de los procesos. En la referencia 10 se podrán encontrar los casos de uso definidos anteriormente.

Casos de Uso actuales:

- Realizar Conciliación del Sistema.
- Realizar Conciliación Manual.
- Obtener Reporte MT-950.
- Cancelar Conciliación.
- Guardar Conciliación.
- Generar Cierre de Conciliación.

- Imprimir Cierre de Conciliación.
- Mostrar Partidas Conciliadas.
- Mostrar Partidas Pendientes.
- Imprimir Partidas Conciliadas.
- Imprimir Partidas Pendientes.

2.3.2. Modelo de casos de uso del sistema

El modelo de casos de uso representa como ocurren los procesos en el sistema. Consiste en tener una buena organización en cuanto a la relación actor-caso de uso. A continuación se muestra el modelo de casos de uso del subsistema Conciliaciones Bancarias.

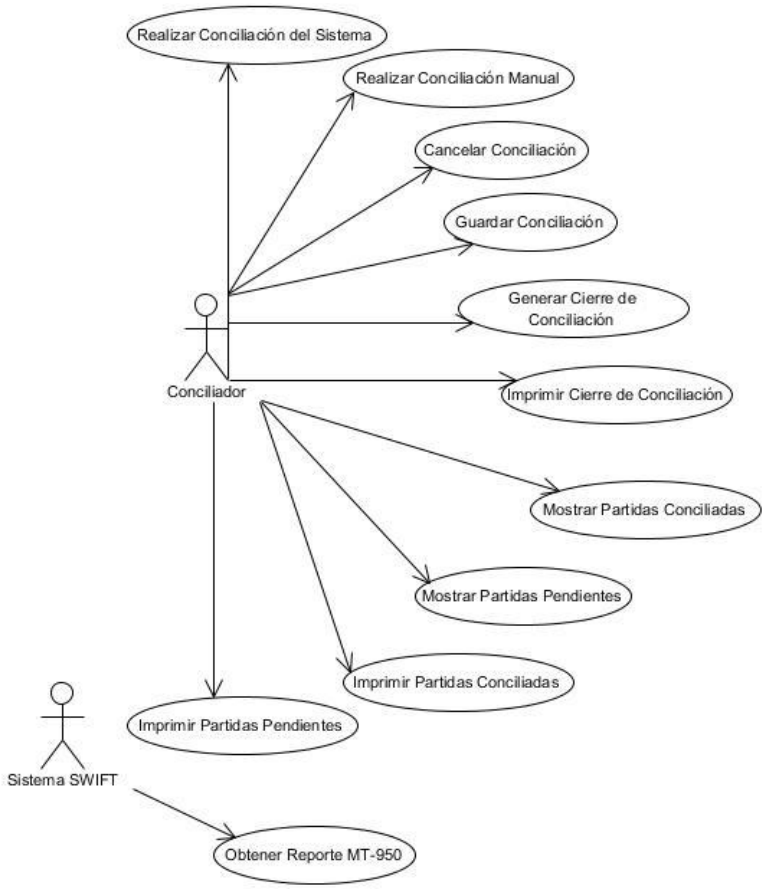


Figura 1: Diagrama de casos de uso del sistema.

2.3.3. Descripción de los casos de uso del sistema

Las descripciones detalladas de los casos de uso del sistema representan una secuencia de pasos a seguir durante la ejecución de un proceso. Seguidamente se muestran las descripciones de los casos de uso más críticos del subsistema Conciliaciones Bancarias.

Caso de usos	Realizar Conciliación del Sistema.
Actor	Conciliador
Resumen	El caso de uso se inicia cuando el Conciliador escoge la opción Realizar Conciliación de Cuenta. El sistema permite realizar una conciliación de forma automática luego de la selección de ciertos campos que se visualizan durante el desarrollo del proceso. El caso de uso finaliza cuando el Conciliador selecciona la opción “Aceptar”.
Precondiciones	La cuenta debe tener movimientos en esa fecha.
Poscondiciones	
Referencia	RF.1.2
Prioridad	Critico
Casos de uso asociados	Cancelar Conciliación, Guardar Conciliación, Generar Cierre de Conciliación.

Flujo Normal de Eventos	
Actor	Respuesta del sistema
1. El Conciliador selecciona la opción “Realizar Conciliación”.	2. El sistema muestra la interfaz para realizar la conciliación. (Anexo 6)
3. El Conciliador selecciona los datos de la cuenta a conciliar y oprime el botón “Aceptar”.	4. El sistema realiza la conciliación con las cuentas que coinciden mostrando solo las partidas pendientes. (Anexo 7)
5. El Conciliador oprime el botón “Aceptar”.	6. El sistema guarda la conciliación y muestra un mensaje “Operación realizada satisfactoriamente”.
	7. Finaliza el caso de uso.
Flujos Alternos	
Sección “”	
Actor	Respuesta del sistema

Tabla 2. Descripción del Caso de Uso Realizar Conciliación del Sistema.

Caso de usos	Realizar Conciliación Manual.
Actor	Conciliador

Resumen	El caso de uso se inicia cuando luego de la conciliación del sistema se muestran las partidas pendientes para que el Conciliador puede realizar la conciliación manual si así lo desea. El caso de uso finaliza cuando el Conciliador selecciona la opción “Conciliar”.
Precondiciones	Deben existir partidas pendientes.
Poscondiciones	
Referencia	RF.1.4
Prioridad	Critico
Casos de uso asociados	Cancelar Conciliación, Guardar Conciliación, Generar Cierre de Conciliación.

Flujo Normal de Eventos	
Actor	Respuesta del sistema
1. El Conciliador selecciona las partidas pendientes a conciliar y oprime el botón “Conciliar”.	2. El sistema concilia las partidas pendientes seleccionadas. (Anexo 8)
	3. Finaliza el caso de uso.
Flujos Alternos	
Sección “Datos Incorrectos”	
Actor	Respuesta del sistema
	1. En caso de que solo se seleccionen partidas en una sola tabla o en ninguna el sistema muestra el mensaje “Debe seleccionar datos en ambas tablas”.
2. El Conciliador acepta el mensaje.	3. El sistema pasa a la acción 1 del flujo normal de eventos.
Sección “Importe Incorrecto”	
Actor	Respuesta del sistema
	4. En caso de que los importes no coincida el sistema muestra el mensaje “El importe no coincide”.
5. El Conciliador acepta el mensaje.	6. El sistema pasa a la acción 1 del flujo normal de eventos.

Tabla 3. Descripción del Caso de Uso Realizar Conciliación de Manual.

Caso de usos	Generar Cierre de Conciliación.
Actor	Conciliador
Resumen	El caso de uso se inicia cuando se termina de realizar la conciliación de

	cuenta y el sistema genera el Cierre de la conciliación permitiéndote ver el mismo si así lo desea el Conciliador.
Precondiciones	Debe realizarse la conciliación de cuenta.
Poscondiciones	
Referencia	RF.1.7
Prioridad	Critico
Casos de uso asociados	Realizar Conciliación del Sistema, Realizar Conciliación Manual.
Flujo Normal de Eventos	
Actor	Respuesta del sistema
	1. Luego de realizada la conciliación el sistema muestra un mensaje preguntado si desea ver el cierre de la conciliación.
2. El Conciliador selecciona la opción (Si-No).	3. En el caso de que sea “Si” se muestra el Cierre de la Conciliación con los datos pertinentes en el otro caso el sistema retorna a la página principal del subsistema. (Anexo 5)
	4. Finaliza el caso de uso.
Flujos Alternos	
Sección “”	
Actor	Respuesta del sistema

Tabla 4. Descripción del Caso de Uso Mostrar Cierre de Conciliación.

2.4. Análisis

Durante el análisis, se analizan los requisitos que se describen en la captura de requerimientos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, incluyendo su arquitectura. [10].

2.4.1. Definición del modelo de paquetes

Para lograr una organización estructural en el desarrollo del subsistema Conciliaciones Bancarias, se definió un modelo de paquetes, permitiendo así agrupar los modelos de gran tamaño en subconjuntos más manuables. Los paquetes son utilizados como contenedores de elementos, donde cada elemento

puede estar contenido dentro de un único paquete, sin embargo un paquete puede contener otro conjunto de paquetes. El subsistema Conciliaciones Bancarias está estructurado en 2 módulos.

- Módulo Conciliación de Cuenta
- Módulo Gestionar Partidas

A continuación se presenta el modelo de paquetes diseñado con el objetivo de definir la estructuración y dependencia entre los módulos que conforman el subsistema Conciliaciones Bancarias.

Figura 2: Modelo de Paquetes del subsistema Conciliaciones Bancarias.

2.4.2. Definición del modelo de clases del análisis

El Diagrama de clases del análisis es un artefacto que permite representar los conceptos en un dominio del problema. Durante su construcción se identifican las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. [10].

A continuación se presenta el diagrama de clases de análisis para el caso de uso Realizar Conciliación del Sistema del módulo Conciliación de Cuenta, los modelos que corresponden a los demás caso de usos se podrán encontrar en los anexos correspondientes a las clases del análisis.

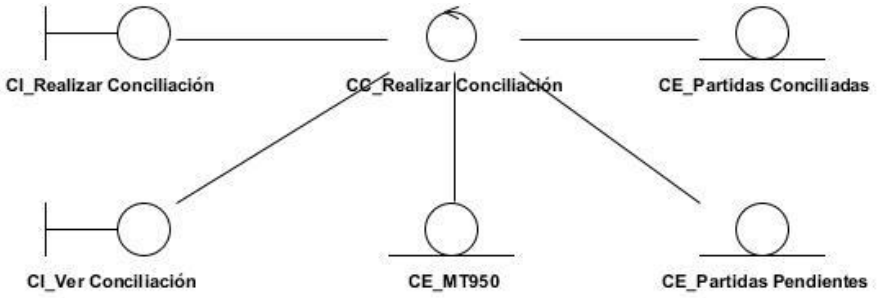


Figura 3: Modelo de Análisis del caso de uso Realizar Conciliación del Sistema.

2.4.3. Fundamentación de la arquitectura

Los módulos y/o componentes del sistema por su parte estarán separados por diferentes capas lógicas según la naturaleza de los mismos. [10].

Las capas lógicas definidas para el sistema se muestran a continuación:

- Capa de Presentación

En esta capa se desarrollará la lógica de presentación. En el servidor se utilizará Spring MVC para recibir, controlar y enviar una respuesta a las peticiones realizadas desde el cliente. En el cliente se utilizará la librería Dojo para generar las interfaces que interactuarán con el usuario. La capa de presentación estará relacionada con la Capa de Negocios y Capa de Dominio.

- Capa de Negocio

Esta capa estará divida en dos subcapas. Estas subcapas son Fachada y Manager. La Fachada será el punto de intercambio entre la capa de presentación y la capa de negocio. La subcapa Fachada delegará a la subcapa Manager la realización de la lógica del negocio. Por otro lado la subcapa Manager tendrá la jerarquía de clases suficiente para implementar el negocio de la aplicación. Esta subcapa utilizará la capa de acceso a datos para obtener datos que están persistidos y la capa de dominio para generar los objetos de dominios.

- Capa de acceso a datos

En esta capa se realizarán todas las operaciones relacionadas con el gestor de base de datos y cualquier recurso que contenga información persistida.

- Capa de Dominio

En esta última se declararán todas las clases que representan entidades del negocio. Dichas clases de dominio estarán presentes en todas las capas anteriormente descritas.

2.5. Diseño

El diseño constituye un enfoque de la ingeniería de software en el que se modela un sistema como un grupo de objetos que interactúan entre sí, representando un dominio en términos de conceptos clasificados de acuerdo a su dependencia funcional. A la vez que se realiza el proceso de diseño se describen los aspectos del sistema a desarrollar. [10].

2.5.1. Diagrama de paquetes

Durante el desarrollo de software resulta muy conveniente agrupar clases y ficheros por diferentes criterios que ayudarán a una fácil comprensión de la aplicación, resultando conveniente el desarrollo de los diagramas de paquetes, los cuales muestran como está dividido el sistema en agrupaciones lógicas mostrando las dependencias entre estas. [14].

A continuación se muestra la estructura y dependencia de paquetes del módulo Conciliación de Cuenta y una explicación de la composición de cada uno. El diagrama de paquetes que comprende el módulo Gestionar Partidas se comporta de manera similar.

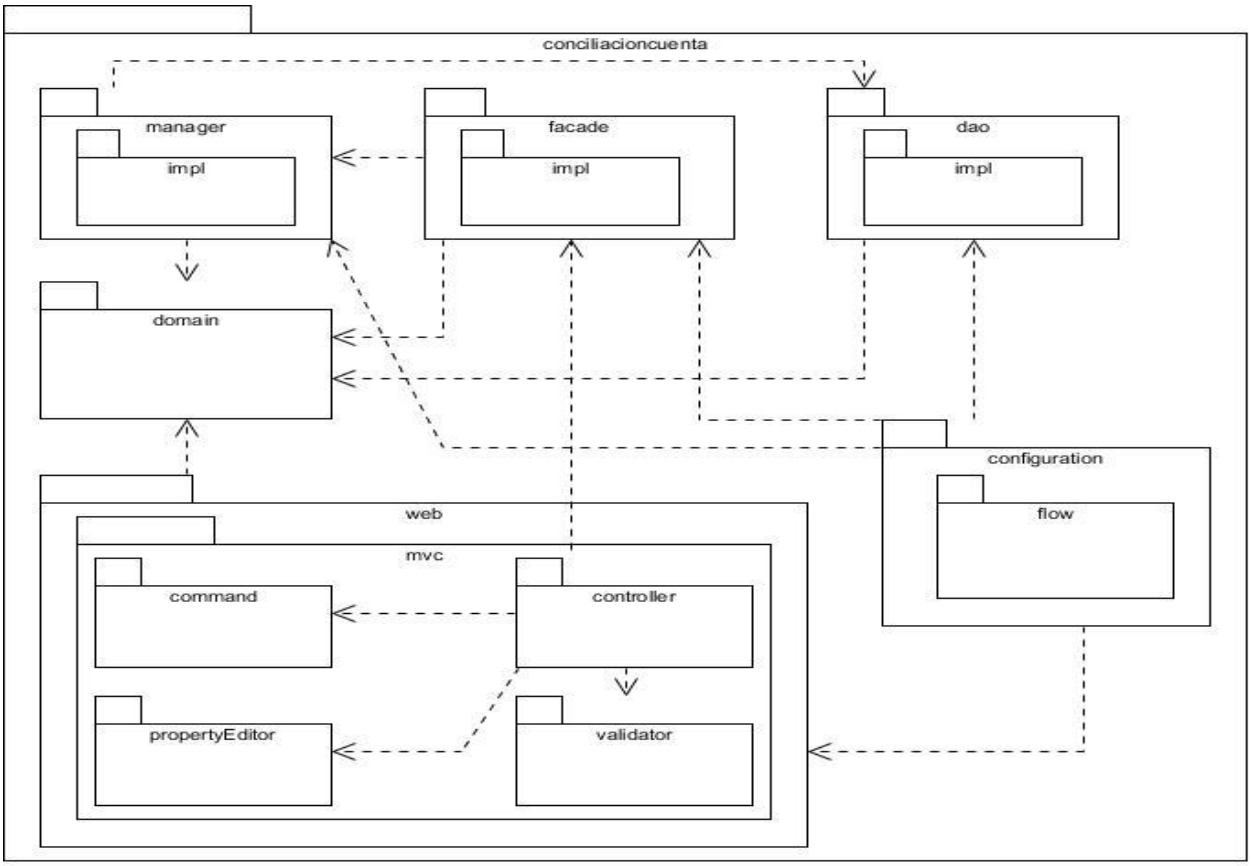


Figura 4: Diagrama de paquetes del módulo Conciliación de Cuenta.

Paquete *configuration*: En este paquete se encuentran los ficheros de configuración en formato XML de los diferentes contextos de Spring para el módulo, son estos:

- -servlet.xml: Define el contexto de Spring MVC.
- -bussiness.xml: Define el contexto para el negocio.
- -webflow.xml: Define el contexto para Spring WebFlow.
- -dataaccess.xml: Define el contexto para acceso a datos.

Paquete *facade*: En este paquete se encuentran la interfaz y su respectiva implementación, encargadas de brindar las funcionalidades que serán usadas por la presentación, así como las que se consumen por otros módulos.

Paquete *manager*: En este paquete se hallan las interfaces y las implementaciones que comprenden la lógica de negocio que existe en el módulo, que será brindada a las capas superiores.

Paquete *dao*: En el paquete dao se encuentran igualmente las interfaces y las implementaciones encargadas de brindar los servicios de comunicación con la base de datos, además de hallarse también los ficheros de mapeo utilizados por Hibernate correspondientes al módulo.

Paquete *domain*: Aquí se localizan las clases relacionadas con el dominio del módulo en cuestión.

Paquete *web*: El paquete web agrupa un conjunto de clases y paquetes que son los encargados de realizar todo lo referente a la presentación en el lado del servidor.

Paquete *mvc*: En este subpaquete se agrupa todo lo referente a la lógica de presentación cuando se hace uso de SpringMVC.

Paquete *controller*: En este paquete se hallan las clases que extienden de los controladores propuestos por Spring MVC, encargadas de responder a las peticiones realizadas por el cliente.

Paquete *command*: Se encuentran las clases que representan los datos que son introducidos y mostrados en los formularios de la interfaz de usuario.

Paquete *propertyEditor*: Agrupa las clases que actúan como intermediarias para convertir datos de la interfaz de usuario en objetos y viceversa.

Paquete *validator*: Cuenta con las clases encargadas de realizar la validación de datos en el lado del servidor.

2.5.2. Patrones de Diseño

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. Es decir: “Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.”[12].

- **Patrones de Asignación de Responsabilidades (GRAPS)**[10]

[10]General Responsibility Assignment Software Patterns: son patrones generales de software para asignación de responsabilidades, aunque se considera que más que patrones, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

Patrón Alta Cohesión

Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos.

En la Anexo 9 se puede observar la utilización del patrón alta cohesión en el subsistema Conciliaciones Bancarias, se puede observar como la clase ConciliacionFacade es la que controla la información de otras clases de las cuales consume recurso, pues ella es la encargada de relacionarse con las mismas, evitando así que el controlador se sobrecargue y su cohesión con la fachada siga siendo alta.

Patrón Bajo Acoplamiento

La característica principal de este patrón es mantener las clases más independientes entre sí y con la menor cantidad de relaciones; la cual posibilita que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización y disminuyendo la dependencia entre las clases, asignándoles una responsabilidad para mantener un bajo acoplamiento.

El bajo acoplamiento se puede ver evidenciado en el Anexo 9, ya que el controlador MT950MultiActionController se mantiene independiente de las relaciones con otras clases pues utiliza a la clase ConciliacionFacade para que ella sea la encargada de dichas relaciones, evitando cualquier posible repercusión que pueda tener la modificación de alguna de estas clases.

Patrón Experto

La creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtendrá un diseño con mayor cohesión y así la información se mantendrá encapsulada (disminución del acoplamiento).

Algunos ejemplos donde se evidencia este patrón en el subsistema Conciliaciones Bancarias son:

- La clase MultiActionController que es el experto en controlar las peticiones desde la vista hacia el negocio.

- La clase Facade que es el experto en crear las instancias de las clases manager donde se realiza la lógica del negocio.
- La clase Manager que es el experto en crear las instancias de las clases DAO para consumir las funcionalidades que dependen de la base de datos.
- La clase DAO que es el experto en la comunicación con la base de datos.

Patrón Creador

La característica principal de este patrón es que permite identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases. Asignarle a la clase B la responsabilidad de crear una instancia de la clase A.

Con la utilización del framework Spring MVC en el subsistema de Conciliaciones Banacarias se implementa el patrón creador el cual está unido a dicho framework. La creación de instancias de otras clases deja de ser responsabilidad de las clases en capa superiores y pasa a ser realizada por las clases de configuración de los módulos propuestos por Spring, tal es el caso de la clase "finixubnc-conciliacion-common-business.xml" la cual al crear el bean[11] gestionarMT950Facade crea dentro de ella una instancia de "mt950Manager", por lo que al crease un objeto de la clase MT950Manager en la clase GestionarMT950Facade no es necesario que se inicialice sus valores pues Spring por medio de los beans creados se encarga de ello.

Patrón Controlador

La característica principal de este patrón es que posibilita la asignación de responsabilidades para controlar el flujo de eventos del sistema, a clases específicas. Sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario enviándolo a las distintas clases según el método llamado.

Las clases controladoras GestionarPartidasMultiActionContoller y MT950MultiActionController constituyen ejemplos de la aplicación de este patrón, las cuales tendrá en cuenta la responsabilidad de manejar los eventos que consisten en gestionar las partidas en el subsistema.

[11]Bean: Componente del software que tiene la particularidad de ser reutilizable y así evitar la tarea de programar los distintos componentes uno a uno.

- **Patrones Estructurales**

Los patrones estructurales están relacionados con cómo las clases y los objetos se combinan para dar lugar a estructuras más complejas y describen las formas de componer los objetos para conseguir una nueva funcionalidad. [12].

Facade (Fachada)

La característica principal de este patrón es que proporciona una interfaz unificada a un conjunto de interfaces de un sistema. Facade define una interfaz de alto nivel, posibilitando que el sistema sea más fácil de usar.

Un ejemplo de cómo se evidencia la utilización del patrón Fachada en el Subsistema Conciliación se encuentra en el Anexo 9 donde se puede ver como en el módulo conciliacioncuenta donde el controlador MT950MultiActionController controla las peticiones del usuario, realizando la conciliación, para ello necesita cargar una serie de datos ubicados en diferentes módulos dentro y fuera del subsistema de Conciliación. Los métodos ListarBancos del Subsistema Contabilidad, el método ListarMonedas del módulo common, el método ListarPartidasPendientes del módulo gestionarPartidas y el método ListarCuentas del propio módulo son métodos que son invocados por el controlador y por lo que el mismo se auxilia de la fachada ConciliacionFacade, que a su vez invoca a las fachadas GestionarMT950Facade, GestionarPartidasFacade, GlobalFacade y al manager ConciliacionManager de los cuales consume los métodos necesarios invocados por el controlador ,evitando así que dicho controlador sobrecargue su funcionamiento en el trabajo con estas fachadas.

Cadena de Responsabilidad

Cuando desde la vista se solicita información que se encuentra en la base de datos, esta es atendida primeramente por los Controller, luego por la Facade, el Manager y finalmente el DAO, evidenciándose de esta manera la utilización de dicho patrón.

- **Patrones de Comportamiento**

Estos patrones se dedican a estudiar las relaciones de llamadas entre los diferentes objetos, proporcionan estrategias comprobadas para modelar la manera en que los objetos colaboran entre sí en un sistema. [12].

Patrón de Acceso a Datos (DAO)

La característica principal de este patrón es que centraliza todo el acceso a datos en una capa independiente, aislándolo del resto de la aplicación. Desacoplando la lógica de negocio de la lógica de acceso a datos, de manera que se pueda cambiar la fuente de datos fácilmente. 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.

- **Patrones de Presentación**

MVC (Modelo Vista Controlador)

El subsistema Conciliaciones Bancarias utiliza a SpringMVC como framework núcleo de la aplicación, lo cual obliga al mismo a utilizar el patrón MVC para su correcto funcionamiento. Un ejemplo del funcionamiento de este patrón en el subsistema (Anexo 10) se evidencia en la utilización de la funcionalidad Listar Partidas Conciliadas, la cual para un correcto funcionamiento utiliza la vista partidasConciliadas realizando la petición necesaria al controlador GestionarPartidasMultiactionController. Este emplea una estructura de negocio compuesta por las clases GestionarPartidasFacade, GestionarPartidasManager y PartidasConciliadasDAO con los métodos pertinentes para responder a dicha petición, usando el modelo PartidasConciliadas para gestionar una correcta respuesta a la funcionalidad antes mencionada.

2.5.3. Diagrama de clases del diseño

El diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Es el modelo de objeto utilizado durante el proceso de análisis y diseño de un sistema informático, en el cual se crea el diseño conceptual de la información que se manejará en el sistema. Sirve además como una entrada fundamental de las actividades de implementación. [10].

A continuación se muestra el modelo de diseño para el módulo Conciliación de Cuenta, el resto de los diagramas se podrán encontrar en los anexos correspondientes a los diagramas de clases del diseño.

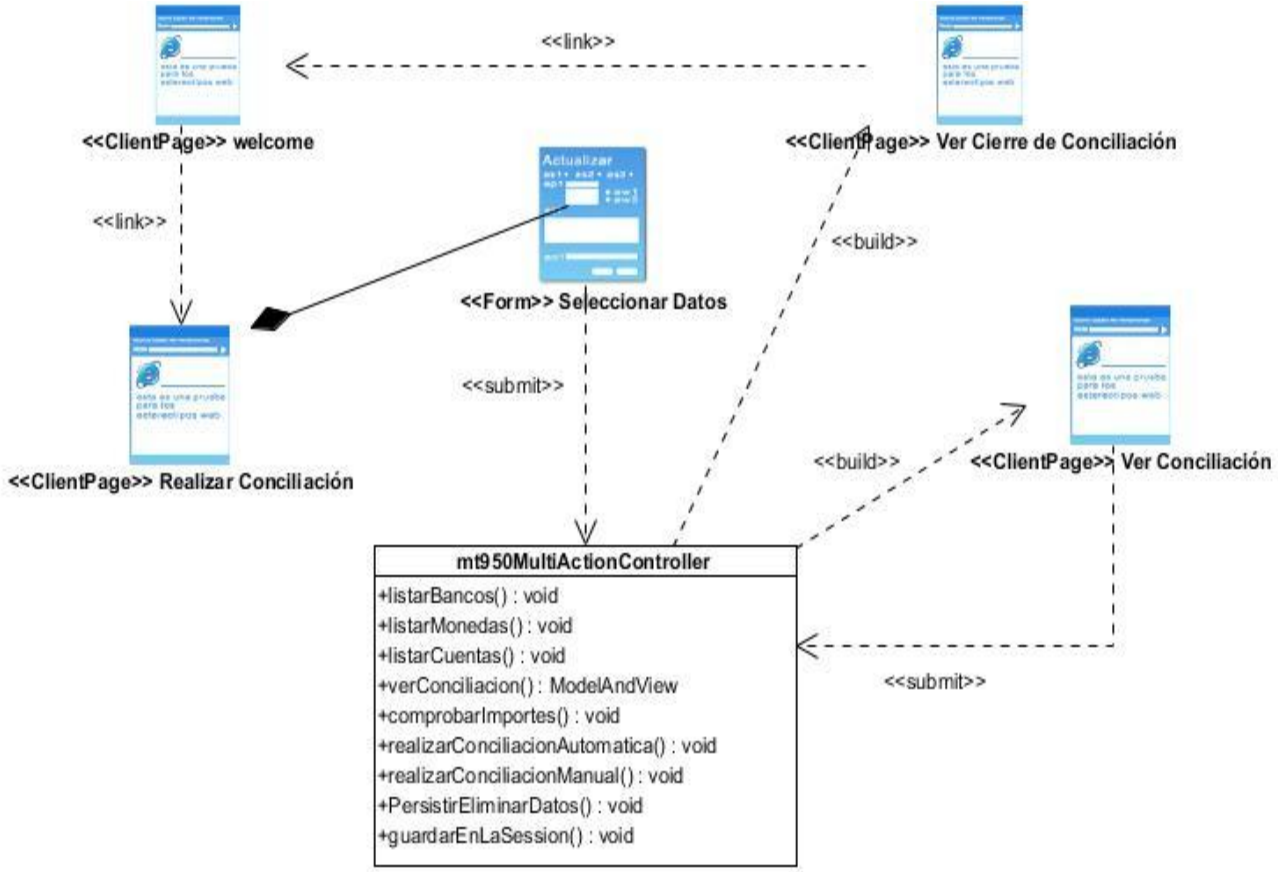


Figura 5: Diagrama de clase de diseño (Capa de presentación) del módulo Conciliación de Cuenta.

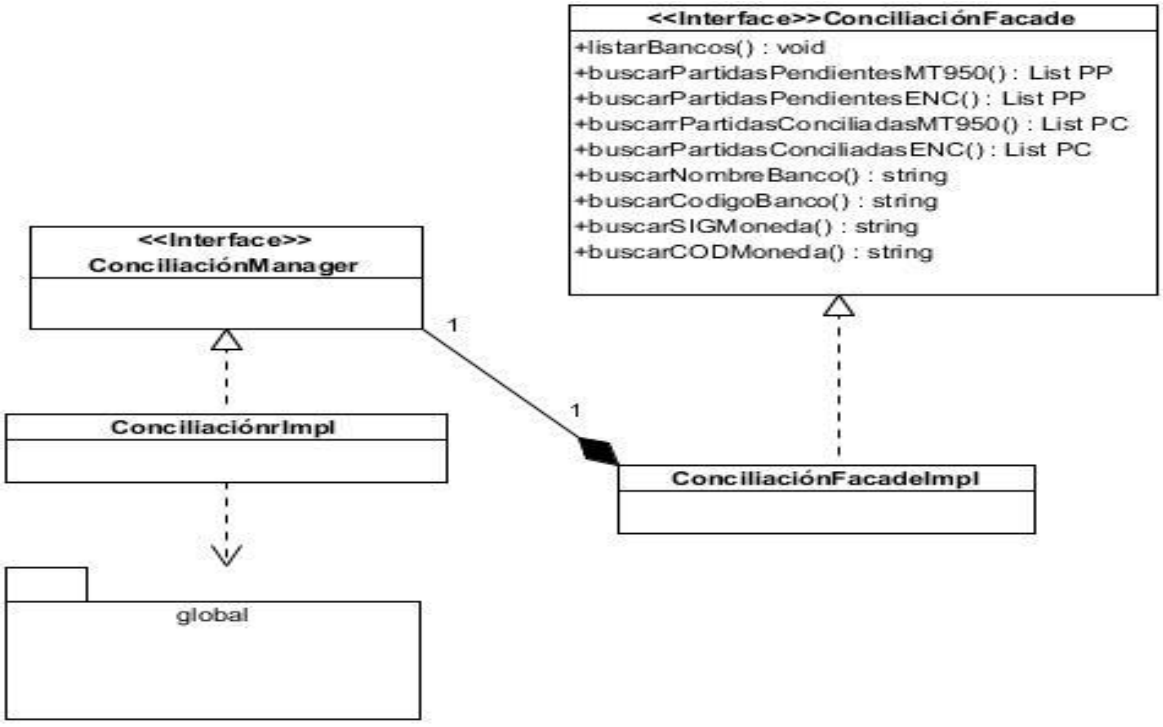


Figura 6: Diagrama de clase de diseño (Capa de negocio) del módulo Conciliación de Cuenta.

2.5.4. Diagrama de interacción del diseño

Los diagramas de interacción son utilizados para modelar los aspectos dinámicos de los sistemas, muestran la realización de un flujo o escenario concreto de un caso de uso en términos de interacción entre objetos del diseño. Se les atribuye una gran importancia debido a la compresión del sistema a la que puede llegarse a través de ellos. [14].

Debido a esto se realizaron diagramas de secuencia, que son un ejemplo de diagramas de interacción que destacan principalmente el orden temporal de los mensajes para varios escenarios de los subsistemas en cuestión, a continuación se presenta el diagrama de secuencia asociados al escenario Realizar Conciliación del Sistema.

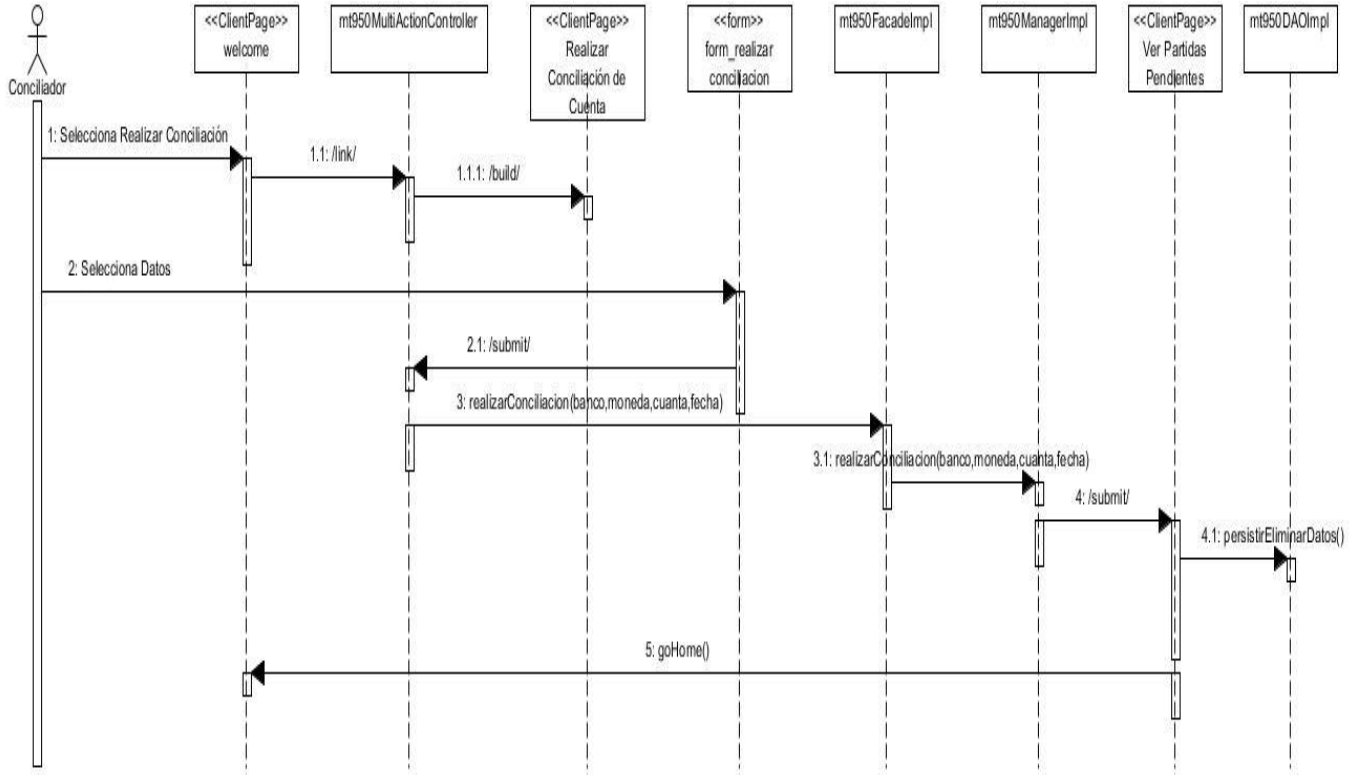


Figura 7: Diagrama de interacción del escenario Realizar Conciliación del Sistema.

2.5.5. 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. [10].

Es necesario resaltar que el subsistema Conciliaciones Bancarias forma parte de un sistema que ya posee una base de datos, sin embargo sólo se muestra el modelo de datos que responde al subsistema en cuestión.

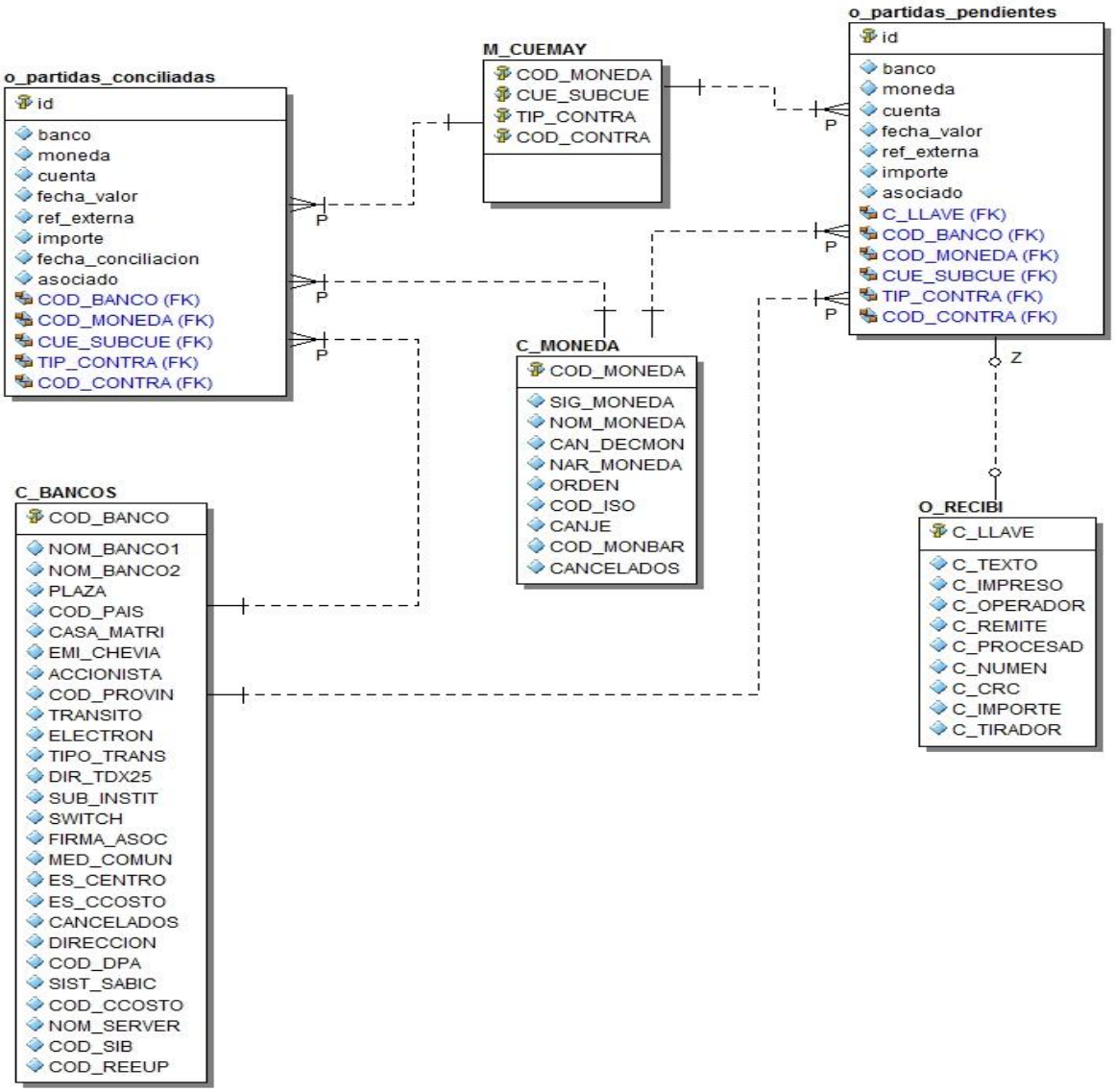


Figura 8: Modelo de Datos del subsistema Conciliaciones Bancarias.

2.6. Modelo de despliegue

Es un modelo detallado de los componentes que se desplegarán a lo largo de la infraestructura del sistema. Detalla las capacidades de red, las especificaciones del servidor, los requisitos de hardware y otra información relacionada al despliegue del sistema propuesto. Es utilizado para visualizar la distribución de los componentes en los nodos físicos. [13].

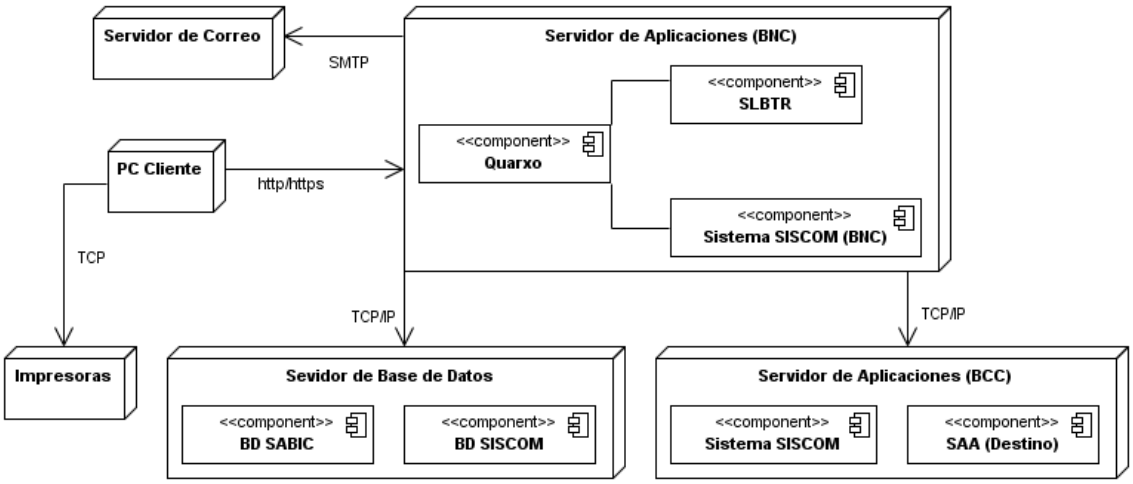


Figura 9: Diagrama de despliegue de QUARXO.

2.7. 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).

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. [10].

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.

La métrica TOC posibilita medir los siguientes atributos de calidad:

- Responsabilidad: Responsabilidad que posee una clase en un marco conceptual correspondiente al modelado de la solución propuesta.
- 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.

En la métrica TOC los atributos de calidad Responsabilidad y Complejidad de implementación son inversamente proporcionales a la Reutilización, lo que puede ser traducido como, mientras mayor sea la Responsabilidad y Complejidad de implementación de una clase, menor será su nivel de Reutilización.

Resultados de la evaluación de la métrica TOC

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente:

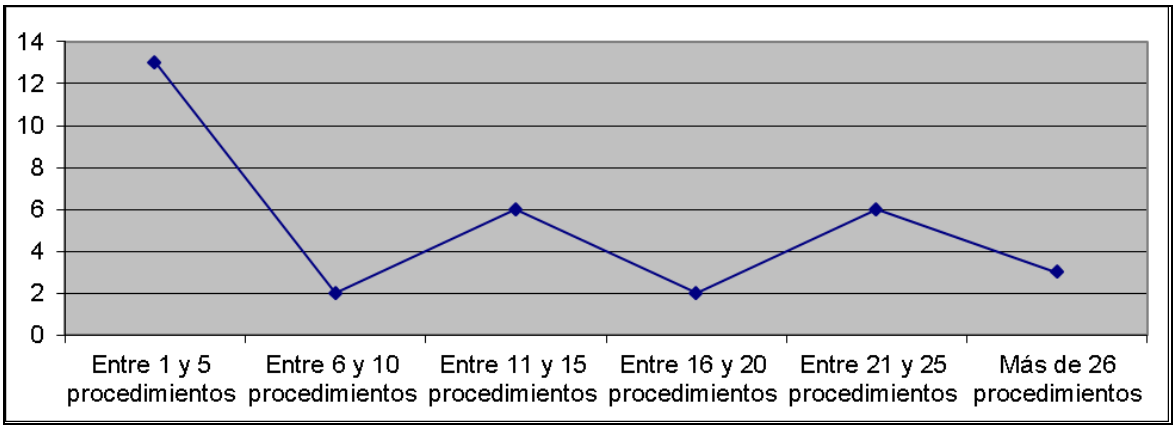


Figura 10: Representación de la cantidad de clases y el número de procedimientos que contienen.

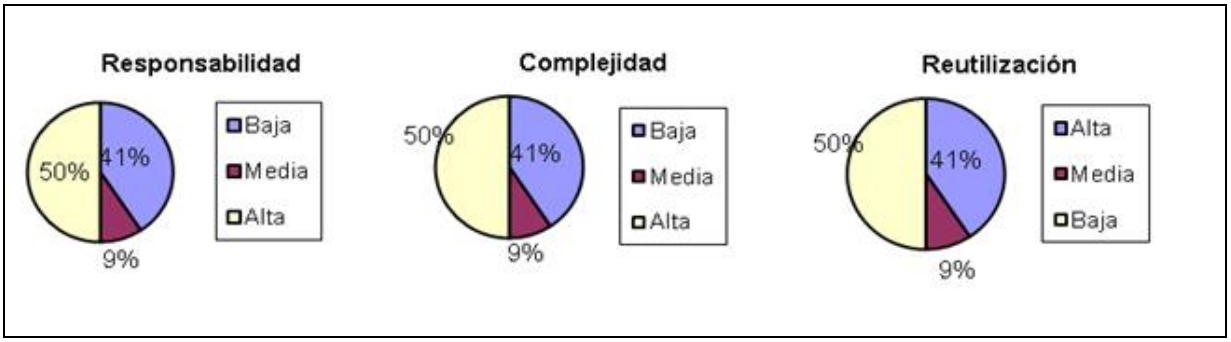


Figura 11: Representación en % de las variables Responsabilidad, Complejidad y Reutilización.

Análisis de los resultados obtenidos en la evaluación de la métrica TOC

Luego de aplicar la métrica TOC, se puede concluir que el diseño propuesto para el subsistema Conciliaciones Bancarias está entre los límites aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio en la mayoría de las clases; de manera que se puede observar cómo

se promueve la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación, lo que promueve el Bajo Acoplamiento entre las clases y facilitará la implementación y la comprobación al desarrollador.

Métrica Relaciones entre clases (RC)

Se refiere al número de relaciones de uso de una clase. En la métrica RC los atributos Acoplamiento, Complejidad de mantenimiento y Cantidad de pruebas son inversamente proporcionales a la Reutilización, es decir mientras mayor sea el Acoplamiento, Complejidad de mantenimiento de una clase y Cantidad de pruebas, menor será su nivel de Reutilización.

La métrica RC posibilita medir los siguientes atributos de calidad:

- Acoplamiento: Dependencia o interconexión de una clase o estructura de clase respecto a otras.
- 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.
- Reutilización: Significa cuán reutilizada es una clase o estructura de clase dentro de un diseño de software.
- Cantidad de pruebas: Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

Resultados de la evaluación de la métrica TOC

Como resultado de la evaluación de la métrica TOC se obtuvo lo siguiente:

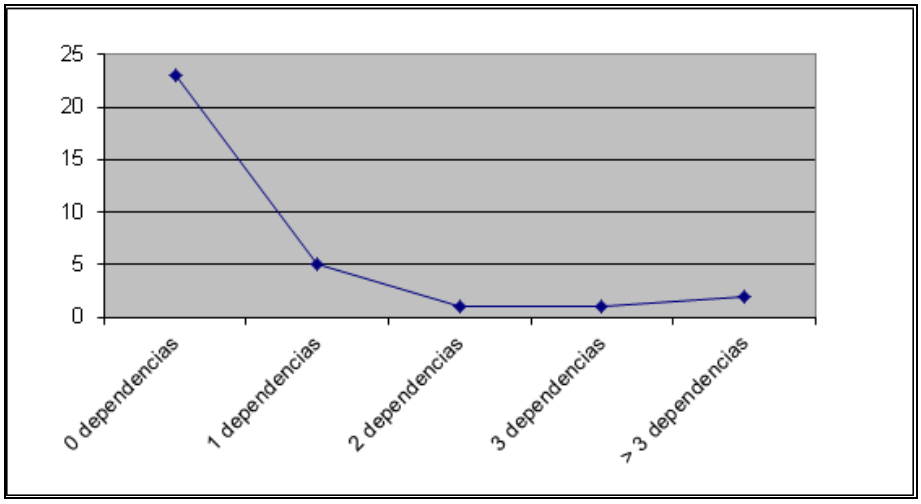


Figura 12: Representación de las asociaciones de uso por cantidad de clases.

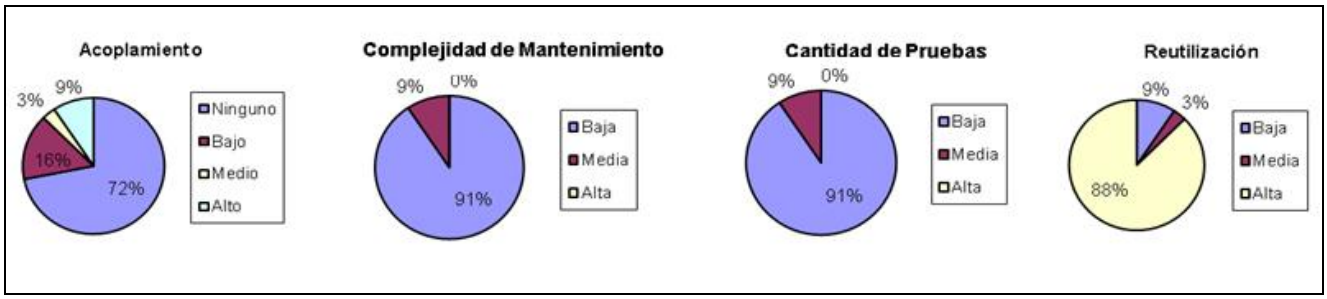


Figura 13: Representación en % de las variables Acoplamiento, Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización.

Análisis de los resultados obtenidos en la evaluación de la métrica RC

Mediante la evaluación de la métrica RC se puede deducir que el diseño del subsistema tiene una calidad aceptable teniendo en cuenta que el 72% de las clases presentes en el diseño cuentan con una baja cantidad de relaciones de uso. Como se puede observar en las gráficas anteriores las clases promueven un bajo nivel de acoplamiento, además la complejidad de mantenimiento y la cantidad de pruebas que se realizarán serán pocas. En consecuencia a esto el grado de reusabilidad es mayor.

En sentido general el resultado de la aplicación de estas métricas demuestra que las clases no se encuentran muy sobrecargadas en responsabilidad, existe además bajo acoplamiento entre las mismas y presentan un alto nivel de reutilización. Indican que el diseño no es complejo, pues la complejidad de mantenimiento es baja, así como la complejidad en las pruebas.

2.8. Conclusiones parciales

En este capítulo se modeló el sistema quedando definido actores, casos de uso así como las especificaciones de cada caso de uso. Durante la etapa de diseño se tuvo en cuenta algunos patrones para modelar el sistema a desarrollar, obteniendo así los artefactos correspondientes al diseño, entre los cuales se encuentran diagramas de secuencia, diagramas de clases del diseño y el modelo de datos, que en conjunto con lo antes planteado permitieron dar paso a la implementación de la solución. Además se aplicaron las métricas TOC y RC para la evaluación del diseño donde se obtuvieron buenos resultados demostrando que el diseño propuesto presenta una buena calidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

El último capítulo del presente trabajo de diploma estará enfocado a dar cumplimiento a dos flujos de mucha importancia en el proceso de desarrollo de software. Primeramente se realizará la implementación del subsistema Conciliación, donde se representará la interacción de los componentes mediante el Diagrama de Componentes y serán detallados los métodos y atributos de las clases más importantes del modelo de diseño presentado. Posteriormente en el capítulo se dará paso a la validación de la solución mediante la aplicación de las pruebas de unidad y funcional en pos de eliminar posibles errores que impidan el cumplimiento de los requisitos funcionales y las perspectivas del cliente.

3.2. Implementación

La implementación constituye uno de los flujos de trabajo más importantes propuestos por RUP, en ella se toma como punto de partida lo arrojado como resultado en el diseño y se implementa el sistema en términos de componentes como ficheros de código binario, código fuente, scripts, ejecutables, entre otros. Su importancia se debe a que se obtiene como consecuencia un sistema ejecutable, siendo uno de los principales objetivos en el desarrollo de software. [14].

3.2.1. Estándares de codificación

Los estándares de codificación se definen por el equipo de desarrollo para lograr estandarización en la programación 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. [14].

3.2.1.1. Convenciones de nomenclatura

La nomenclatura de las clases está puntualizada por la utilización de la notación Pascal Casing, la cual define que los nombre e identificadores pueden estar compuestos por múltiples palabras juntas y la primera letra de cada palabra irá siempre en mayúsculas y se obvia el uso de artículos. [14].

Ejemplo: Mt950Manager. En este ejemplo el nombre de clase está compuesto por 2 palabras iniciadas cada una con letra mayúscula.

Para el nombrado de las clases hay que tener en cuenta el rol que esta desempeñan en el sistema. A continuación se presentan las nomenclaturas organizadas por los paquetes a los que pertenecen las clases.

controller: Las clases incluidas en este paquete, después del nombre se le incorpora el nombre del controlador de Spring del cual hereda. Ejemplo: Mt950MultiActionController.

command: Las clases que se ubican en este paquete se nombran con el nombre de la clase más la palabra Command. Ejemplo: Mt950Command.

facade: Las clases que se ubican en este paquete se nombran con el nombre de la clase más la palabra Facade y dentro del subpaquete impl tendrán el mismo nombre anterior más la palabra Impl. Ejemplo: Mt950Facade, Mt950FacadeImpl.

manager: Las clases que se ubican en este paquete se nombran con el nombre de la clase más la palabra Manager y dentro del subpaquete impl tendrán el mismo nombre anterior más la palabra Impl. Ejemplo: Mt950Manager, Mt950ManagerImpl.

dao: Las clases que se ubican en este paquete se nombran con el nombre de la clase más la palabra DAO y dentro del subpaquete impl tendrán el mismo nombre anterior más la palabra Impl. Ejemplo: Mt950DAO, Mt950DAOImpl.

En general el nombre de los métodos y los atributos de las clases se escriben con la inicial del identificador en minúscula, en caso de ser un nombre compuesto se utilizará la notación Camel Casing, que es muy equivalente a la Pascal Casing con la excepción de que la letra inicial comienza con minúsculas. Al igual se le aplica esta notación a los nombres de ficheros de código JavaScript y sus funciones y variables internas.

3.2.2. 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. [14].

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

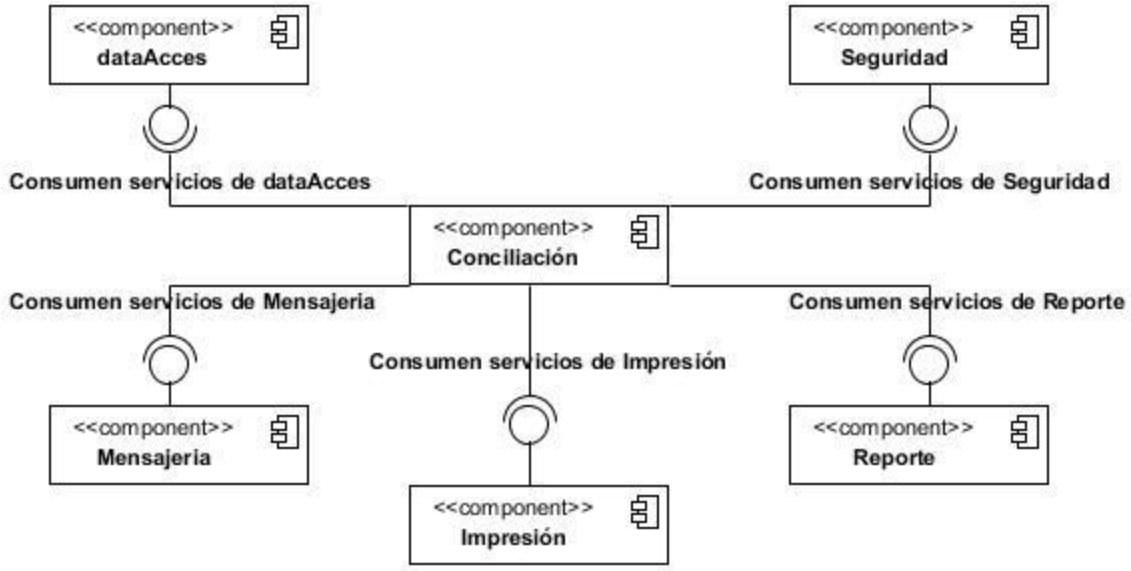


Figura 14: Modelo de componentes.

A continuación se explican de manera general cada uno de los 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.

3.2.3. Descripción de las clases y las funcionalidades

Teniendo en cuenta el diseño de las clases correspondientes al subsistema Conciliaciones Bancarias realizado anteriormente, a continuación serán descritos los atributos y métodos de aquellas que son más importantes para el sistema desde el punto de vista funcional.

Nombre: GestionarPartidasMultiActionController	
Tipo de clase: Controladora	
Atributo	**Tipo**
partidasFacade	GestionarPartidasFacade
Para cada responsabilidad:	
Nombre:	**Descripción**:
listarBancos(HttpServletRequest request, HttpServletResponse esponse)	Se encarga de listar todos los Bancos con los cuales el BNC tiene algún tipo de operación financiera.
buscarPartidasPendientes(HttpServletRequest request, HttpServletResponse response)	Se encarga de buscar en la base de datos las Partidas Pendientes tanto de los mensajes 950 como de los estados de las cuentas bancarias que concuerden con los parámetros que se envían y mostrarlas en una nueva interfaz.

listarPartidasPendientesMT950(HttpServletRequest request, HttpServletResponse response)	Se encarga de buscar en la base de datos las Partidas Pendientes de los mensajes 950 que concuerden con los parámetros que se envían.
listarPartidasPendientesENC(HttpServletRequest request, HttpServletResponse response)	Se encarga de buscar en la base de datos las Partidas Pendientes de los estados de las cuentas bancarias que concuerden con los parámetros que se envían.
buscarPartidasConciliadas(HttpServletRequest request, HttpServletResponse response)	Se encarga de buscar en la base de datos las Partidas Conciliadas tanto de los mensajes 950 como de los estados de las cuentas bancarias que concuerden con los parámetros que se envían y mostrarlas en una nueva interfaz.
listarPartidasConciliadasMT950(HttpServletRequest request, HttpServletResponse response)	Se encarga de buscar en la base de datos las Partidas Conciliadas de los mensajes 950 que concuerden con los parámetros que se envían.
listarPartidasConciliadasENC(HttpServletRequest request, HttpServletResponse response)	Se encarga de buscar en la base de datos las Partidas Conciliadas de los estados de las cuentas bancarias que concuerden con los parámetros que se envían.
imprimirPartidasC (HttpServletRequest request, HttpServletResponse response)	Se encarga de imprimir los reportes correspondientes a las Partidas Conciliadas.
imprimirPartidasP (HttpServletRequest request, HttpServletResponse response)	Se encarga de imprimir los reportes correspondientes a las Partidas Pendientes.

Tabla 5. Descripción de la clase GestionarPartidasMultiActionController.

Nombre: Mt950ManagerImpl

Tipo de clase: Manager	
Atributo	**Tipo**
dao	Mt950DAO
procedure	EstadoNuestrasCuentasDaoStoreProcedure
imprimirCierre	ImprimirCierre
globalFacade	GlobalFacade
Para cada responsabilidad:	
Nombre:	**Descripción**:
listarMonedas()	Se obtienen todas las monedas con las que se pueden registrar las cuentas.
buscarNombreBanco(String codBanco)	Se obtiene el nombre del banco según el código del mismo.
buscarPPendientes950(final String banco,final String moneda,final String cuenta)	Se encarga de buscar todas las Partidas Pendientes de los mensajes 950 que concuerden con los parámetros de búsqueda.
buscarPPendientesENC(final String banco,final String moneda,final String cuenta)	Se encarga de buscar todas las Partidas Pendientes de los reportes de estado de las cuentas bancarias que concuerden con los parámetros de búsqueda.
buscarCodigoBanco(final String nomBanco)	Se obtiene el código con que se identifica el banco cuyo nombre concuerde con el parámetro enviado
buscarSIGMoneda(String codMoneda)	Se obtiene la sigla de la moneda según el código de la misma.
buscarPPendientesSWIFT(String banco,String moneda, String cuenta)	Se encarga de obtener el reporte de partidas pendientes a través de SWIFT.
listarCuentas(String banco, String moneda)	Se obtienen todas las cuentas pertenecientes a un banco y una moneda específica.
buscarCodigoMoneda(String nomMoneda)	Se obtiene el código de una moneda según el

	nombre de la misma.
obtenerEstadoNC(Date fecha_ini, Date fecha_fin, String banco,String moneda)	Se obtiene el reporte Estado de "Nuetras Cuentas" desde el procedimiento que lo ejecuta.
listarBancos()	Se obtienen todos los bancos.
exportReportToPdfCierre(String fileName, List<String> lista,Map parametros)	Se utiliza para mostrar el cierre.

Tabla 6. Descripción de la clase Mt950ManagerImpl.

Nombre: PartidasConciliadasDAOImpl	
Tipo de clase: Data Access Object	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción**:
buscarPartidasConciliadasMT950(final String banco,final Date fechaI,final Date fechaF)	Se encarga de buscar todas las Partidas Conciliadas de los mensajes 950 que concuerden con los parámetros de búsqueda.
buscarPartidasConciliadasENC(final String banco,final Date fechaI,final Date fechaF)	Se encarga de buscar todas las Partidas Conciliadas de los reportes de estado de las cuentas bancarias que concuerden con los parámetros de búsqueda.

Tabla 7. Descripción de la clase PartidasConciliadasDAOImpl.

3.3. Validación de la implementación de los subsistemas

Durante el proceso de desarrollo de software las posibilidades de errores son múltiples por lo que se hace necesario detectar a tiempo las imperfecciones e irregularidades del mismo, además de proporcionar una visión objetiva de la madurez y calidad de los procesos asociados.

Posterior al flujo de Trabajo de Implementación definido en la metodología RUP, se encuentra el de Prueba, el cual es imprescindible para verificar si el sistema cumple con los requisitos propuestos por el cliente.

El aspecto fundamental que rige esta etapa es determinar mediante las pruebas, cómo y en qué sentido el subsistema Conciliaciones Bancarias cumple con las expectativas del cliente, a partir de los requisitos establecidos y las restricciones impuestas. En ese ámbito se trazan un conjunto de objetivos dentro de los que se sitúan:

- Verificar la implementación del subsistema.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar los errores y asegurar que estos sean corregidos de la mejor manera.

3.3.1. Pruebas de Software

Conjunto de técnicas experimentales para la búsqueda de fallos en los programas, que determinan en cierto grado la calidad de un producto. [14].

Las pruebas que se le realizaron al subsistema corresponden al nivel de Unidad, la cual se deriva en Pruebas Estructurales (Caja Blanca) y Pruebas Funcionales (Caja Negra). 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.

3.3.1.1. Método de Prueba (Caja Blanca)

Esta prueba consiste específicamente en diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento. Para la realizar esta prueba de calidad se utilizó el framework JUnit el cual se puede integrar al IDE de desarrollo Eclipse. Este framework permite la automatización de las pruebas de aplicaciones en Java, consta de un conjunto de clases que el programador puede utilizar para construir sus casos de prueba y ejecutarlos automáticamente. Además mejora el diseño de implementación, haciéndolo flexible y testeable. [17]

Para realizar las pruebas de caja blanca con JUnit se toma como objeto de estudio el método ListarMonedas () que se encuentra en la clase MT950MultiActionController, el cual se muestra a continuación.

```
public void listarMonedas(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
    String banco = request.getParameter("codBanco");
    List<NomMoneda> monedas=mt950Facade.listarMonedas(banco);
    JSONObject json = new JSONObject();
    JSONArray array = new JSONArray();
    for (int i = 0; i <monedas.size(); i++) {
    JSONObject data = new JSONObject();

        data.put("nombre", monedas.get(i).getSigMoneda());
        data.put("value", monedas.get(i).getCodMoneda());
        array.add(data);
    }

    json.put("items", array);
    json.put("identifier", "value");
    json.put("label", "nombre");
    json.put("value", "value");

try {
        ResponseUtil.escribirDatosEnElResponse(response, json);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figura 15: Método testeado por el framework JUnit.

Para realizarles las pruebas a dicho método primeramente se crea una clase la cual debe extender de TestCase, en este caso dicha clase se nombra TestJUnit en la cual se realizan una serie de pasos los cuales se muestran y describen a continuación.

```
public class TestJUnit extends TestCase {

    MT950MultiActionController mt950MultiActionController;

    private MockControl controlHttpServlet;
    private HttpServletRequest mockhttpServletRequest;
    private MockControl controlHttpResponse;
    private HttpServletResponse mockhttpServletResponse;
    private MockControl controlHttpSession;
    private HttpSession mockHttpSession;

    protected void setUp()throws Exception {
        mt950MultiActionController = new MT950MultiActionController();
        ApplicationContext context = new ClassPathXmlApplicationContext(
        "classpath:cu/uci/finixubnc/JUnit/conciliacion-context.xml");
        MT950ManagerImpl manager = (MT950ManagerImpl) context.getBean("mtManager");
        GestionarMT950FacadeImpl facade = new GestionarMT950FacadeImpl();
        facade.setManager(manager);

        mt950MultiActionController.setMt950Facade(facade);

        controlHttpServlet= MockControl.createControl(HttpServletRequest.class);
        mockhttpServletRequest=(HttpServletRequest) controlHttpServlet.getMock();

        controlHttpResponse= MockControl.createControl(HttpServletResponse.class);
        mockhttpServletResponse=(HttpServletResponse) controlHttpResponse.getMock();

        controlHttpSession= MockControl.createControl(HttpSession.class);
        mockHttpSession = (HttpSession) controlHttpSession.getMock();
        super.setUp();
    }
```

Figura 16: Clase TestJUnit para realizar la prueba.

Primeramente se crea un objeto de la clase en la cual se encuentra el método que se va a testear y después dos simulacros, uno de HttpServletRequest y otro de HttpServletResponse que son los parámetros que se les pasa a dicho método, además de sus respectivos MockControl encargados de su control. Posteriormente en el método setUP() se inicializan todos los atributos anteriormente creados, además de un contexto, el cual contiene todos los beans de las clases necesarias para la ejecución del procedimiento que se va a probar.

```
protected void tearDown() {
    controlHttpServlet.verify();

}

public void testListarMonedas() throws Exception{
    mockhttpServletRequest.getParameter("codBanco");
    controlHttpServlet.setReturnValue("0001");

    controlHttpServlet.replay();
    assertTrue(mt950MultiActionController.listarMonedas(mockhttpServletRequest,mockhttpServletResponse));
}
```

Figura 17: Métodos tearDown y testListarMoneda de la clase TestJUnit.

En el método tearDown es donde se verifican si las llamadas a los métodos se realizaron correctamente, es el encargado de informar la existencia de los errores en la realización de las pruebas. Las verificaciones en este método se realizan de forma automática para todos los test definidos en la clase que se encuentra en este caso TestJUnit.

Una vez definido el tearDown se crea un método, el cual contendrá los elementos para realizar el test. En dicho procedimiento creado (testListarMonedas), se específica como son recibidos los parámetros y cuál es el identificador de cada uno, en el método que será sometido a prueba. Posteriormente mediante assertTrue() se ejecuta procedimiento que está siendo analizado y se espera que el framework termine el análisis. Según el resultado del mismo JUnit muestra una ventana en correspondencia con este: si es satisfactorio, mediante una línea de color verde, en caso contrario de color rojo. A continuación se muestra el caso favorable para dicha prueba.

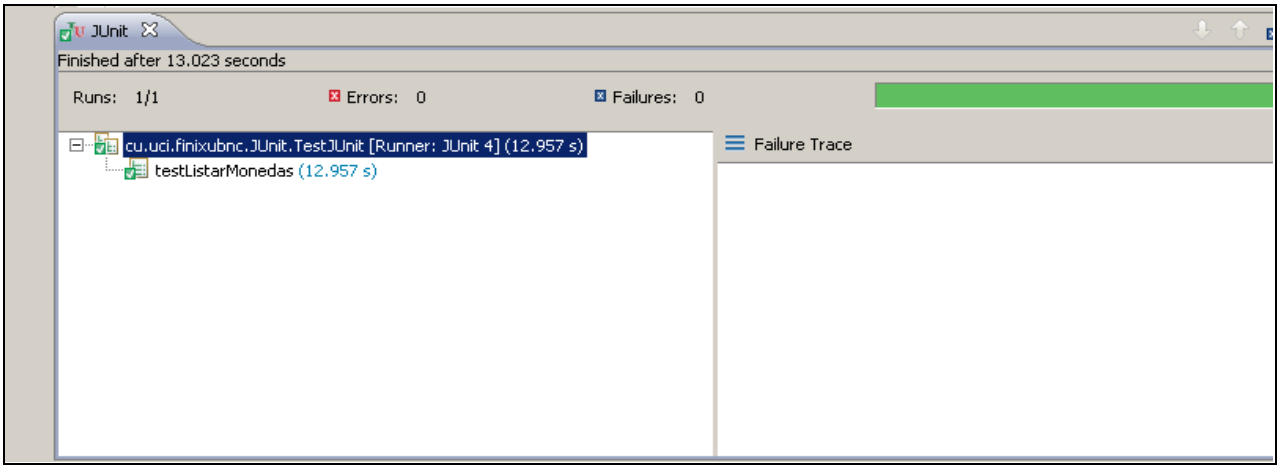


Figura 18: Consola de JUnit.

3.3.1.2 Método de Prueba (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.

La prueba de Caja Negra no es una alternativa a las técnicas de prueba de la 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 Funcionales realizadas al subsistema Conciliaciones Bancarias del Sistema QUARXO en su versión 1.0 se llevaron a cabo por parte del grupo de calidad del CEIGE (Centro de Informatización de la Gestión de Entidades). Primeramente se realizó la solicitud al Jefe de Pruebas y una vez aceptada la

misma se generaron los casos de pruebas del subsistema. Se realizaron un total de 2 iteraciones siendo detectadas 3 No Conformidades en la primera que fueron corregidas para la segunda iteración logrando el resultado de 0 No Conformidades. Se emplearon un total de 2 horas efectivas de trabajo en la siguiente distribución: 1h (06/11/2012), 1h (06/12/2012) en un total de 2 turnos de trabajo, con 1 probador en cada turno, en este caso el mismo, y toda la actividad estuvo dirigida por un Jefe de Pruebas. Luego se emitió el acta de liberación del subsistema firmada por el "Especialista del Laboratorio de Calidad" y el "Responsable del Equipo de Desarrollo" quedando validado funcionalmente el subsistema en cuestión.

Los resultados obtenidos a través de la aplicación de los métodos de prueba expuestos con anterioridad fueron satisfactorios desde el punto de vista interno y funcional, atendiendo al correcto comportamiento del mismo ante diferentes situaciones.

El subsistema Conciliaciones Bancarias una vez probado y liberado por parte del grupo de calidad del CEIGE, se encuentra listo para entrar a las Pruebas de Integración y Aceptación, esta última debe llevarse a cabo en el Banco Nacional de Cuba, donde será 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.

3.4. Conclusiones parciales

Una vez desarrolladas las tareas que dan por cumplido este capítulo, se concluye que la solución implementada y posteriormente validada, desde el punto de vista funcional, permite llevar a cabo los procesos relacionados con la Conciliación Bancaria en el BNC, contribuyendo a la ejecución de las operaciones correspondientes.

CONCLUSIONES

Una vez concluido el presente trabajo de diploma se puede afirmar que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos:

- Se analizaron ventajas y deficiencias de sistemas informáticos tanto nacionales como internacionales vinculados a la Conciliación Bancaria, donde la dificultad principal para su utilización resultó ser el alto costo de sus licencias. Evidenciándose de esta manera la necesidad de desarrollar una solución informática capaz de ejecutar dichas funcionalidades y que se ajustara a las necesidades del BNC.

- Fue necesario mejorar el análisis y diseño previamente realizado para llevar a cabo la implementación del subsistema Conciliaciones Bancarias a partir de los requisitos definidos.

- Se evaluó el subsistema a través de pruebas de software efectuadas a nivel de unidad, las cuales arrojaron resultados favorables posibilitando el cumplimiento de las funcionalidades previstas para el mismo.

La propuesta exhibe valor técnico; se destaca por la incorporación de principios por los que se mide la factibilidad de un diseño de software como lo es la utilización de patrones, lo que posibilitó la reutilización y el mantenimiento del sistema.

La importancia de la solución radica en la realización de los procesos referentes a la Conciliación Bancaria en el BNC a través del subsistema que funciona rápido y correctamente, permitiendo el ahorro de tiempo, recursos y el control de errores.

RECOMENDACIONES

- Realizar la integración del subsistema Conciliaciones Bancarias con el componente de mensajería SLBTR.
- Mejorar la solución propuesta en futuras versiones del subsistema a partir de los estudios realizados en la presente investigación.

BIBLIOGRAFÍA

1. **Rodríguez, Lic. Romelia.** Conciliación Bancaria. Guayana (Bolivar): s.n., 2007.
2. **SICSA.** SIC solucions. [En línea] [Citado el: 10 de Nomviembre de 2011.] http://www.sicsa.com/.
3. **ISISSA**. Sistema Isis Classic. [En línea] [Citado el: 10 de Noviembre de 2011.] www.sistemaisis.com/software-de-gestion-de-bancos.htm.
4. **Winledger**. [En línea] [Citado el: 10 de Noviembre de 2011.] www.winledger.com/conciliacion.php.
5. **Sicob**. [En línea] [Citado el: 10 de Noviembre de 2011.] www.sicob.com.mx.
6. **Editran**. Web de Ayuda. [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.editran.info/pa_conciliacion_bancaria.html.
7. **Conciliacion Express**. [En línea] 2009. [Citado el: 10 de Noviembre de 2011.] http://www.conciliacionexpress.com/index.php.
8. **Portal SIAFI**. Portal SIAFI. [En línea] [Citado el: 10 de Noviembre de 2011.] http://chorti.sefin.gob.hn/siafi/beneficiarios.html.
9. **Concepto de Contabilidad**. PromoNegocio. [En línea] Enero de 2008. [Citado el: 10 de Noviembre de 2011.] http://www.promonegocios.net/contabilidad/concepto-contabilidad.html.
10. **Edgar Naranjo Fuentes, Yelena Reyes Hidalgo**. Análisis y Diseño del subsistema Conciliación Bancaria de QUARXO. 2011.
11. **Jacobson, Ivar , Booch, Grady y Rumbaugh, James .** El proceso unificado de desarrollo de software. 3ra Edición. España: : Addison-Wesley, 2007
12. **Barroso y Bello.** Diseño e Implementación del subsistema Cartas de Créditos del Proyecto SAGEB. 2010.
13. **Iglesias, Adolfo Miguel.** Doc. Arquitectura del sistema de modernización bancaria (Finixu). La Habana : s.n., Noviembre 2008.
14. **Manuel A. Borroto Santana.** Diseño e Implementación de los subsistemas Créditos y Depósitos del Sistema QUARXO. 2011.
15. **ECURED.** Metodologías de Desarrollo de Software. [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software.
16. **ProgrammerWorld.NET**. Best Java IDE . [En línea] Lingaya's Institute of Management and Technology Faridabad, India, 2007. [Citado el: 10 de Noviembre de 2011.] http://faq.programmerworld.net/programming/best-java-ide-review.html.
17. **EcuRed. Prueba de Caja Blanca.** [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.ecured.cu/index.php/Pruebas_de_caja_blanca.

18. **La Rosa y Del Castillo, Alberto Jesus y Agramonte Rodríguez, Hiran.** Herramienta para la instalación y configuración de clústeres del Contenedor de Servlets Apache Tomcat. La Habana : s.n., 2009.
19. **the JBoss Way**. [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.jboss.org/.
20. **GlassFish - Open Source Application Server**. [En línea] [Citado el: 10 de Noviembre de 2011.] http://glassfish.java.net.
21. **Control de Versiones**. [En línea] [Citado el: 10 de Noviembre de 2011.] http://producingoss.com/es/vc.html.
22. **Control de versiones con GIT**. [En línea] [Citado el: 10 de Noviembre de 2011.] http://ecentinela.com/control-de-versiones-con-git/.
23. **¿Qué es PHP?** [En línea] Dattatec Ayuda. [Citado el: 10 de Noviembre de 2011.] http://dattatecayuda.com/%C2%BFque-es-php/1860..
24. **Visual Studio.** [En línea] [Citado el: 10 de Noviembre de 2011.] http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express..
25. **Chapter 15. Integrating with other web frameworks**. [En línea] Spring Source. [Citado el: 10 de Noviembre de 2011.] http://static.springsource.org/spring/docs/2.5.x/reference/web-integration.html.
26. **Core JavaServer™ Faces**, Second Edition. Santa Clara, California : s.n., 2007. 978-0-13-173886-7.
27. **Los frameworks JS más usados**. anieto2K. [En línea] 2007. [Citado el: 10 de Noviembre de 2011.] http://www.anieto2k.com/2006/10/11/los-frameworks-js-mas-usados.
28. **Santana, Vera , y otros, y otros.** 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. Ecuador : Guayaquil, 2009.

GLOSARIO DE TÉRMINOS

Servlet: Es utilizado para generar páginas web de forma dinámica a partir de parámetros de una petición de un navegador web, utilizando para este objetivo el acceso a bases de datos, flujos de trabajo y otros recursos.

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

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.

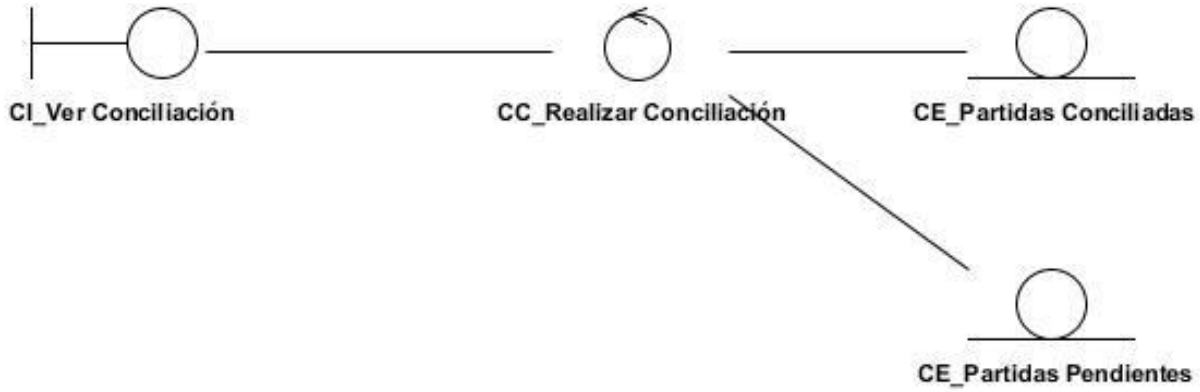
IDE: Es un entorno de desarrollo integrado, constituyendo un programa informático compuesto por un conjunto de herramientas para la programación.

CVS: Concurrent Versions System o simplemente CVS, es una aplicación informática que implementa un sistema de control de versiones: mantiene el registro de todo el trabajo y los cambios en los ficheros (código fuente principalmente) que forman un proyecto (de programa) y permite que distintos desarrolladores (potencialmente situados a gran distancia) colaboren.

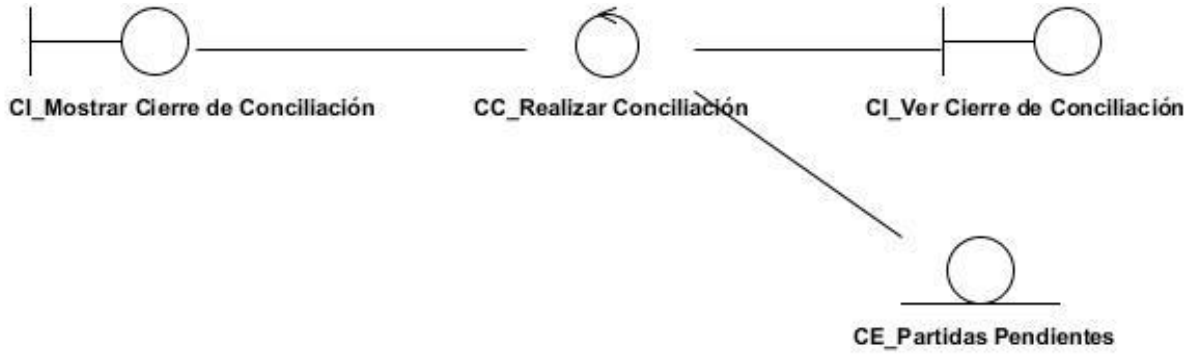
Estado de Cuentas: Es la verificación y resumen, de los distintos créditos y los distintos deudores de una cuenta en específico.

Libro de Banco: Es la bitácora o 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.

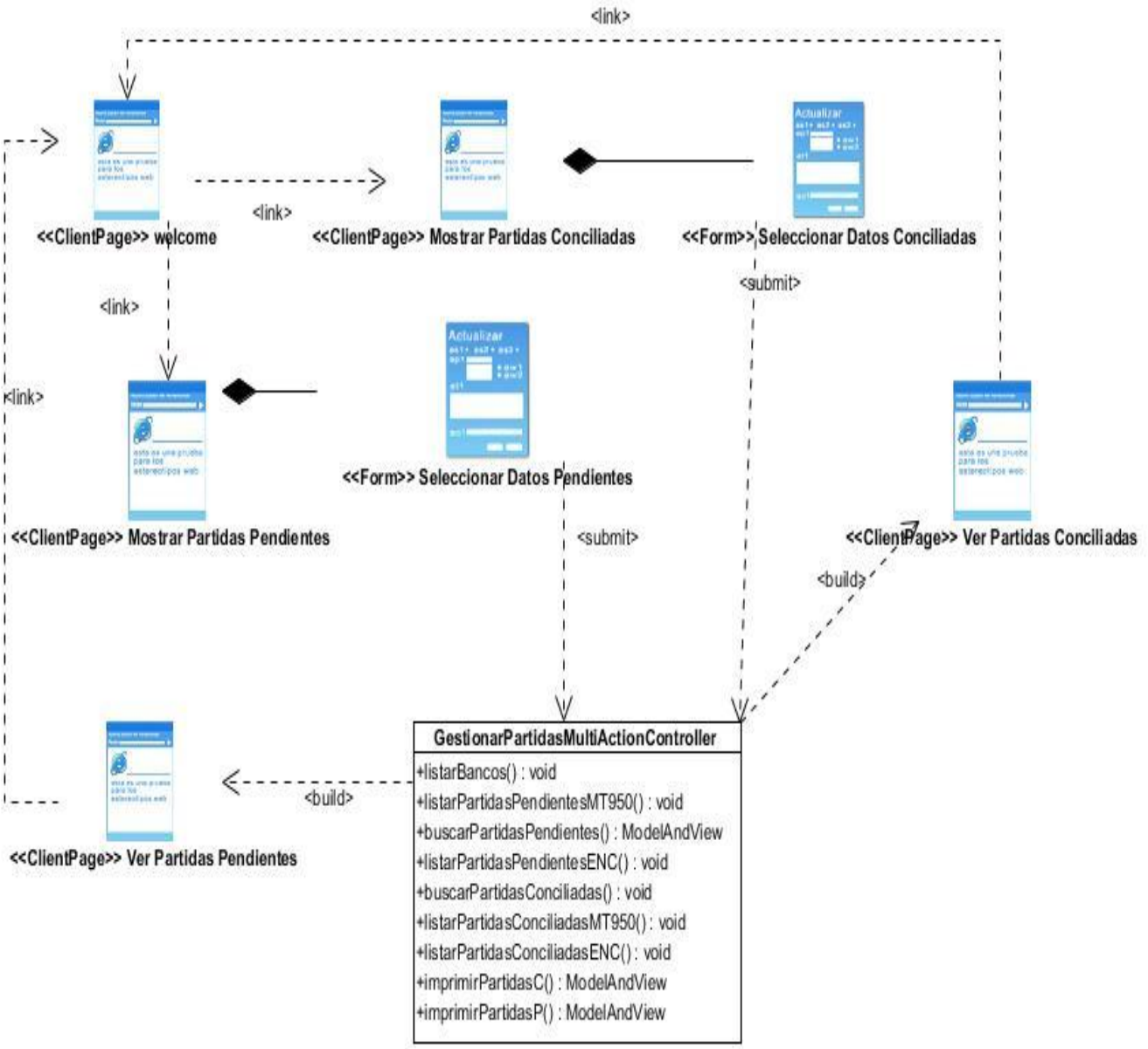
ANEXOS



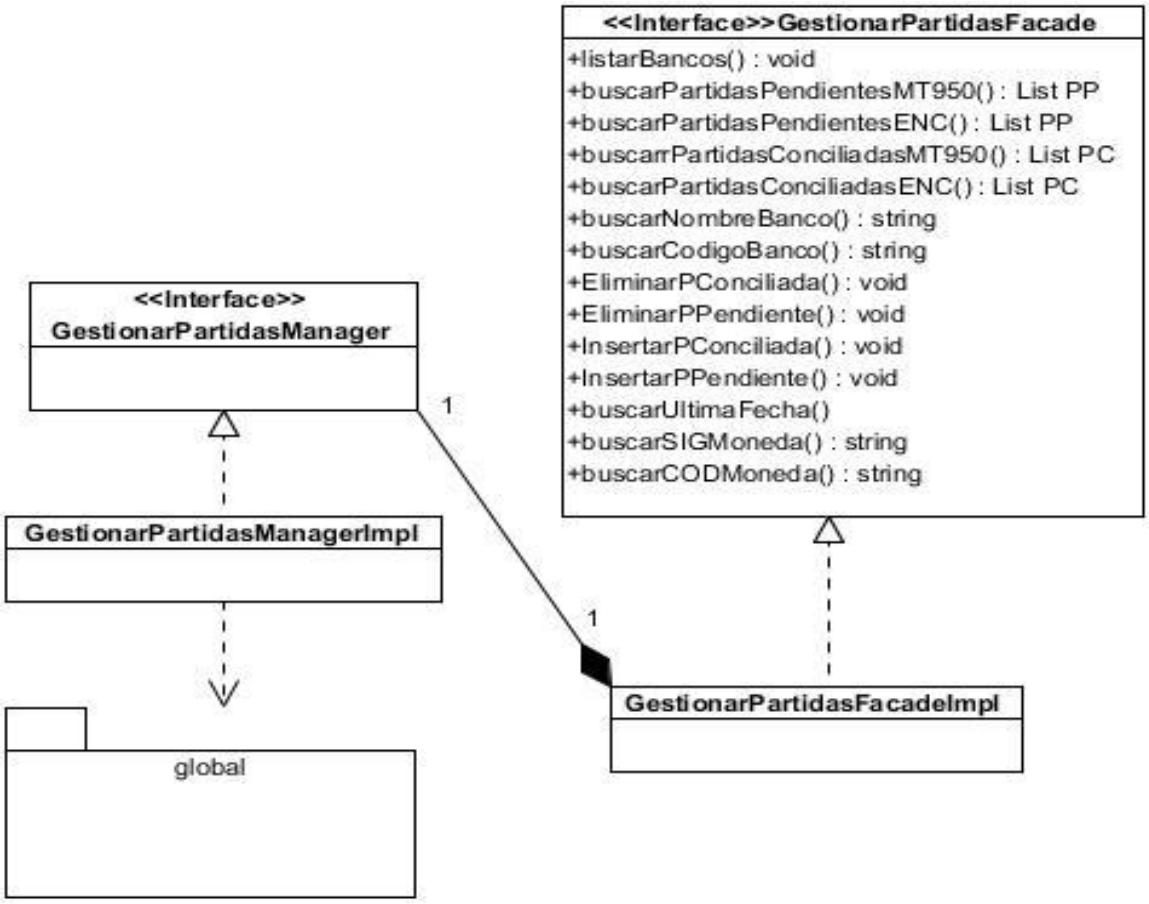
Anexo 1. Modelo de Análisis del caso de uso Realizar Conciliación Manual.



Anexo 2. Modelo de Análisis del caso de uso Gestionar Cierre.

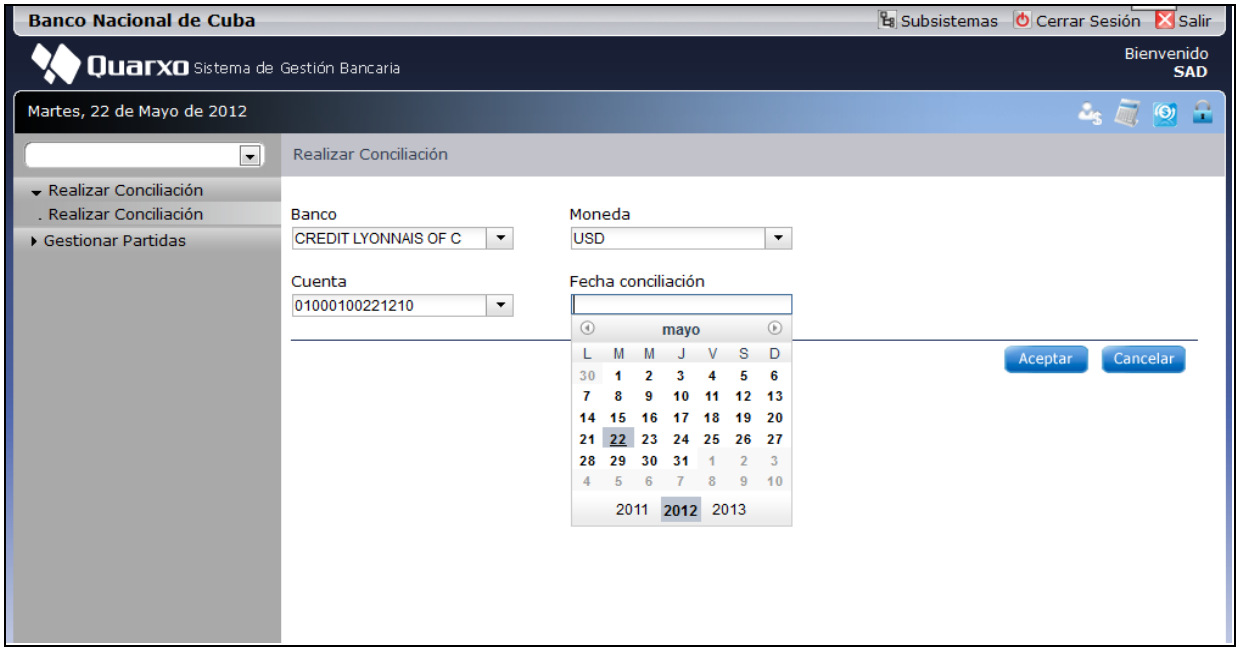


Anexo 3. Diagrama de clase de diseño (Capa de presentación) del módulo Gestionar Partidas.

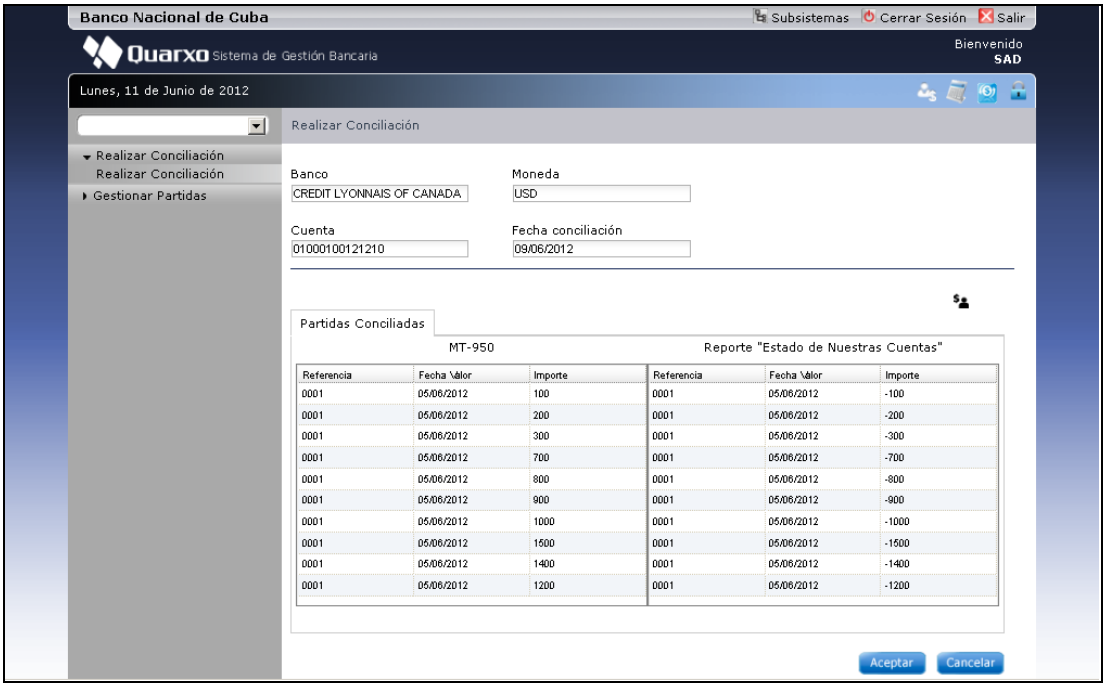


Anexo 4. Diagrama de clase de diseño (Capa de negocio) del módulo Gestionar Partidas.

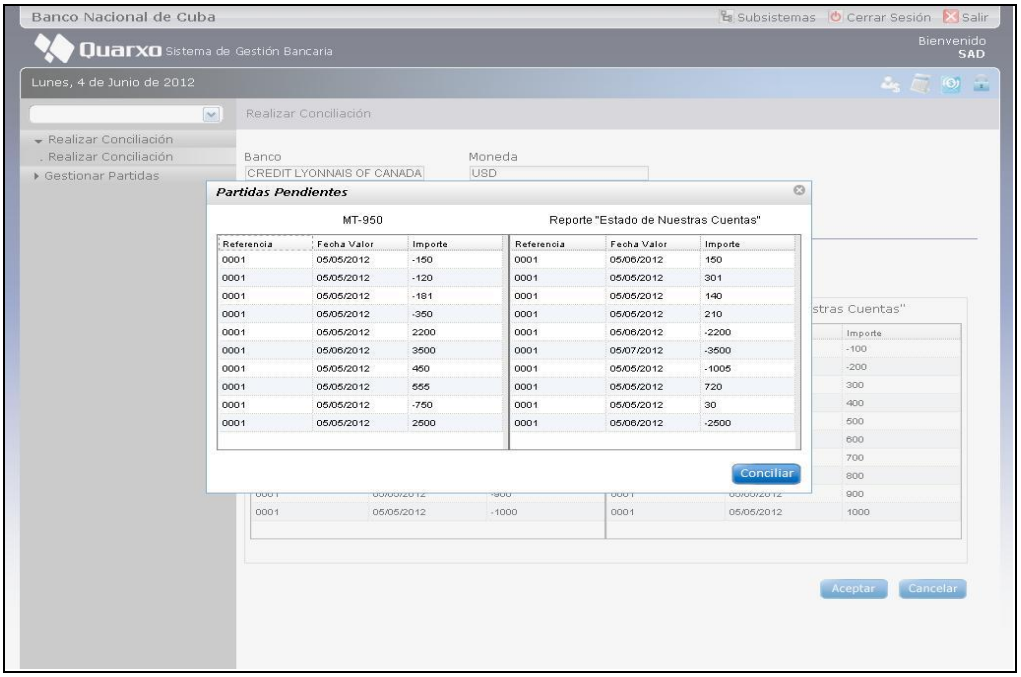
Anexo 5: Interfaz Principal del subsistema Conciliaciones Bancarias.



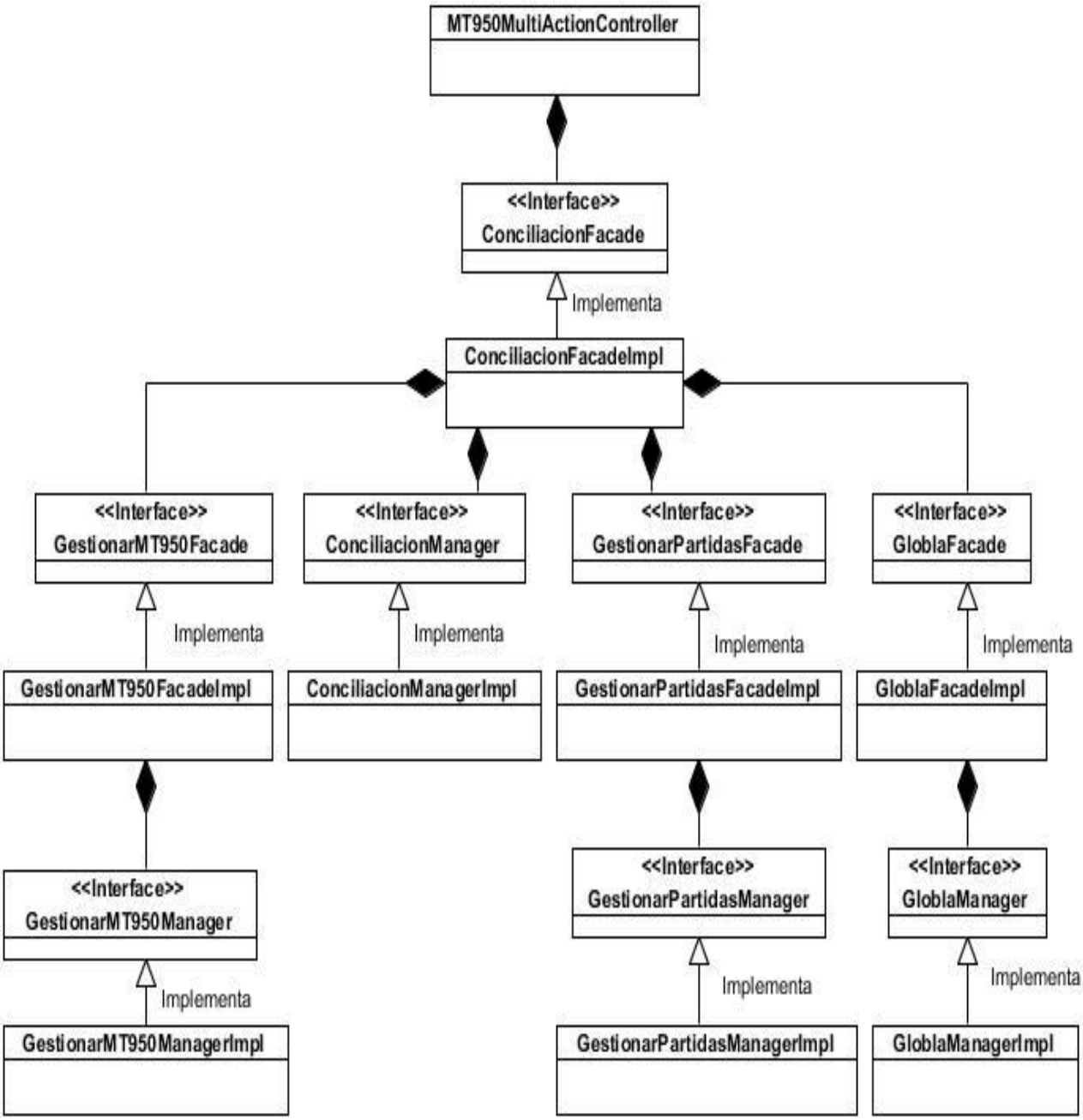
Anexo 6: Interfaz del caso de uso Realizar Conciliación del Sistema.



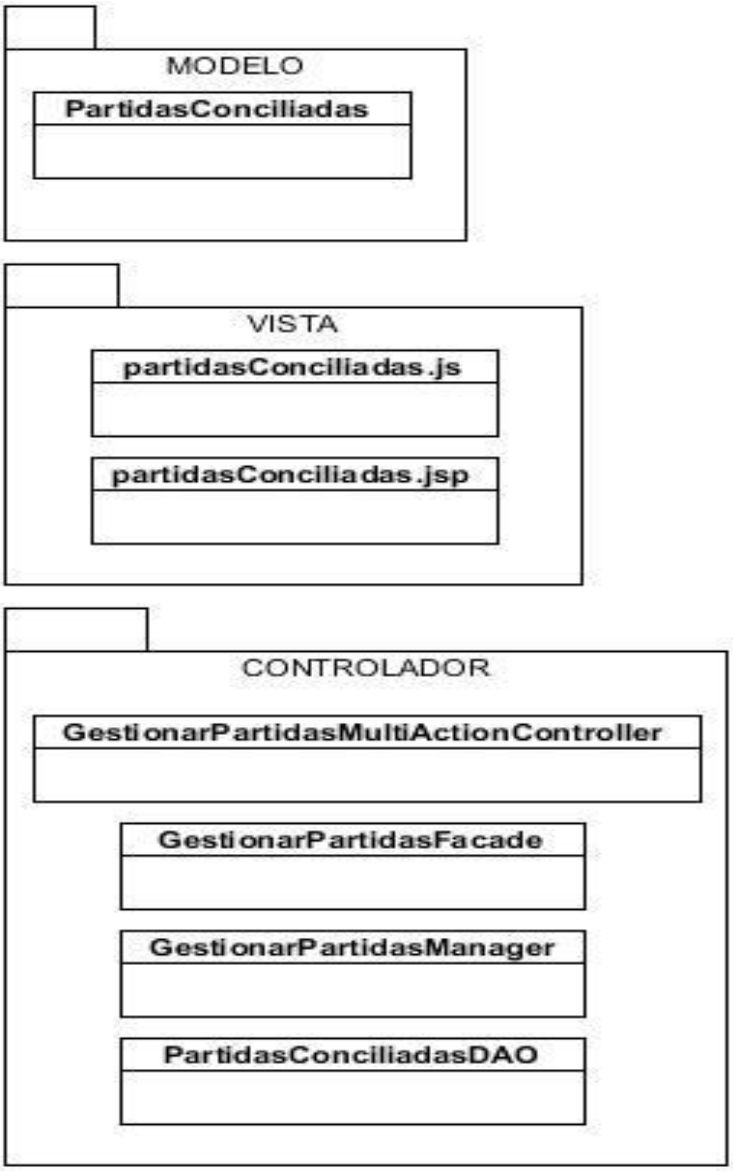
Anexo 7: Interfaz del caso de uso Realizar Conciliación del Sistema.



Anexo 8: Interfaz del caso de uso Realizar Conciliación Manual.



Anexo 9: Patrón Facade.



Anexo 10: Patrón Modelo Vista Controlador.