Universidad de las Ciencias Informáticas

Facultad 3

Título: Desarrollo del subsistema Títulos Valores fase 2.

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

Autor: Aylén Jomarrón Jomarrón.

Tutores: Ing. Dariel Membribes Rodríguez.

La Habana, Junio 2013

“Año 55 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Aylén Jomarrón Jomarrón

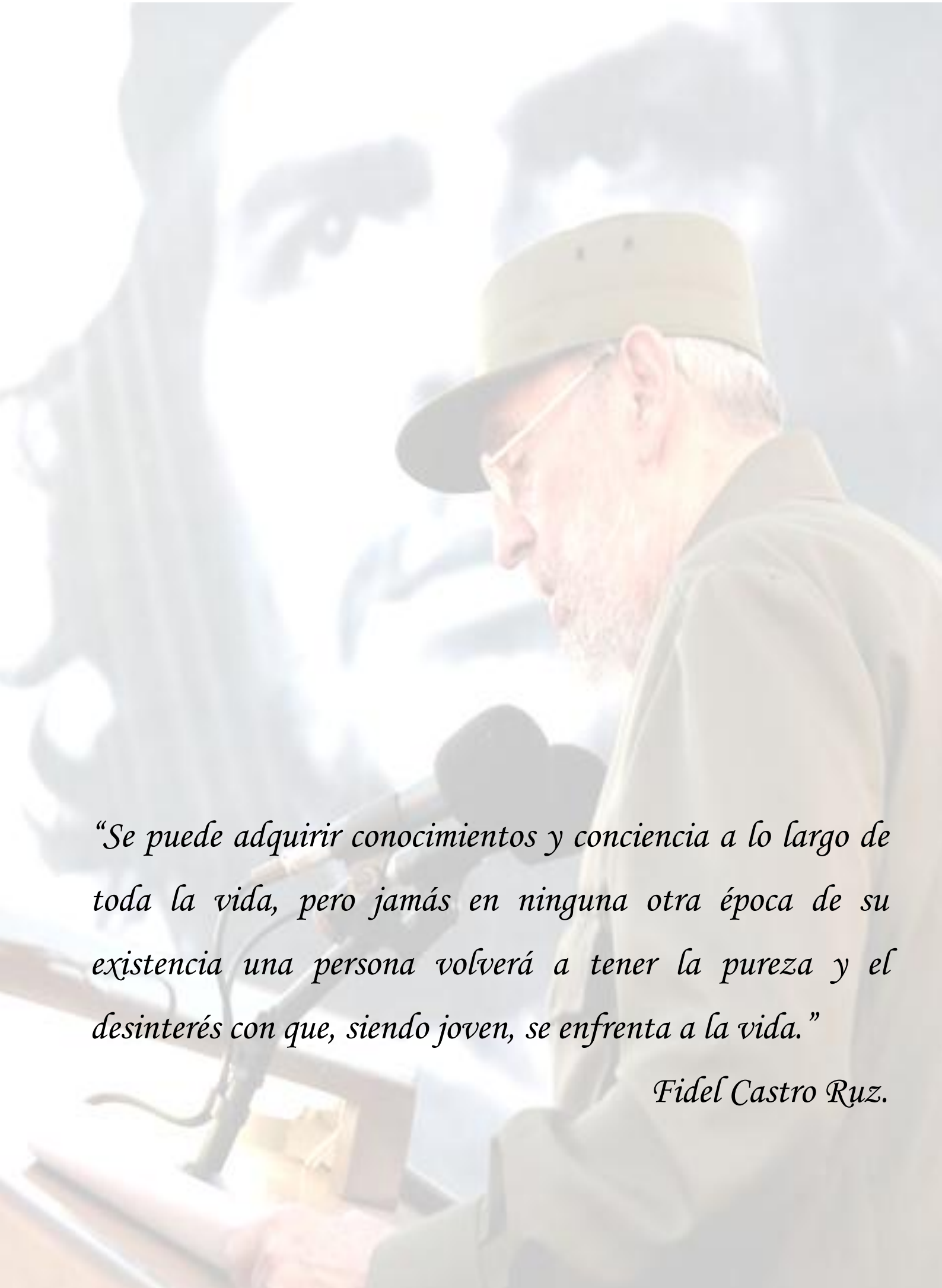
_____

Firma del Autor

Ing. Dariel Membribes Rodríguez.

_____

Firma del Tutor



"Se puede adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida."

Fidel Castro Ruz.

AGRADECIMIENTOS

A mi mamá Maira Jomarrón Vega, por ser madre y padre a la vez y más que eso por ser mi mejor amiga, por darme todo lo que tengo, porque sin ella nunca hubiese logrado llegar hasta aquí, gracias mami por ser mi guía en la vida.

A mis abuelos Gabriela y Ramón por ser mis segundos padres, por su amor y dedicación.

A mi hermana Yaimé, por su confianza y apoyo, su cariño y por quererme tanto.

A toda mi familia que siempre me ha guiado y apoyado en mis decisiones.

A mi tutor Dariel Membribes Rodríguez por toda su ayuda y apoyo, porque sin él no hubiese sido posible el desarrollo del presente trabajo.

A Juan Javier Dans Moreno por dejarme ser parte de su vida, por sus consejos, su ayuda incondicional, por ser tan especial como lo ha sido y por ser un gran amigo.

A Jorge Vargas García por su ayuda y sus enseñanzas, por ser mi amigo ante todo, por haberme soportado tanto.

A mis amigas Yadini, Yadelis, Liset y Diana, por estar presentes en mi vida, por apoyarme en los momentos difíciles, por los instantes que compartimos y por ayudarme siempre que lo necesité.

A todas aquellas personas que de una forma u otra han sido parte de mi vida y que han contribuido a que hiciera posible el sueño de ser ingeniera.

A todos muchas gracias, de corazón.

Aylén Jomarrón Jomarrón.

DEDICATORIA

El presente trabajo va dedicado especialmente a mi papá Alberto Jomarrón Leyva, que aunque no está conmigo físicamente, lo llevo en mi corazón y ha sido mi motivación principal para llegar hasta aquí. Te amo papito.

A mi mamá por ser tan especial, por apoyarme en todo momento y por su amor. Te quiero mami.

A mis abuelos, mis hermanas en especial a Yai, mis sobrinos especialmente a Manuel Alberto, a mi primo y a mi tío y a toda mi familia en general que son tan especiales para mí porque son lo que más quiero en este mundo.

A mis amistades y compañeros de la universidad que siempre los llevaré en mi corazón.

Aylén Jomarrón Jomarrón.

RESUMEN

El uso de las Tecnologías de la Información y las Comunicaciones (TIC) permiten a las organizaciones la optimización de sus recursos y mejora en sus operaciones, logrando aumentar su eficiencia y competitividad. Las empresas y entidades cubanas han decidido aplicar las nuevas tecnologías de la información a sus espacios de trabajo para obtener los beneficios que trae consigo la utilización de las mismas. La Universidad de las Ciencias Informáticas tiene como principal objetivo promover el estudio de esta importante rama que es la informática. Dentro de la propia universidad existen varios centros de desarrollo de software. Uno de ellos es el Centro de Informatización de la Gestión de Entidades el cual dirige sus resultados hacia la esfera de la gestión de empresas y entidades. Dentro de este centro se encuentra el proyecto Banco, creado para desarrollar un sistema de gestión bancaria que permita al Banco Nacional de Cuba tener todas sus operaciones informatizadas. De dicho sistema se obtuvo la primera versión, el cual fue nombrado Quarxo y se encuentra ya en ejecución en el banco. Esta primera versión no satisface todas las necesidades de los operadores bancarios ya que existen departamentos que aún realizan sus procesos en el formato tradicional. El departamento de Negociaciones, por ejemplo, debe gestionar los medios de pago de forma manual, ocasionando pérdida de tiempo y gastos de recursos materiales. El presente trabajo de diploma tiene como objetivo desarrollar los módulos Letra de cambio y Pagaré del subsistema Títulos Valores perteneciente al sistema bancario Quarxo para facilitar la gestión de los procesos asociados al pago de las negociaciones que se realizan en la entidad.

Palabras claves: Sistema bancario Quarxo, Títulos Valores**,** Letra de cambio, Pagaré.

Tabla de contenidos

Tabla de contenidos

INTRODUCCIÓN

Con el transcurso del tiempo el hombre ha ido evolucionando y junto a él se han ido perfeccionando los diferentes campos de la ciencia y la tecnología. La informática ha tenido un desarrollo exponencial en el mundo, que ha propiciado la aparición e implementación de nuevas formas para el registro de eventos y actividades, tratamiento de los procesos de control, gestión y toma de decisiones. El diseño de nuevas herramientas para la administración de la información permite que las Tecnologías de la Información y las Comunicaciones (TIC) se encuentren presentes en todas las esferas y sectores de la sociedad (Salud, 2003). Las TIC desempeñan un papel relevante en la economía mundial por su capacidad de atracción de inversión y generación de valor, lo cual se ve reflejada en las nuevas capacidades productivas. La industria del software y servicios informáticos ha sido una de las más dinámicas a escala global en los últimos años. Esto no es sorprendente, considerando que el software desempeña un papel clave dentro del conjunto de avances tecnológicos (Casals, 2010).

El gobierno cubano se dio a la tarea de desarrollar y perfeccionar la industria del software, con el objetivo de desarrollar sistemas para informatizar la sociedad y escalar en el mercado del software a nivel global (Casals, 2010). El proceso de Informatización de la Sociedad Cubana, se ha definido como aquel en que se aplican las Nuevas Tecnologías de la Información y las Comunicaciones (NTIC) a las diferentes esferas y sectores de la sociedad. Para de esta manera lograr una mayor eficacia y eficiencia con la optimización de recursos y el logro de mayor productividad y competitividad en dichas esferas y sectores (Salud, 2003). La Universidad de las Ciencias Informáticas (UCI), surgida al calor de la Batalla de Ideas fue concebida precisamente para impulsar el desarrollo de la informática en el país. La UCI desempeña un papel protagónico desde su surgimiento mediante la producción de soluciones y servicios informáticos. Adscrito a la Facultad 3 de la propia universidad se encuentra el Centro de Informatización de la Gestión de Entidades (CEIGE). Dicho centro productivo dirige sus resultados hacia la esfera de la gestión de empresas y entidades. La producción se concentra en el desarrollo de proyectos generalmente de gran magnitud (Obregón, 2012). Dentro de este centro se encuentra el proyecto Banco, el cual fue creado para desarrollar un sistema de gestión bancaria que brinde apoyo al Sistema Bancario Cubano (SBC), específicamente al Banco Nacional de Cuba (BNC).

El SBC es el conjunto de entidades o instituciones que dentro de una economía, prestan los servicios de banco, es decir, almacenan los fondos y garantizan que estén disponibles para cuando se soliciten (Muñoz, 2008). El SBC está encabezado por el Banco Central de Cuba (BCC) y constituido por nueve bancos comerciales, quince instituciones financieras no bancarias, once oficinas de representación de bancos extranjeros en Cuba y cuatro oficinas de representación de instituciones financieras no

bancarias (BCC, 2013). El BNC, perteneciente al SBC, tiene la necesidad de enfrentar el proceso de informatización que aportará con las nuevas tecnologías, la eficacia y eficiencia que requieren sus servicios, un adecuado control de los recursos y como resultado una mejor calidad en la atención a sus clientes.

Producto de la necesidad de informatizar sus procesos para mejorar la eficiencia de los servicios y un adecuado control de los recursos materiales el BNC le solicitó a la UCI el desarrollo de un sistema informático. Como resultado se obtuvo la primera versión del sistema Quarxo, el cual se encuentra actualmente en explotación en dicho banco. Esta primera versión de Quarxo no satisface todas las necesidades de los operadores bancarios ya que existen departamentos que aún realizan sus procesos en el formato tradicional. Entre los subsistemas que tiene implementado este sistema bancario se encuentra el de Títulos Valores, el mismo no está completo debido a que le falta la gestión de las Letras de cambio y Pagarés, por tal motivo estos procesos se realizan de forma manual, lo que provoca la ocurrencia de errores humanos, aumentando la pérdida de tiempo y gastos en recursos materiales. Como consecuencia, la gestión de Títulos Valores por parte de los negociadores se vuelve un proceso engorroso e ineficiente y en numerables ocasiones la información no se tiene en el momento preciso para la toma de decisiones oportuna.

A partir de la problemática planteada se define como **problema a resolver**: la gestión actual de los procesos asociados a los Títulos Valores se ve afectada por las deficiencias de su ejecución en el BNC.

Definiéndose como **objeto de estudio**: los procesos asociados a la gestión de los Títulos Valores.

Según lo planteado anteriormente se establece como **campo de acción**: los procesos asociados a la gestión de los Títulos Valores en el BNC.

Una vez planteada la problemática se define como **objetivo general**: desarrollar el Subsistema Títulos Valores fase 2 empleando el modelo de desarrollo de CEIGE, de manera que contribuya a la mejora en el desempeño de los procesos asociados a los Títulos Valores en el BNC.

Se plantea la siguiente **idea a defender**: si se desarrolla el Subsistema Títulos Valores fase 2 empleando el modelo de desarrollo de CEIGE, se contribuirá a la mejora en el desempeño de los procesos asociados a los Títulos Valores en el Banco Nacional de Cuba, minimizando el gasto de tiempo y recursos materiales.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico relativo a las soluciones de software que gestionan los Títulos Valores.
- Modelar los procesos del negocio de Títulos Valores en una segunda fase.
- Realizar el análisis y diseño de las funcionalidades asociadas a los Títulos Valores en su segunda fase.
- Implementar el subsistema Títulos Valores en una segunda fase bajo la tecnología propuesta.
- Validar la solución propuesta.

Para el estudio preliminar de la investigación se utilizaron los siguientes métodos científicos:

Métodos Teóricos:

- Histórico – Lógico: fue utilizado para realizar un estudio de las tendencias históricas y actuales de los sistemas bancarios que existen en el mundo y determinar su influencia en el problema actual de la investigación.

- Analítico - Sintético: se utilizó para realizar un análisis de la documentación empleada para el desarrollo de la investigación.

- Modelación: utilizado para modelar los requerimientos funcionales que se proponen en la solución, su relación con los procesos actuales y el diseño de los nuevos componentes visuales, las entidades relacionales y sus características.

Métodos Empíricos:

- Entrevistas: este método fue de gran utilidad para realizar el estudio de los procesos actuales referentes a la gestión de los Títulos Valores en el BNC y sus características (Ver Anexo 1).

El trabajo está estructurado por 3 capítulos en los que se encuentra el contenido distribuido de la siguiente forma:

En el **primer capítulo** se muestra la fundamentación teórica de la investigación: el estado del arte de algunas de las soluciones informáticas existentes para la gestión de los Títulos Valores a nivel internacional y nacional. Se definen los conceptos principales que se deben conocer para comprender

el negocio. Se detalla el lenguaje, las herramientas y el modelo de desarrollo utilizados en la solución obtenida. Se realiza un estudio de las técnicas de la ingeniería de requisitos y los patrones de software más utilizados.

En el **segundo capítulo** se profundiza en el análisis y diseño, dejando claro lo que se realiza en cada una de estas fases. Se describe la arquitectura como base para el desarrollo de la aplicación. Se realiza el modelo de negocio, especificando la descripción de los procesos de negocios. Se detallan los requisitos del sistema, patrones arquitectónicos y de diseño utilizados en la concepción de la solución. Se muestran los artefactos obtenidos según lo propuesto en el modelo de desarrollo de CEIGE para alcanzar la solución final.

En el **tercer capítulo** se abordan temas correspondientes a la implementación del sistema y las pruebas que se realizan en la solución. Explicando aspectos como estándares de codificación, la interacción de los componentes a través de los diagramas correspondientes. Se realiza la validación de la solución a partir de la aplicación de pruebas de caja negra y caja blanca y por último se validan los resultados de la investigación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el capítulo se describen los elementos principales que fundamentan el contenido del trabajo. Se definen los principales conceptos que deben ser conocidos para comprender en su totalidad el negocio. Se realiza un estudio profundo de las soluciones informáticas para la gestión de las Letras de cambio y Pagarés, además del estudio realizado de algunas de las técnicas de captura y validación de los requisitos. Se caracterizan los estilos arquitectónicos, patrones de desarrollo de software, herramientas, lenguajes y modelo de desarrollo utilizado para el desarrollo de la solución.

1.2 Conceptos fundamentales

1.2.1 Banco Nacional de Cuba (BNC)

El BNC fue creado mediante la Ley No. 13, del 23 de diciembre de 1948, como banco central del Estado, con autonomía orgánica, personalidad jurídica independiente y patrimonio propio. El 23 de febrero de 1998 se promulga el Decreto Ley No. 181, que recoge la estructura, funciones y actividades asignadas como organización bancaria internacional. El BNC desarrolla un activo trabajo en las negociaciones y firmas de acuerdos vinculados a la renegociación de la deuda externa del país, obteniendo nuevos créditos con bancos e instituciones extranjeras, con mejores condiciones financieras. Otra de las prestaciones que brinda es conceder financiamientos en moneda libremente convertible a entidades nacionales (BCC, 2013).

Entre sus funciones se encuentra (BCC, 2013):

- Obtener y otorgar créditos en moneda nacional y en moneda libremente convertible.
- Emitir garantías bancarias de todo tipo con previa evaluación de la gestión económico-financiera de la entidad solicitante.
- Fijar las tasas de interés que deberán aplicar a las operaciones que efectúe el Banco, dentro de los límites que establezca el BCC.
- Constituir entidades de seguro de crédito oficial a la exportación con arreglo a la legislación vigente en materia de seguros.
- Librar, aceptar, descontar, avalar y negociar, en cualquier forma, Letras de cambio, Pagarés y otros documentos mercantiles negociables denominados en moneda nacional y divisas.

En el BNC son muy empleados los Títulos Valores para determinar el pago de las negociaciones que se realizan. Por lo que es importante conocer que son los Títulos Valores, sus características principales y los tipos que existen. A continuación se definen algunos conceptos que se deben conocer para realizar la presente investigación.

1.2.2 Títulos Valores

El Título Valor o Título de Crédito le atribuye a un documento firmado por el deudor un valor determinado de carácter patrimonial. Este documento puede hacerlo efectivo el acreedor, en su oportunidad, para que de esta manera se cumpla la prestación establecida en el mismo (EcuRed, 2013). Son documentos mercantiles en los que está incorporado un derecho privado patrimonial, por lo que el ejercicio del derecho está vinculado jurídicamente a la posesión del documento (Barboza Parra, 1991).

Los Títulos Valores son medios de pago que se utilizan generalmente para pagar las Cartas de Crédito que entran y salen constantemente del BNC. Se paga el importe que en el mismo esté definido a la persona que posea este documento. Se emplean en la mayoría de las negociaciones que actualmente se realizan para acordar todo lo referente al pago de dichas negociaciones.

1.2.2.1 **Características de los Títulos Valores** (Barboza Parra, 1991)

- Literalidad del Título, lo cual implica el hecho ¨...en virtud del cual los derechos del poseedor se rigen, en cuanto a su extensión o modalidades, por el texto literal del documento, no pudiendo serle opuesto al poseedor nada que no esté allí expresado¨.
- Autonomía de los mismos, lo cual se traduce en el hecho de que ¨...el Título de crédito transmite en su circulación, a cada adquiriente un derecho desvinculado de la situación jurídica que tenía el transmitente¨.
- Legitimación, entendida como el derecho que asiste al poseedor de un título para exigir el pago de la obligación que él expresa.

1.2.2.2 **Clasificación de los Títulos Valores** (Rincones, 2003)

Títulos Valores a la orden: son aquellos que se extienden a favor de una persona determinada, pudiendo ésta transmitirlos a otra persona por medio de la fórmula del endoso. No es necesario notificar a la persona obligada al pago (deudor) la transmisión efectuada.

Títulos Valores al portador: son aquellos que reconocen un derecho a favor de la persona indeterminada que posea el documento. Se pueden transmitir por la mera entrega del documento a otra persona.

Títulos Valores nominativos: son aquellos que reconocen un derecho a favor de una persona determinada. Para la transmisión de estos, además de la entrega del documento, es necesaria notificar al emitente (deudor) para la inscripción de la misma en su libro registro de títulos.

Endosos en Títulos Valores

El endoso constituye uno de los requerimientos para la transferencia de cualquier Título Valor nominativo y a la orden. Mediante el endoso, el endosante transfiere su posición de tenedor legítimo del título, permitiendo al nuevo tenedor del título ejercer los derechos cambiarios en los términos que faculte el endoso (Moreno, 2005). Es la fórmula que permite al tenedor del título trasmitirlo a otra persona.

Los Títulos Valores que más se utilizan actualmente en el BNC son: Cheques, Chequeras, Cartas Remesas, Letras de cambio y Pagarés. A continuación se muestran las características principales de dos de estos Títulos Valores: Letras de cambio y Pagarés.

1.2.3 Letra de cambio

Se trata de un instrumento de cobro y pago no generado por una entidad financiera pero que se mantiene como uno de los instrumentos físicos más utilizados en el ámbito empresarial. Presenta características legales que facilitan la ejecución de la deuda en los tribunales (Gadea, 2007). Incorpora el derecho patrimonial consistente en la obligación de dar una suma determinada de dinero conforme a los sistemas de actualización o reajuste de capital legalmente admitidos. Es el modelo o paradigma de los Valores Negociables en general y de los Títulos Valores en particular (Francisco, 2011).

Las Letras de cambio son un instrumento o medio de pago utilizado para el pago de las negociaciones que se adquieren con otras instituciones. Deben ser pagados en el mismo lugar donde fue expedido o en un sitio diferente, siempre que esté expresado en el propio documento. Pertenecen a los Títulos Valores a la orden, que se gira a favor de determinada persona. Se transmite mediante el endoso y con la entrega del título a la persona que recibirá el dinero.

Personas que intervienen (Beaumont, 2010)

- Librador (emisor): es la persona acreedora de la deuda y quien emite la Letra de cambio para que el deudor o librado la acepte y se haga cargo del pago del importe de la misma.
- Librado (deudor): es quien debe pagar la deuda expresada en la Letra de cambio cuando llegue la fecha de pago.
- Beneficiario (el que cobrará): es la persona que tiene en su poder la Letra de cambio y a quien se le debe abonar.

Vencimientos (Beaumont, 2010)

- Vencimientos a la vista: implica que una vez que el beneficiario presenta la Letra de cambio al deudor esta debe ser pagada.
- Vencimiento a día cierto determinado: la realización de este pago es fijado en el texto de la Letra de cambio.
- Vencimientos a día cierto indeterminado: solo es posible cuando ocurre un hecho futuro como por ejemplo, la muerte de una persona.
- Vencimientos por ciertos sucesivos: también llamado por cuotas, se utiliza para realizar pagos periódicos determinados.

1.2.4 Pagaré

El Pagaré es un documento que consiste en la promesa pura y simple de pagar una determinada cantidad de dinero en un futuro a su legítimo tenedor. A partir de la emisión del mismo queda determinado el momento a partir del cual se podrá hacer efectivo su cobro. Es un Título Valor o instrumento financiero que es usado para obtener recursos monetarios. Pueden ser al portador o endosables, esto quiere decir, que se pueden transmitir a terceras personas (Gadea, 2007).

Un Pagaré es un documento privado que se desarrolla de forma legal y contiene la promesa incondicional de pagar una suma de dinero en el lugar y fecha determinados en el propio documento. En el Pagaré intervienen tres personas: el emisor, girador y beneficiario. El primero es quien confecciona el documento, el segundo es la persona obligada a pagar el documento cuando llegue la fecha de vencimiento y el tercero es la persona que tiene en su poder el Pagaré y quien recibirá el pago.

Vencimientos (Gadea, 2007)

- Vencimiento a la vista: una vez presentado el Pagaré al girado este debe pagar.
- Vencimiento a día cierto determinado: se fija la fecha en el mismo texto del documento.

- Vencimiento a día cierto indeterminado: es posible solamente cuando ocurre un hecho futuro como la muerte de una persona.
- Vencimiento por ciertos sucesivos: es llamado también por cuotas, se utiliza para realizar pagos en determinados períodos.

Luego de conocidos los principales conceptos y características de algunos Títulos Valores que permiten entender cómo son empleados en el BNC, se pasa al estudio realizado de los sistemas de gestión bancaria existentes a nivel nacional e internacional. Para esto se realizó una búsqueda general de varios sistemas bancarios, destacando las características, beneficios y deficiencias que presentan. Entre los sistemas informáticos de gestión bancaria estudiados se encuentra el Sistema Automatizado para Banca Internacional de Comercio (SABIC), que una de las soluciones empleadas en los bancos del país. Como soluciones extranjeras se obtuvieron el Easy Pagarés, Easy Letras y Letras de Cambio 2002 v2.05. A continuación se exponen los resultados obtenidos del estudio realizado.

1.2.5 Sistema Automatizado para la Banca Internacional de Comercio (SABIC)

El SABIC es el sistema encargado de la gestión de cuentas, clientes, Letras de cambio, transferencias, créditos, préstamo, negociación, conciliaciones, entre otros procesos contables. Es un sistema diseñado y desarrollado por la Dirección de Sistemas Automatizados del BCC para satisfacer las necesidades de procesamiento de datos de bancos e instituciones no bancarias (Perdomo Bello, 2009). Entre las características fundamentales del sistema están la contabilización multimoneda, la contabilización en tiempo real y la realización de transacciones. Partiendo de su diseño, ofrece fortaleza e integridad en la información que maneja y es flexible a los requerimientos de cada entidad (Meneses, y otros, 2010).

Presenta deficiencias al no presentar puntos de integración con el sistema utilizado para enviar y recibir los mensajes SWIFT[1] (SISCOM[2]). Como consecuencia de esto los operadores del banco deben introducir los mismos datos tanto en un sistema como en el otro; necesitando así un tiempo considerable y el doble de esfuerzo para realizar las operaciones bancarias que llevan consigo la generación y recepción de mensajes (Meneses, y otros, 2010). Está desarrollado sobre tecnologías obsoletas lo que proporciona un control irregular de la mayoría de los procesos que se ejecutan en los

[1]*Swift*: (en inglés: Society for World Wide Interbank Financial Telecommunication) es una organización que posee una red internacional de comunicaciones financieras entre bancos y otras entidades financieras.

[2]Sistema de Comunicación mensajería SWIFT usado entre los bancos a nivel mundial.

departamentos del banco. Además, no ofrece respuestas precisas y de gran complejidad como lo requieren las actividades bancarias que actualmente se realizan en el BNC (Perdomo Bello, 2009).

Luego de analizado este sistema bancario se puede llegar a la conclusión de que a pesar de las ventajas que propicia el SABIC y teniendo en cuenta las deficiencias que presenta no puede seguir siendo utilizado por el BNC. El mismo no cumple con los requerimientos de esta institución y no satisface en su totalidad todos los procesos que se ejecutan actualmente, ocasionado que los operadores bancarios realicen un trabajo de baja calidad e ineficiente. Por lo que surge la necesidad de buscar otro sistema que pueda ser empleado en el BNC.

1.2.6 Easy Pagarés (Admin, 2013)

Easy Pagarés es una aplicación especialmente desarrollada para la gestión y emisión de Pagarés totalmente configurable y adaptable al formato de cada banco. Entre sus características principales cabe destacar:

- Facilidad de uso.
- Posibilidad de adaptar el formato de impresión de Pagarés a los formatos vigentes de cada banco.
- Gestión e impresión de cheques y talones con control de cobro.
- Emisión de cartas informativas a proveedores con las facturas saldadas tras la emisión del pagaré.
- Gestión de vencimientos y fecha de cobro del pagaré por parte del beneficiario.

A pesar de las características que presenta este sistema informático, tiene una gran deficiencia y es que está desarrollado sobre tecnologías privativas. Lo que ocasiona que el estado cubano no pueda adquirir las licencias necesarias para ejecutar esta aplicación dentro del país, ya que Cuba está optando por la independencia tecnológica, es decir, el uso de tecnologías libres. Por lo que se imposibilita la utilización del mismo en el BNC.

1.2.7 Easy Letras (Administrador, 2013)

Easy Letras es una aplicación especialmente desarrollada para la gestión y emisión de Letras de cambio totalmente configurable y adaptada al formato oficial en vigor. Entre sus características principales cabe destacar:

- Facilidad de uso.

- Posibilidad de adaptar el formato de impresión de las Letras de cambio para cualquier impresora con capacidad de imprimir en formato sobre.
- Permite importar desde ficheros de texto (separados por tabuladores), los datos de clientes y empresas libradoras.
- Se encuentra disponible para las siguientes plataformas: Windows 9x, 2000 y XP.

A pesar de las características antes expuestas, este sistema presenta como principal deficiencia que está desarrollado sobre tecnologías privativas. Lo que trae consigo que el estado cubano no pueda adquirir las licencias necesarias para ejecutar esta aplicación dentro del país, porque Cuba está optando por la utilización de software libre. Por lo que se imposibilita la utilización de esta solución dentro del BNC.

1.2.8 Letras de Cambio 2002 v2.05

Letras de Cambio 2002 v2.05 es un sistema desarrollado para garantizar la gestión y emisión de Letras de cambio. La información que se puede almacenar en este sistema afecta tanto a las empresas que participan en la transacción, como a los clientes y bancos que aparecerán en la misma. Permite seleccionar diferentes tipos de monedas y modificar la apariencia del documento formal. Otras funciones de la aplicación son las siguientes: selección del tipo, tamaño y color de la fuente; además de guardar los últimos datos de la letra en memoria. Se puede generar múltiples Letras de cambio y se podrán imprimir de forma rápida y automatizada especificando los datos del cliente y los vencimientos a crear. Se encuentra disponible para las siguientes plataformas: Windows 98 o superior (Trevenque, 2005).

La principal desventaja que propicia este sistema es que está desarrollada sobre tecnologías privativas. Por lo tanto esta solución informática no puede ser aplicada en el BNC, porque Cuba está optando por la utilización de tecnologías libres. Por lo que se descarta la utilización de estos sistemas exportados y de esta manera se ahorra el presupuesto del país.

Luego de analizadas estas soluciones informáticas que tienen como objetivo principal la gestión de las Letras de cambio y Pagarés, se puede llegar a la conclusión de que no son los sistemas más factibles para el BNC. De aquí que se evidencia la importancia del desarrollo de un sistema de gestión bancaria, por lo que se obtiene la primera versión del sistema Quarxo. Con la puesta en marcha de esta aplicación se han logrado mejorar los procesos que se llevan a cabo en los departamentos del banco y ha facilitado la labor que realizan los operadores bancarios, logrando que se realice el trabajo con

mayor eficiencia. En aras de mejorar este sistema se está llevando a cabo una segunda versión, donde se le adicionarán funcionalidades que aún se realizan de forma manual como es: la gestión de las Letras de cambio y Pagarés, pertenecientes al subsistema Títulos Valores. Para lograr un mayor entendimiento de estos procesos se realizó un estudio de varias técnicas y actividades que permitieron obtener un resultado factible. A continuación se muestra el estudio realizado.

1.3 Ingeniería de Requisitos

1.3.1 Actividades de la Ingeniería de Requisitos

Para realizar el proceso de la disciplina Requisitos, existen varias técnicas estandarizadas que ofrecen un marco de desarrollo avanzado para garantizar la calidad del resultado. Estas técnicas proponen diferentes métodos para desarrollar aplicaciones web. La selección de las técnicas y los resultados que se obtengan, depende tanto del equipo de análisis y desarrollo, como de los clientes o usuarios que participan. El proceso de especificación de requisitos se puede dividir en tres actividades fundamentales:

1. **Captura de requisitos**

   Es el proceso de obtener los requisitos del sistema por medio de la observación de los sistemas existentes, discusiones con los usuarios potenciales y proveedores, el análisis de tareas. Esto puede implicar el desarrollo de uno o más modelos y prototipos del sistema que ayudan al analista a comprender el sistema a especificar (Sommerville, 2005).

2. **Especificación de requisitos**

   Es la actividad de traducir la información recopilada durante la actividad de análisis en un documento que define un conjunto de requisitos. En este documento se pueden incluir dos tipos de requisitos: los requisitos del usuario, que son declaraciones abstractas de los requisitos del cliente y del usuario final del sistema y los requisitos del sistema, que son una descripción más detallada de la funcionalidad a proporcionar (Sommerville, 2005).

3. **Validación de requisitos**

   Esta actividad comprueba la veracidad, consistencia y completitud de los requisitos. Durante este proceso, inevitablemente se descubren errores en el documento de requisitos. Se debe modificar entonces para corregir estos problemas (Sommerville, 2005).

1.3.2 Técnicas de captura de requisitos

Las técnicas de captura de requisitos son de gran utilidad porque permiten obtener con mayor facilidad los requisitos que debe cumplir el sistema. Ayudan a comprender qué es lo que necesita el cliente según sus necesidades y de esta manera se logra obtener una buena solución. Algunas de las técnicas de captura de requisitos que existen son las que a continuación se detallan:

Entrevistas: resulta una técnica muy aceptada dentro de la ingeniería de requisitos y su uso es ampliamente extendido. Permite al analista tomar conocimiento del problema y comprender los objetivos de la solución que se busca. Mediante esta técnica el equipo de trabajo se acerca al problema de una forma natural. Las entrevistas son esenciales en el proceso de la captura de requisitos, con su aplicación se puede obtener una amplia visión del trabajo y las necesidades del usuario (Escalona, 2002).

Tormenta de ideas: es una técnica de reuniones en grupo que tiene como objetivo que los participantes expresen sus ideas de forma libre. Es la mera acumulación de ideas o de información sin que sean evaluadas las mismas. El grupo de personas que participa en estas reuniones deben ser como máximo 10, siendo uno de ellos el moderador de la sesión, pero sin carácter de controlador. Es sencilla de usar y de aplicar como técnica, ya que no requiere tanto trabajo en grupo. Suele ofrecer una visión general de las necesidades del sistema, pero regularmente no sirve para obtener los detalles concretos del mismo, por lo que se suele aplicar sólo en los primeros encuentros (Escalona, 2002).

Mapa Conceptual: son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste (Escalona, 2002).

Casos de Uso: los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas, entre otros) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores. La ventaja principal es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se

acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades (Escalona, 2002).

Comparación de terminología: uno de los problemas que surge durante la elicitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste) (Escalona, 2002).

1.3.3 Técnicas de validación de requisitos

Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos (Escalona, 2002). Para esto se emplean varias técnicas las cuales facilitan en gran medida esta validación. Entre las técnicas existentes se encuentran las que a continuación se detallan:

Revisiones de requisitos: los requisitos son analizados sistemáticamente por un equipo de revisores. Una revisión de requisitos es un proceso manual que involucra a personas tanto de la organización del cliente como de la contratista. Las revisiones de requisitos pueden ser informales o formales (Sommerville, 2005).

Auditorías: esta técnica consiste en un chequeo de los resultados contra una Checklist predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada (Escalona, 2002).

Prototipos: algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (Escalona, 2002).

Generación de casos de prueba: los requisitos deben poder probarse. Si las pruebas para esto se conciben como parte del proceso de validación, a menudo revela los problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, normalmente significa que los requisitos serán difíciles de implementar y deberían ser considerados nuevamente. Desarrollar pruebas para los requisitos del

usuario antes que se escriba código es una parte fundamental de la programación extrema (Sommerville, 2005).

Matrices de trazabilidad: esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos (Escalona, 2002).

1.4 Diseño de Software

En el diseño se modela el sistema para que soporten todos los requisitos para que puedan ser tratados y administrados correctamente. Lo que contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima fase. Los artefactos generados en esta etapa son más formales y específicos de una implementación. En la elaboración de la solución es preciso tener presente ciertos estándares, estilos y patrones que son muy útiles en la obtención de un diseño de software robusto.

1.4.1 Arquitectura de Software

En el desarrollo de aplicaciones informáticas la arquitectura de software es un término muy frecuente y de gran referencia ya que sirve para guiar la implementación y el desarrollo del sistema desde etapas tempranas del diseño. Se refiere a la estructura del sistema, constituida por componentes de software, propiedades externamente visibles de esos elementos y las relaciones entre ellos (Rodríguez, 2008). Es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (Clements, 2002).

La arquitectura de software es la definición de todas las clases del sistema según la funcionalidad que tiene cada una de ellas. Es la agrupación de cada paquete con todos sus componentes y las relaciones de unos con otros. Es de manera general la estructura física que tendrá el sistema antes y después de implementado.

1.4.2 Descripción de la arquitectura a emplearse en el desarrollo del sistema

La segunda versión del sistema Quarxo estará desarrollada con tecnologías y componentes Open Source (código abierto), en la plataforma Java. La aplicación será multiplataforma, soportando sistemas Unix y Windows y se desplegará en un entorno Web, todo esto al igual que en la primera versión. La solución del sistema se comunicará con otros sistemas por diferentes vías, tanto por servicios web, mensajería, invocación remota u otros tipos de comunicación. En la seguridad se

utilizará el protocolo de seguridad Secure Socket Layer (SSL, por sus siglas en inglés) para la transmisión de los datos, utilizando la encriptación de la información. Para evitar la pérdida de información cuando no exista conectividad en el instante que se envíe alguna información se utilizará la mensajería asíncrona (Iglesias, 2009). Bajo toda esta arquitectura serán desarrollados los módulos Letra de cambio y Pagaré del subsistema Títulos Valores en una segunda versión del sistema bancario Quarxo. Esta fue la arquitectura definida por el equipo de desarrollo del proyecto Banco al realizar un estudio a inicios de la primera versión y será la misma que se empleará para dar solución al problema planteado.

1.5 Patrones

1.5.1 Patrones de Arquitectura

Los patrones de arquitectura están relacionados fundamentalmente con la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes. Describen un problema particular y recurrente del diseño, que surge en un contexto específico y presenta un esquema genérico y probado de su solución (Veloso, 2004).

Patrón Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres capas o componentes distintos. El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML (*del inglés, Hypertext Markup Language*) que se visualiza en el navegador y el código que proporciona de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista, enviarlos al modelo y manejar también las respuestas del modelo y transmitirlas hacia la vista (Pantoja, 2004).

Este patrón propone dividir la aplicación en tres capas el Modelo, la Vista y el Controlador. El modelo es la representación del dominio o datos del sistema, la vista es la encargada de presentar la interfaz al usuario, en sistemas web. El controlador es el que decide qué información es la que se envía a la vista, es el encargado de escuchar los cambios en la vista y enviarlos al modelo, el cual le regresa los datos a la vista como un ciclo.

1.5.2 Patrones de Diseño

Los Patrones de Diseño son soluciones simples a problemas específicos y comunes del diseño orientado a objetos. Son soluciones que se basan en la experiencia y que se ha demostrado que funcionan. Son descripciones de clases cuyas instancias colaboran entre sí y brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares (Veloso, 2009).

Patrones GRASP

Los Patrones Generales para Asignar Responsabilidades: GRASP (en inglés: General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones, su nombre se debe a la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos. GRASP destaca 5 patrones principales: Experto, Creador, Bajo acoplamiento, Alta cohesión, Controlador (Larman, 1999), (Shalloway, 2004).

- **Experto:** consiste en asignar una responsabilidad al experto en información, es decir, a la clase que cuenta con la información necesaria para cumplir con la responsabilidad. Se refuerza el encapsulamiento y se favorece el bajo acoplamiento.
- **Creador:** guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Su propósito principal es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Brinda soporte al bajo acoplamiento.
- **Controlador:** este patrón consiste en asignar la responsabilidad a una clase de manejar los mensajes correspondientes a eventos de un sistema. El controlador es un intermediario entre la interfaz de usuario y el núcleo de las clases donde reside la lógica de la aplicación. Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación para aumentar la reutilización de código y a la vez tener un mayor control.
- **Bajo acoplamiento:** pretende asignar una responsabilidad para mantener el bajo acoplamiento, es decir, el diseño de clases más independientes, que no se relacionen con muchas otras, que reducen el impacto de los cambios, que son más reutilizables y acrecienten la oportunidad de una mayor productividad.
- **Alta cohesión:** su objetivo es asignar una responsabilidad, de modo que la cohesión siga siendo alta. Las clases con alta cohesión se caracterizan por tener responsabilidades

estrechamente relacionadas y no realizar un trabajo enorme. Una clase con alta cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño.

Patrones GoF

El grupo de GoF (*del inglés, Gang of Four*) clasificaron los patrones en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento (Shalloway, 2004), (Gamma, y otros, 1995).

- **Creacionales**: los patrones creacionales se relacionan con las formas de crear las instancias de los objetos. Su objetivo es abstraer el proceso de instanciación y a su vez ocultar los detalles de cómo los objetos son creados o inicializados.
- **Estructurales**: describen cómo las clases y objetos pueden ser combinados para formar estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- **Fachada**: patrón estructural que trata de simplificar la interfaz entre dos sistemas o componentes de software ocultando un sistema complejo detrás de una clase que funciona como pantalla o fachada. La idea principal es la de ocultar todo lo posible la complejidad de un sistema, el conjunto de clases o componentes que lo definen, de forma que solo se ofrezca un punto de entrada al sistema cubierto por la fachada. Su uso aísla los posibles cambios que se puedan producir en alguna de las partes.
- **Cadena de Responsabilidad**: se encarga de evitar el acoplamiento del remitente de una petición a su receptor, dando a más de un objeto la posibilidad de manejar la petición.
- **Comportamiento**: ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

1.6 Tecnologías y herramientas empleadas en la solución

Para el desarrollo de la primera versión del sistema Quarxo se emplearon las tecnologías y herramientas que a continuación se describen. Las mismas fueron seleccionadas por el equipo de trabajo del proyecto Banco luego de un estudio realizado. Fueron elegidas ya que cumplen con las expectativas de desarrollar software libre y de esta manera se cumple con las políticas trazadas en el BNC de abogar por la utilización de tecnologías libres.

1.6.1 Lenguaje de modelado

Lenguaje Unificado de Modelado (UML) 2.0

Es uno de los lenguajes de modelado de procesos más conocidos y utilizados en la actualidad. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para describir un "plano" del sistema, incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, detallar los artefactos en el sistema, documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo (Grau, y otros, 2008).

1.6.2 Herramienta CASE. Visual Paradigm

CASE es un acrónimo para Computer-Aided Software Engineering, es una herramienta que ayuda al ingeniero de software a desarrollar y mantener software. Es una combinación de herramientas de software y metodologías de desarrollo. Es definido como herramientas individuales para ayudar al desarrollador de software o administrador de proyecto durante una o más fases del desarrollo de software (o mantenimiento) (Bergin, 1989).

Visual Paradigm es una herramienta de modelado visual para todo tipo de diagramas UML. Es compatible con la gestión extensiva de casos de uso, diagramas de SysML (Systems Modeling Language), requisitos y el diseño de bases de datos (con ERD, por su sigla en inglés, Entity-Relationship Diagrams) y entrega los esfuerzos más eficaces en el análisis y diseño de sistemas para UML (32). La versión empleada durante el desarrollo de los artefactos que contiene el flujo de trabajo de Análisis y Diseño fue Visual Paradigm 5.0 para UML (Paradigm, En Línea).

1.6.3 Notación para el Modelado de Procesos de Negocio (BPMN)

Es una notación gráfica estandarizada que permite el modelado de procesos de negocio. El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio (stakeholders). Entre estos interesados están los analistas de negocio (quienes definen y redefinen los procesos), los desarrolladores técnicos responsables de implementar los procesos) y los gerentes y administradores del negocio (quienes monitorizan y gestionan los procesos). En síntesis BPMN tiene la finalidad de servir como lenguaje

común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación (Rodríguez, En Línea). La versión empleada para obetener la solución de la presente investigación fue BPMN 1.2.

1.6.4 Leguaje de Programación. Java

Java es un lenguaje sencillo orientado a objetos, es independiente de plataforma, por lo que un programa hecho en Java se ejecutará igual en un PC con Windows que en una estación de trabajo basada en Unix. Su capacidad multihilo y su robusta integración que tiene con el protocolo TCP/IP, lo hace un lenguaje ideal para Internet (Java, En Línea). La versión utilizada en el desarrollo del producto Quarxo fue Java (TM) 6 Update 20 versión: 6.0.200.

1.6.5 Entorno de Desarrollo Integrado (IDE): Eclipse

El Eclipse es una plataforma que ha sido diseñada desde el principio en la creación web integrando herramientas para el desarrollo de aplicaciones. El valor de la plataforma es fomentado por el desarrollo rápido de las funciones integradas y basadas en un modelo de plugin[3]. Proporciona una interfaz común de usuario (UI) de modelo para trabajar con herramientas. Está diseñado para ejecutarse en múltiples sistemas operativos, al mismo tiempo que ofrece una integración sólida con cada sistema operativo subyacente (Eclipse, En línea). La versión utilizada en el desarrollo del producto Quarxo fue Eclipse 3.3.

1.6.6 Apache Tomcat 6.0

Apache Tomcat es un contenedor web basado en el lenguaje Java que actúa como motor de Servlets y Java Server Pages (JSPs). Se ha convertido en la implementación de referencia para las especificaciones de Servlets y JSPs. Está desarrollado en un entorno abierto Open Source (Apache, En Línea). Durante el desarrollo de Quarxo se utilizó la versión Tomcat 6.0.

1.6.7 Marco de Trabajo (Framework)

1.6.7.1 Spring Framework

Spring Framework es libre y está compuesto por un conjunto de módulos, de los cuales se pueden tomar aquellos que faciliten la implementación del trabajo en cuestión. El centro de Spring está basado en el principio de Inyección de Dependencias. Esta técnica hace externa la creación y el manejo de las

[3]Plugin: módulo de hardware o software que adiciona una característica o un servicio específico a un sistema existente.

dependencias de las clases, con lo que se logra una mayor limpieza y claridad en el código. (Clara, y otros, 2003) Se utiliza la versión 2.0.6 en el desarrollo de Quarxo.

1.6.7.2 Hibernate

Hibernate es un marco de trabajo para resolver el problema de persistencia de datos en Java, entre la aplicación y la base de datos relacional, dejando al desarrollador concentrarse en los problemas de la aplicación. La integración de Hibernate es sencilla con aplicaciones anteriores y nuevas. Resumiendo es un marco de trabajo Java para el Mapeado Objeto/Relacional (Object/Relational Mapping, ORM por sus siglas en inglés) (Christian, 2005). Se utiliza en el desarrollo de Quarxo la versión 3.5.

1.6.7.3 Dojo Toolkit

DOJO es un marco de trabajo de JavaScript que ofrece muchos instrumentos, como son controles del interfaz de usuario, efectos, utilidades comunes; es una API para gestionar el acceso asíncrono al servidor (Dojo, En línea). Para el desarrollo del producto Quarxo se utiliza la versión 1.3.

1.6.8 Servidor de Base de Datos: Microsoft SQL Server 2005

Microsoft SQL[4] Server 2005 es un servidor de base de datos basado en el modelo relacional. Se caracteriza por brindar soporte para transacciones y procedimientos almacenados y posee como lenguajes de consulta al SQL y al T-SQL (en inglés: Transact-SQL). Permite además trabajar en modo cliente-servidor y promueve la escalabilidad y seguridad de la información (Dhingar, y otros, 2007). Este servidor de bases de datos fue utilizado a petición del cliente.

1.6.9 Modelo de Desarrollo de Software

El modelo de desarrollo de software empleado es el del Centro de Informatización de Entidades (CEIGE) versión 1.1, el cual describe los pasos que se deben seguir para la producción de los productos informáticos que se desarrollan en los proyectos que pertenecen a este centro. En este modelo de desarrollo se especifican las actividades que se realizan en todo el ciclo de vida de los proyectos, además de que se detallan los artefactos que se deben generar en cada momento independientemente de las herramientas que se utilicen para ello.

[4] (Structure Query Language, Lenguaje Estructurado de Consultas por sus siglas en español)

1.6.9.1 Descripción de las fases del ciclo de vida de los proyectos (Obregón, 2012)

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

Los objetivos de la fase son:

- Asegurar la factibilidad del proyecto.
- Establecer un plan para la ejecución del proyecto.

Hitos:

- Plan de desarrollo de software.
- Acta de inicio del proyecto firmada.

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

El objetivo de esta fase es:

- Obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales.

Hito:

- Producto liberado por entidad certificadora de calidad.

En esta fase se ejecutan las disciplinas modelado de negocio, requisitos, análisis y diseño, implementación, pruebas internas y pruebas de liberación.

1.6.9.2 Ciclo de vida de proyectos del CEIGE

El ciclo de vida de los proyectos del CEIGE tiene en cuenta las actividades de cada una de las fases y áreas de procesos que plantea el nivel dos de CMMI establecido en la UCI. Esta abarca el total de acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o

producto final, sin embargo, se debe adaptar a las características particulares del proyecto que puede que no ejecute determinada disciplina, así como la elaboración de determinados artefactos del total aquí definido (Obregón, 2012).

El ciclo de vida de los proyectos que pertenecen al centro CEIGE está estructurado de la siguiente manera:

- Estudio Preliminar.
- Modelado del negocio.
- Requisitos.
- Análisis y diseño.
- Implementación.
- Pruebas internas.
- Pruebas de liberación.

1.7 Conclusiones Parciales

Al finalizar el capítulo se puede llegar a las siguientes conclusiones:

- Se vuelve difícil desde el punto de vista económico para el país costear cualquiera de los sistemas informáticos ya existentes a nivel internacional para la gestión de las Letras de cambio y los Pagarés.

- El SABIC que es uno de los sistemas informáticos nacionales que se utiliza en los bancos del país, no tiene implementado la gestión de las Letras de cambio y Pagarés, además que está desarrollado sobre tecnologías obsoletas.

- Se evidencia la necesidad de realizar una segunda versión del sistema Quarxo de manera que permita gestionar todos los Títulos Valores que se utilizan en el BNC y de esta manera facilitar el trabajo de los operadores bancarios.

- En correspondencia con lo definido por el proyecto Banco se utilizará el modelo de desarrollo de software de CEIGE, el cual define los artefactos que se deben obtener en cada fase de desarrollo.

- Se utilizará UML como lenguaje de modelado, Visual Paradigm como herramienta CASE, Java como lenguaje de programación. Los marcos de trabajo que se utilizarán son: Spring, como

núcleo de la aplicación, Hibernate para la persistencia de datos y Dojo Toolkit, para decorar las interfaces de usuario. Como herramienta de desarrollo será usado Eclipse, como gestor de base de datos SQL Server 2005 y como contenedor web el Apache Tomcat. Todo esto fue seleccionado por el proyecto banco luego de un estudio realizado.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

2.1 Introducción

En este capítulo se realiza una descripción del análisis y el diseño, dejando claro lo que se realiza en cada una de estas fases. Se describen los procesos de negocios que se llevan a cabo en el departamento de negociaciones del BNC. Se definen los artefactos propuestos por el modelo de desarrollo del CEIGE para lograr alcanzar la solución. Se especifican los patrones de arquitectura y de diseño utilizados en la concepción de la solución y se valida el diseño mediante la métrica Tamaño Operacional de Clase (TOC).

2.2 Modelado de Negocio

El modelado de negocio es la fase destinada a comprender los procesos de negocio de la organización cliente. Se entiende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. En esta fase se obtuvo el modelado de los procesos que se desean informatizar en el departamento de negociaciones del BNC. A continuación se describen los procesos Letra de cambio y Pagaré para lograr un mayor entendimiento del negocio y de esta manera posibilitar el desarrollo de un software con la mayor calidad posible.

2.2.1 Descripción del proceso Letra de cambio

El diagrama de proceso de Letra de cambio (Ver Fig. 1), muestra cómo se realiza el trabajo actualmente con las Letra de cambio en el BNC. El proceso se inicia cuando llega una nueva negociación al departamento de negociaciones, el negociador recibe esta nueva negociación, la contabiliza y luego confecciona la Letra de cambio. La supervisora de esta área procede a revisar la Letra de cambio en busca de algún error, si el documento está correcto se firma y se envía a la empresa (librado), para que proceda a firmar la letra. La empresa recibe la letra, lo firma y la envía nuevamente al banco. En el banco es recibida por la secretaria del departamento de negociaciones, la misma es la encargada de ponerla al cobro y lo archiva hasta que se cumpla el plazo de pago. En caso de que exista algún error cuando la supervisora revisa el documento de Letra de cambio, esta procede a desechar dicho documento e informarle al negociador para que este lo confeccione nuevamente y a partir de ahí se realizan los mismos pasos antes mencionados.

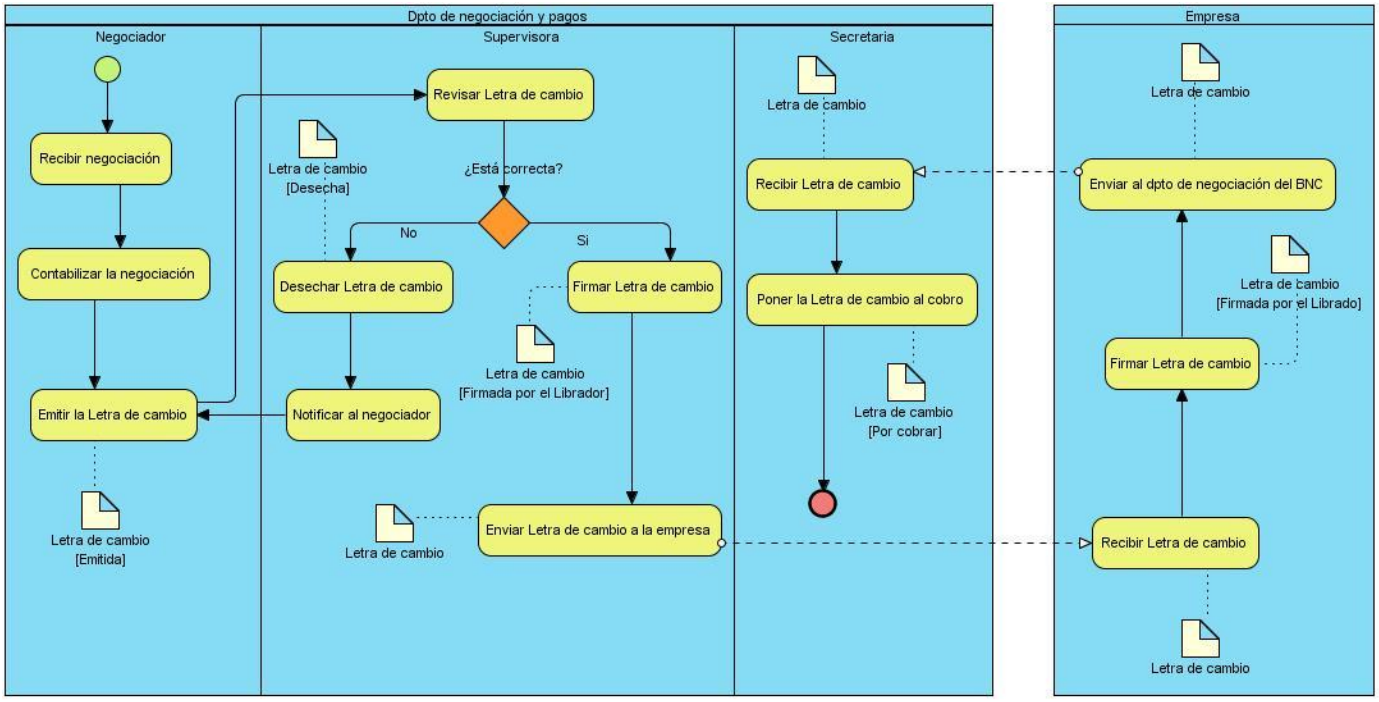


Fig. 1 Proceso Letra de cambio

2.2.2 Descripción del proceso Pagaré

El diagrama del proceso Pagaré (Ver Fig. 2), muestra el trabajo que se realiza con este medio de pago en el banco. El proceso se inicia cuando el negociador del departamento de negociaciones hace todos los trámites para obtener los documentos de embarque de una nueva negociación, luego procede a elaborar el borrador del Pagaré de forma digital (formato Word). La supervisora de esta área se encarga de la revisión de este documento, si está correcto le notifica al negociador para que elabore el Pagaré en el papel de seguridad. Luego este documento es enviado hacia el departamento jurídico donde es recibido por los jurídicos del banco, los que proceden a validarlo mediante la firma y el cuño correspondiente. Cuando ha sido aprobado se envía nuevamente al departamento de negociaciones donde es recibido por la secretaria, que es la encargada de enviar el Pagaré hacia el país con el cual se realiza la negociación. En caso de que la supervisora detecte algún error en el borrador del Pagaré, este se lo notifica al negociador, el cual tiene que corregirlo y luego elaborar el documento en el papel de seguridad, a partir de este paso se realizan todos los demás antes descritos.

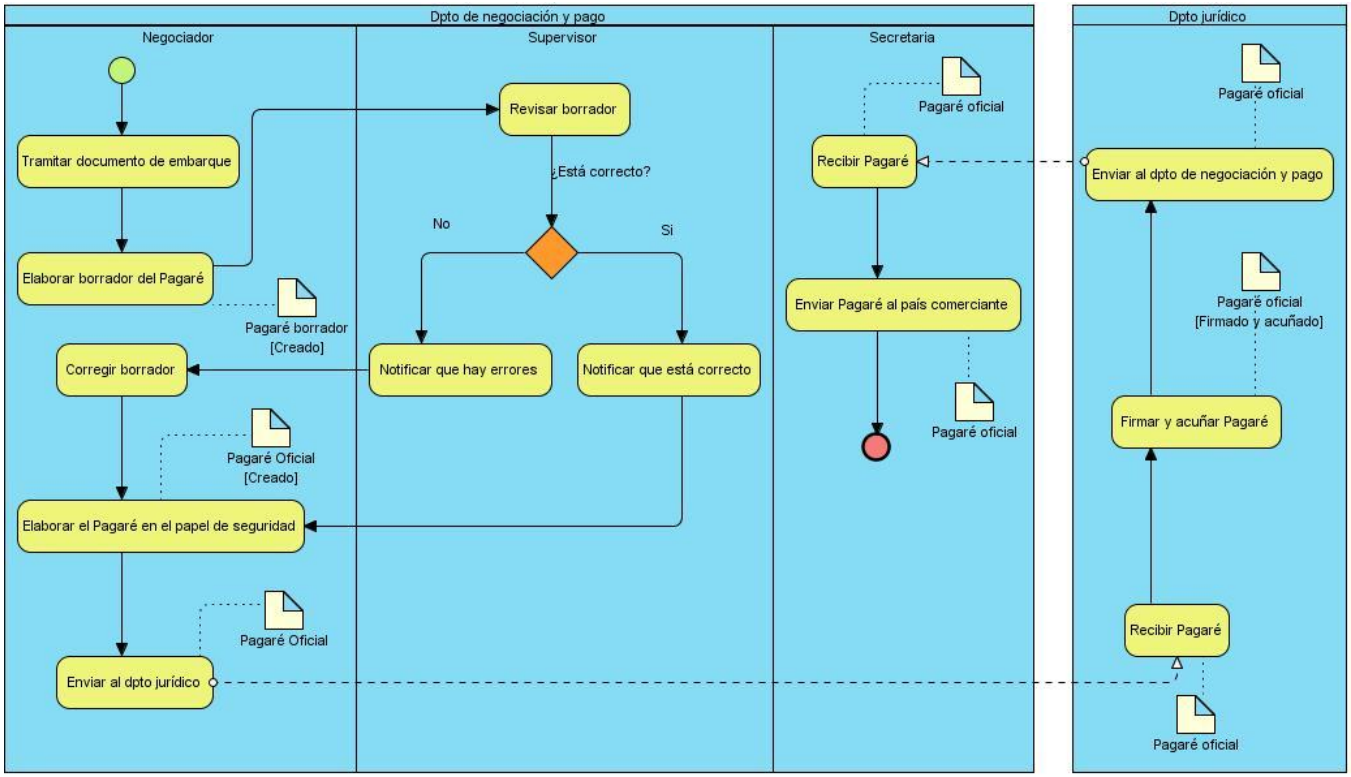


Fig. 2 Proceso Pagaré.

Análisis crítico de la ejecución de los procesos

Los negociadores del departamento de negociaciones presentan como principal deficiencia la incapacidad de poder elaborar y almacenar en tiempo la totalidad de las operaciones que realizan, entre ellas se encuentra la gestión de Letras de cambio y Pagaré, debido a que se ejecutan de manera manual, lo que ocasiona que ocurran errores humanos y con ello pérdida de tiempo y recursos materiales. Además, se vuelve un problema a la hora de archivar tantos documentos en formato duro, lo que ocasiona que se pierdan o se deterioren algunos de ellos, de igual manera si se desea buscar uno en específico se pierde mucho tiempo buscando entre tantos documentos. Todo esto provoca que en numerables ocasiones la información no exista en el momento preciso para apoyar a la toma de decisiones oportuna de la entidad.

2.3 Especificación de los requisitos del sistema

El esfuerzo principal en la fase de Requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de artefactos que describen todas las interacciones que tendrán los

usuarios con el software y que responden a los requisitos funcionales del sistema. Se especifican todo los requisitos funcionales del sistema (Obregón, 2012).

2.3.1 Técnicas para la Captura de Requisitos

Se combinan las siguientes técnicas de captura de requisitos: entrevistas y tormenta de ideas. Se entrevistó a la supervisora del departamento de negociaciones y a la directora de esta área conjuntamente. Esto se realizó mediante preguntas (Ver Anexo 1) que ayudaron a esclarecer con precisión el funcionamiento de todo el trabajo de esta área. Con el resultado de las preguntas realizadas se conoció en detalles cuál eran los procesos que se debían informatizar, qué es lo que se hace en cada uno de estos procesos y cuál era el resultado que ellos como clientes necesitaban. La segunda técnica de captura empleada fue la tormenta de ideas, la cual fue dirigida por la supervisora del departamento de negociaciones. Para esto se realizaron varios encuentros con negociadores y directivos del área, los cuales aportaron las ideas que tenían al respecto y los resultados que ellos necesitaban, estas ideas fueron tomadas en cuenta para la obtención de la solución final.

2.3.2 Requisitos Funcionales

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, es la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (Pressman, 2005).

Con la utilización de las técnicas de captura de requisitos y los métodos científicos empleados se identificaron un total de 11 requisitos funcionales los cuales fueron asociados con la propuesta de solución y son los que se muestran seguidamente:

Gestionar Letra de cambio

El sistema permitirá emitir, buscar, actualizar, consultar, cancelar y pagar una Letra de cambio en el sistema. A continuación se muestra una tabla con los requisitos y una breve descripción de los mismos (Ver Tabla 1).

Tabla 1: Requisitos funcionales del Gestionar Letra de cambio.

Nº	Funcionalidad	Descripción	Complejidad
[RF1.]	Emitir Letra de cambio	Se registra la Letra de cambio en el sistema.	Media.

[RF2.]	Buscar Letra de cambio	Buscan las Letras de cambio en el sistema, según los campos seleccionados.	Media.
[RF3.]	Actualizar Letra de cambio	Actualiza la Letra de cambio seleccionada.	Media.
[RF4.]	Consultar Letra de cambio	Consulta la Letra de cambio seleccionada.	Baja.
[RF5.]	Cancelar Letra de cambio	Se cambia el estado de la Letra de cambio a cancelada.	Baja.
[RF6.]	Pagar Letra de cambio	Se cambia el estado de la Letra de cambio a pagada.	Media.

Gestionar Pagaré

El sistema permitirá registrar, buscar, actualizar, consultar e imprimir un Pagaré en el sistema. A continuación se muestra una tabla con cada uno de estos requisitos y una breve descripción de los mismos (Ver Tabla 2. Requisitos funcionales del Gestionar Pagaré.).

Tabla 2. Requisitos funcionales del Gestionar Pagaré.

Nº	Funcionalidad	Descripción	Complejidad
[RF1.]	Registrar Pagaré	Se registra el Pagaré en el sistema.	Media.
[RF2.]	Buscar Pagaré	Buscan los Pagarés en el sistema, según los campos seleccionados.	Media.
[RF3.]	Actualizar Pagaré	Actualiza el Pagaré seleccionado.	Media.
[RF4.]	Consultar Pagaré	Consulta el Pagaré seleccionado.	Baja.
[RF5.]	Imprimir Pagaré	Se imprime el Pagaré seleccionado.	Baja.

A continuación se muestra la descripción textual del requisito Emitir Letra de cambio (Ver Tabla 3), para el resto de las descripciones (Ver Anexo 2).

Tabla 3. Descripción textual del requisito Emitir Letra de cambio.

Precondiciones	**El usuario se ha identificado y autenticado ante el sistema y tiene permisos para ejecutar esta acción.** **Debe existir un número consecutivo de serie de la Letra de cambio en el sistema.**
Flujo de eventos	

Flujo básico Emitir Letra de cambio	
1	El usuario introduce los datos requeridos de la Letra de cambio: Número de serie Fecha de emisión Importe Ordenante Moneda Cuenta del librado Librado Sucursal Banco Plaza Vencimiento de la letra Vencimiento de la negociación Referencia original Referencia corriente País
2	El usuario presiona el botón Aceptar.
3	El sistema valida los datos introducidos.
4	Si los datos son correctos el sistema los emite.
5	El sistema muestra el mensaje: “Operación realizada satisfactoriamente”.
6	El usuario presiona Aceptar en el mensaje.
7	Concluye el requisito.
Pos-condiciones	
1	Se emitió en el sistema una nueva Letra de cambio.
Flujos alternativos	
Flujo alternativo 4.a Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A.
Flujo alternativo 4.b Información incompleta	
1	El sistema señala los datos vacíos y permite llenarlos.
2	El usuario llena los campos vacíos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A.
Flujo alternativo 1.a El usuario cancela la acción	
1	Concluye el requisito.
Pos-condiciones	
1	N/A.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual: CIG-QF2-N-TV-i1301.

2.3.3 Validación de requisitos

Para validar los requisitos se emplearon tres técnicas de validación de requisitos, las mismas son: prototipos, revisiones de requisitos y generación de casos de prueba. Para esto se efectuó la construcción de prototipos (Ver Fig. 3) (Ver Anexo 4 y Anexo 5) por cada requisito funcional, de forma tal que se visualizaran los diferentes datos que tendría cada una de las interfaces. Así el usuario pudo tener una idea de la estructura de la interfaz final del sistema. La segunda técnica fue empleada al realizarse la descripción de cada uno de los requisitos funcionales (Ver Tabla 1) (Ver Anexo 2 y Anexo 3) los cuales fueron revisados y analizados sistemáticamente por el analista principal del proyecto, también fueron revisados por el cliente, el cual firmó cada uno de ellos (Ver Anexo 6). La tercera y última técnica fue empleada al generarse casos de pruebas por cada uno de los requisitos funcionales, para que los mismos fueran probados en la etapa de pruebas. A continuación se muestra uno de los prototipos diseñados, el mismo corresponde al RF Emitir Letra de cambio (Ver Fig. 3).

Fig. 3 Prototipo del RF Emitir Letra de cambio.

2.4 Diseño de las capas lógicas

Para ganar en organización en el desarrollo y en el despliegue del sistema, se agrupan los módulos y componentes por subsistema, cada subsistema tiene uno o más módulos y/o componentes estrechamente relacionados con las funcionalidades que ejecutan. Los módulos y/o componentes están separados por diferentes capas lógicas según la naturaleza de los mismos (Iglesias, 2008). Las capas lógicas definidas son las que se muestran a continuación (Ver Fig. 4).

- Capa de Presentación: lógica de presentación.
- Capa de Negocios: fachada y lógica de negocio.
- Capa de Acceso a Datos: desarrollo de Daos. Clases para el acceso a la base de datos.
- Capa de Dominio: dominios o entidades persistentes.

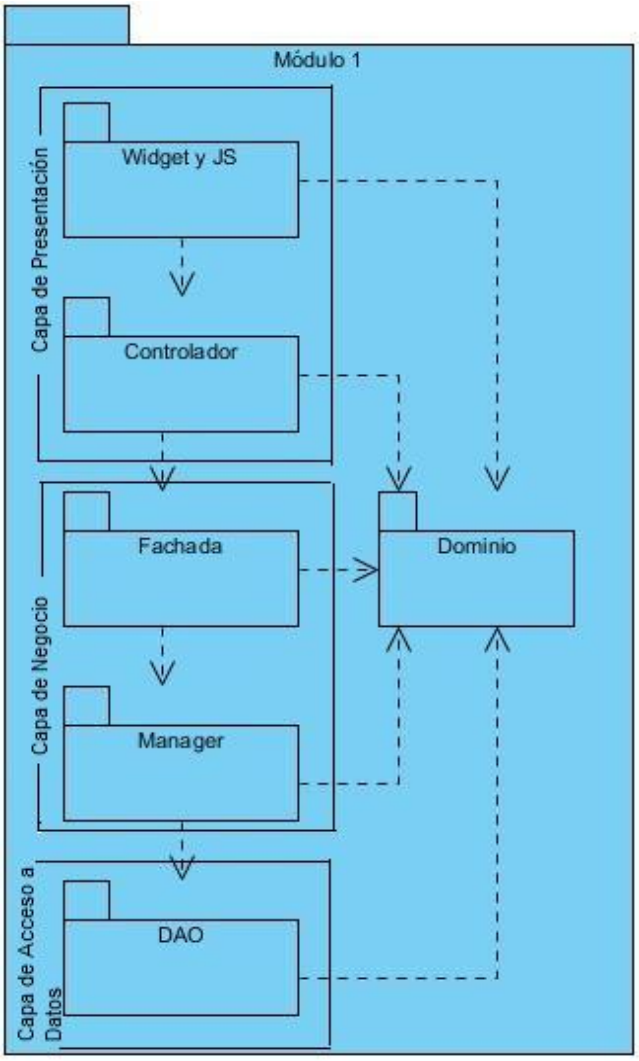


Fig. 4 Estructura de las capas lógicas del sistema.

Capa de Presentación:

Esta capa está dividida en dos partes. Una subcapa del lado del servidor (Controlador) (Ver Fig. 5), encargada de recibir todos los pedidos de la interfaz de usuario, controlar el flujo de presentación del sistema y enviar las respuestas correspondientes a la interfaz de usuario. La otra subcapa estará en el cliente (Widget y JS), utilizándose en los componentes visuales de JavaScript para manejar los eventos y validaciones del lado del cliente. La subcapa colocada en el lado del servidor estará relacionada con la capa de Negocios y con la capa transversal de Dominio (Ver Fig. 6 y Fig. 7) (Iglesias, 2008).

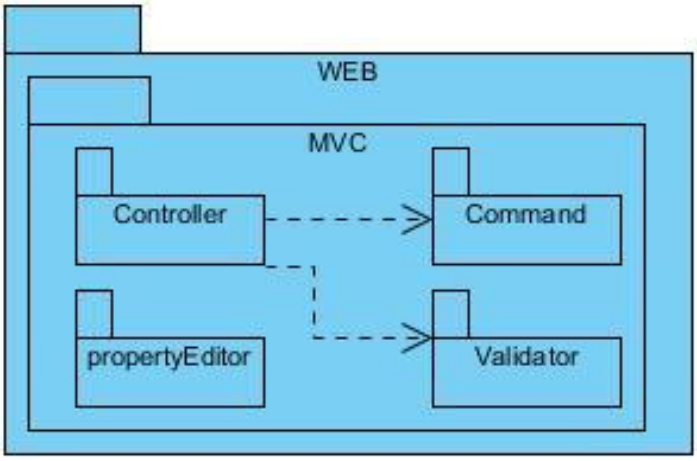


Fig. 5 Capa de presentación del lado del servidor.

En la Fig. 5 se muestran las clases que se desarrollan sobre Spring MVC. Este sub-paquete tendrá en principio los siguientes sub-paquete:

- propertyEditor: clases que convierten datos (ejemplos de tipos de datos: banco, moneda, país) en objetos determinados y que se relaciona con las clases controladoras y con la Fachada.
- Validator: clases para desarrollar las validaciones en el servidor.
- Command: clases para mapear datos que se introducen desde el cliente a objetos del negocio.
- Controller: clases que heredan de las clases “Controller” que brinda Spring MVC.

Capa de Dominio:

En esta capa se declaran todas las clases que representan entidades del negocio. Estas clases de dominio están presentes en todas las capas anteriormente descritas (Iglesias, 2008). A continuación se muestra una imagen de esta capa (Ver Fig. 6).

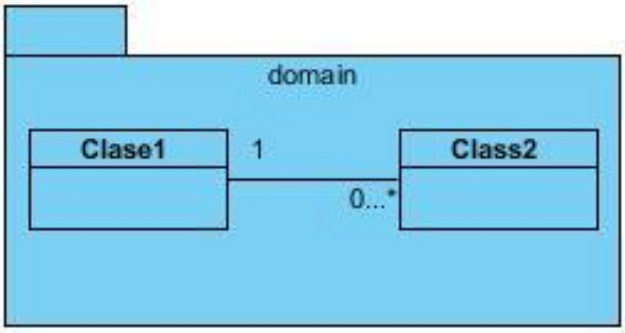


Fig. 6 Capa de Dominio.

Capa de Negocio:

Esta capa está divida en dos subcapas principales: Facade (fachada) y Manager (desarrollo del negocio), sin dejar de incluir otras que se necesiten y que estén relacionadas con el negocio. En la fachada se exponen todas las funcionalidades que la capa de presentación necesita. Esta capa invoca métodos de la subcapa de desarrollo del negocio. En la capa de desarrollo del negocio se implementa el negocio de los módulos en cuestión y de aquí se accede de ser necesario a la capa de acceso a datos (Ver Fig. 8), a otras capas de negocios y/o a la capa de dominio (Ver Fig. 6) (Iglesias, 2008). A continuación se muestra una imagen de la capa de negocios (Ver Fig. 7).

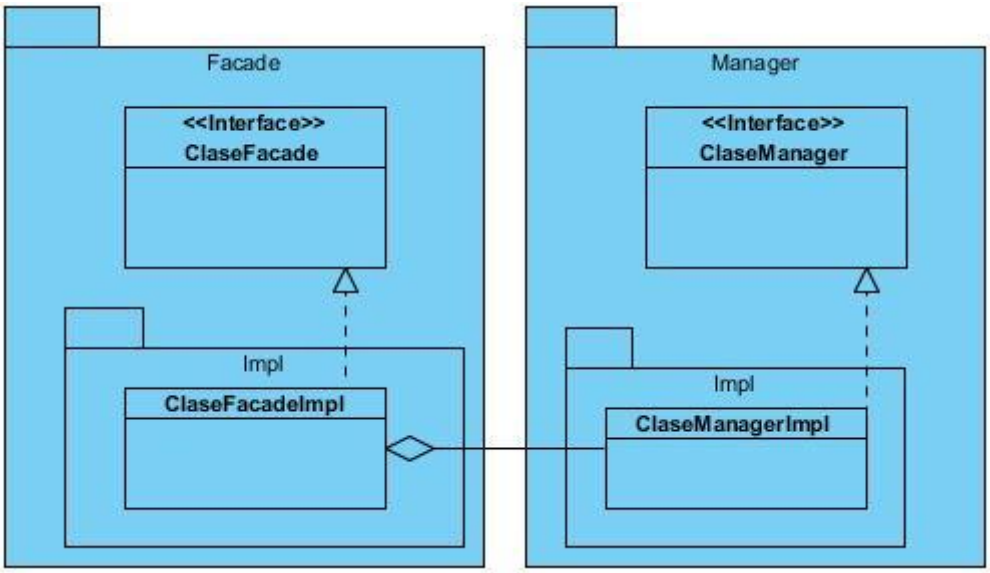


Fig. 7 Capa de negocio.

Capa de Acceso a Datos:

En esta capa se implementarán los métodos encargados de interactuar con el gestor de Base de Datos. Esta capa tendrá solamente dependencia con la capa de dominio (Ver Fig. 6) (Iglesias, 2008). A continuación se muestra una imagen de la capa de acceso a datos (Ver Fig. 8).

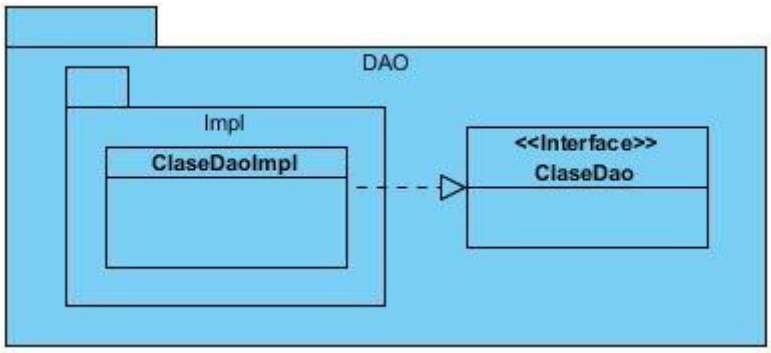


Fig. 8 Capa de Acceso a Datos.

2.5 Diseño del subsistema Títulos Valores

A partir de los resultados obtenidos en la fase de análisis se comienza con el desarrollo de los artefactos de la fase de diseño. Se modela el sistema y se obtiene su forma (incluida la arquitectura) para que soporte todos los requisitos –incluyendo los requisitos no funcionales y otras restricciones- que se le suponen. El diseño proporciona una comprensión detallada los requisitos e impone una estructura del sistema que se pretende conservar lo más fielmente posible cuando se da forma al sistema (Jacobson, y otros, 2000).

El diseño es una de las principales etapas del desarrollo de software, en este se sientan las bases para posteriormente entrar en la etapa de implementación. Con un buen diseño elaborado será más fácil para el programador llevar cada clase al código. Además, es factible a la hora de realizar cambios en el sistema ya que se contará con diagramas de diseño que permitirán observar con presición el conjunto de clases y componentes con sus funcionalidades y de esta manera comprender mejor cada detalle del sistema.

2.5.1 Diagrama de paquetes

Durante el desarrollo de software resulta muy conveniente agrupar clases y ficheros por diferentes criterios para lograr la organización y facilitar la comprensión del código de la aplicación, resultando conveniente el desarrollo de los diagramas de paquetes, los cuales muestran cómo está dividido el sistema en agrupaciones lógicas mostrando las dependencias entre las mismas (Jacobson, y otros, 2000). En la Fig. 9 se ilustra el diagrama de paquetes del módulo Letra de cambio y el del módulo Pagaré se puede ver en los anexos (Ver Anexo 7).

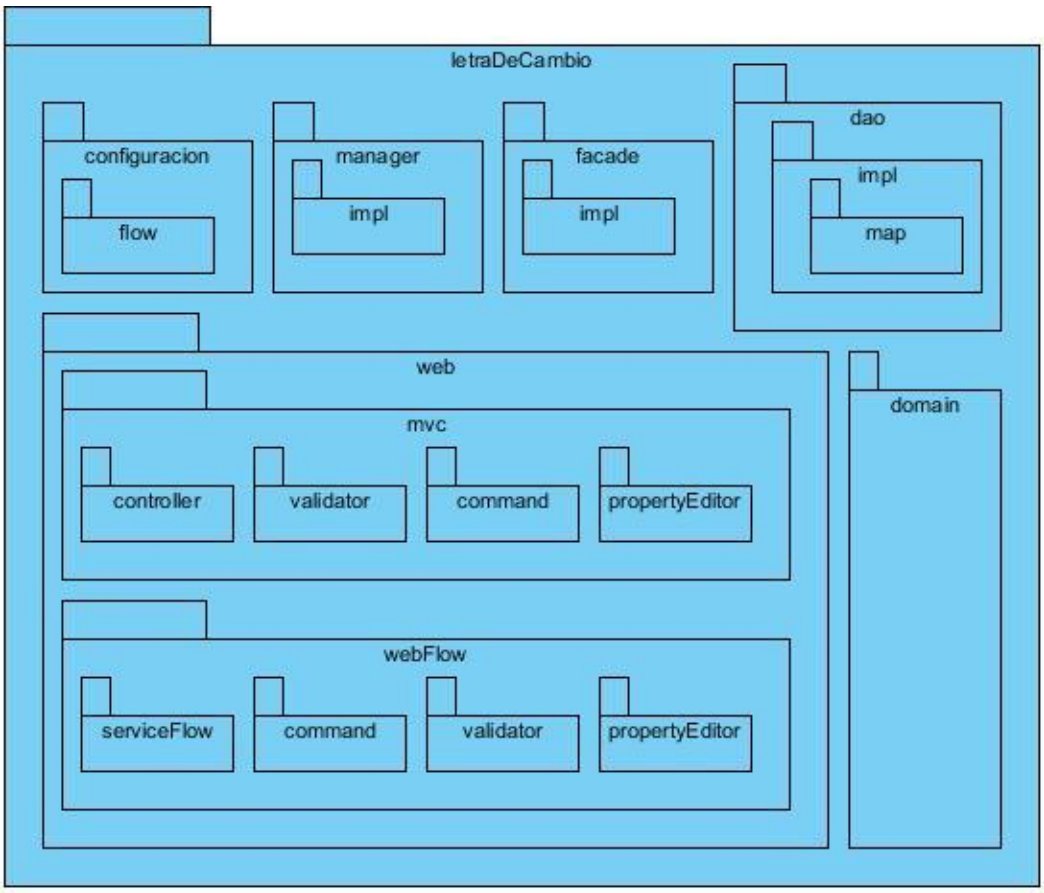


Fig. 9 Diagrama de paquetes del módulo Letra de cambio.

A continuación se describe la estructura de los paquetes para el desarrollo del sistema desde la visión de subsistema hasta la estructura de un paquete en un módulo.

- **configuration**: ficheros .properties para indicar los módulos que se desplegarán.
- **manager**: las interfaces del paquete manager.
  - ✓ **impl**: las implementaciones de las interfaces del paquete manager.
- **facade**: las interfaces de la fachada del negocio.
  - ✓ **impl**: las implementaciones de las interfaces de la fachada.
- **dao**: interfaces Daos.
  - ✓ **impl**: las implementaciones de las interfaces Daos.
    - **map**: los ficheros de mapeo utilizados por Hibernate.
- **web**: las clases pertenecientes a la lógica de presentación. Fundamentalmente las clases controladoras de Spring MVC y las clases que se utilizarán para Spring Web Flow.
  - ✓ **mvc**: agrupa las clases que se utilizarán sobre Spring MVC.
    - **command**: las clases command.

- **controller**: los controladores de Spring MVC.
- **validator**: las clases que validarán los datos que se introduzcan en las páginas.
- **propertyEditor**: las clases propertyEditor.

✓ **webFLow**: las clases para realizar el trabajo de Spring Web Flow.

- **command**: las clases command.
- **flowHandler**: las clases FlowHandler.
- **propertyEditor**: las propertyEditor.
- **serviceFlow**: las clases que sirven como controladores entre la fachada y la capa de presentación.
- **validator**: las clases que validan los datos que se introduzcan en las páginas.

➢ **domain**: las entidades persistentes que pertenecen al módulo.

2.5.2 Diagrama de clases del diseño

Una realización de caso de uso del diseño es una colaboración del modelo de diseño que describe cómo se realiza un caso de uso específico y cómo se ejecuta en término de clases de diseño y sus objetos. Una realización contiene diagramas de clases que muestran sus clases de diseño participantes y diagramas de interacción que muestran la realización de un flujo o escenario en concreto, en términos de interacción entre objetos (Jacobson, y otros, 2000).

Específicamente los diagramas de clases de diseño son muy útiles porque muestran la estructura de las clases que después serán escritas en algún lenguaje de programación. A continuación se muestra una parte del diagrama de clases del diseño del módulo Letra de cambio (Ver Fig. 10), en los anexos se podrá ver el diagrama del módulo Pagaré, además de ver el de Letra de cambio completo (Ver Anexo 8 y Anexo 9). En este diagrama se evidencia la relación de las páginas clientes y los formularios con las clases que atienden sus peticiones.

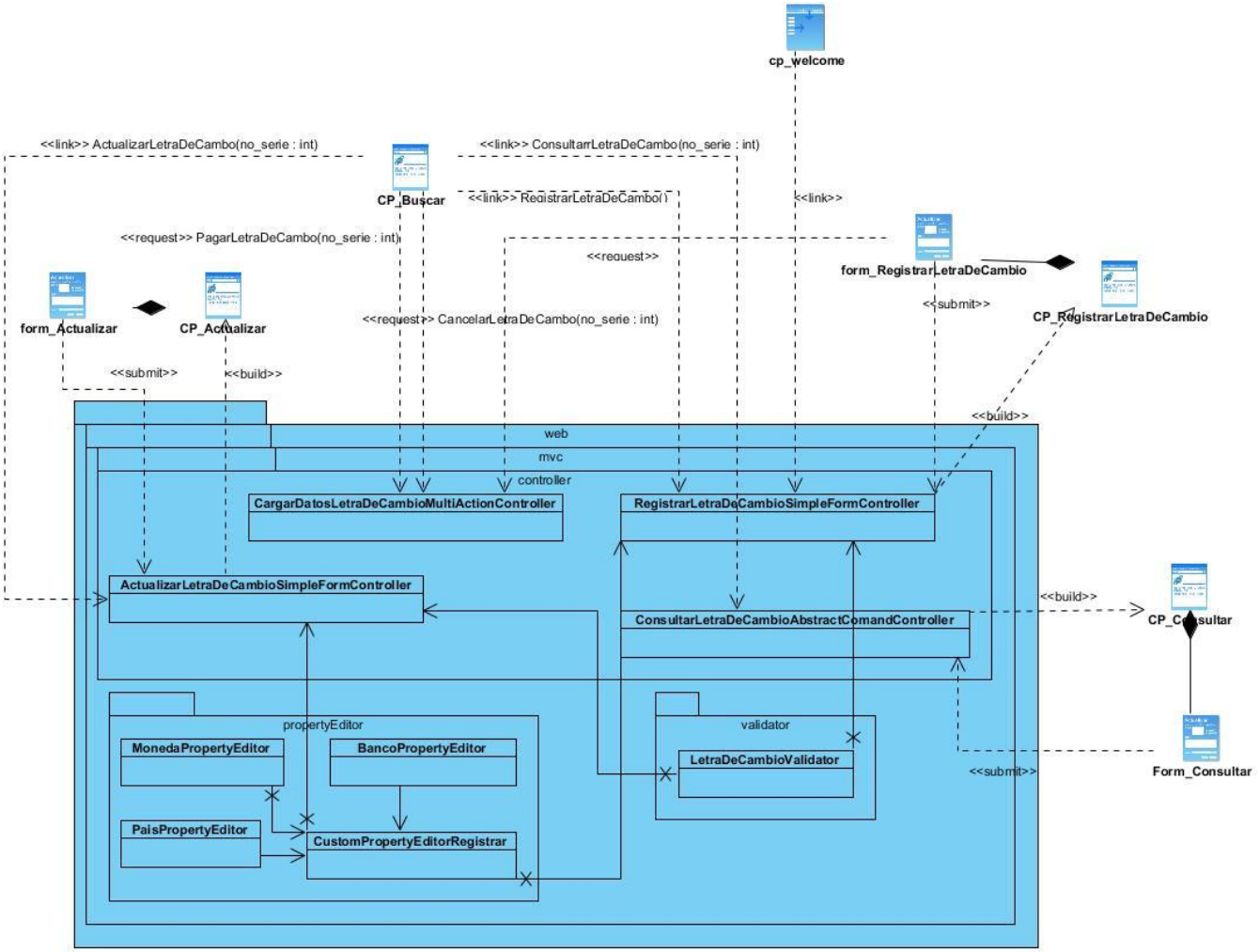


Fig. 10 Diagramas de clases del diseño del módulo Letra de cambio.

2.5.3 Diagrama de secuencia

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 (Pressman, 2005). A continuación se muestra el diagrama de secuencia del requisito Consultar Letra de cambio (Ver Fig. 11), el resto de los diagramas podrán verse en los anexos (Ver Anexo 10 y Anexo 11).

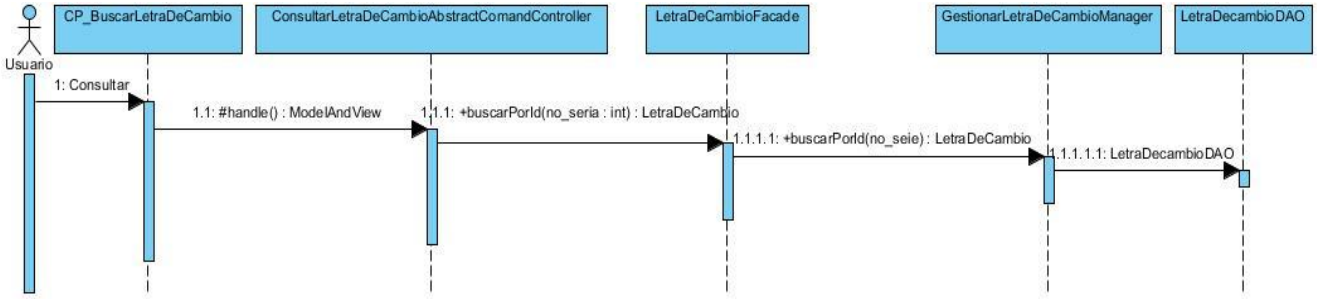


Fig. 11 Diagrama de secuencia Consultar Letra de cambio.

2.5.4 Modelo de datos

El modelo de datos es un modelo abstracto que representa cómo se deben usar y representar los datos. Además, describe la estructura de la base de datos, las relaciones entre los datos y las restricciones que deben cumplirse entre ellos (Elmasri, 2002). A continuación se muestra el modelo de datos (Ver Fig. 12) del módulo Letra de cambio, el diagrama del módulo Pagaré puede verse en los anexos (Ver Anexo 12).

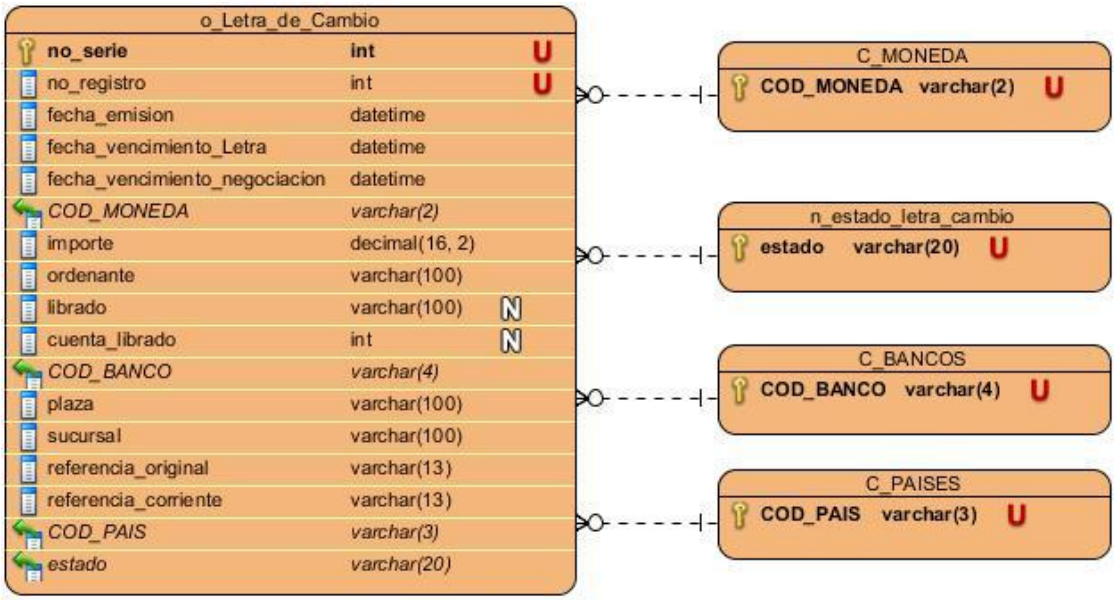


Fig. 12 Modelo de datos de Letra de cambio.

2.6 Patrones de diseño utilizados

Generalmente, para elaborar el diseño se utilizan un grupo de patrones o modelos para lograr los objetivos esperados. Estos son considerados como procedimientos para solucionar diversos problemas del mismo tipo y son la base para la búsqueda de soluciones a dificultades comunes en el

desarrollo de software. Para definir el diseño de la solución propuesta se tuvieron en cuenta varios de patrones, a continuación se especifican los utilizados.

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

Patrones GRASP

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

Ejemplo: RegistrarLetraDeCambioSimpleFormController.

Experto: este patrón fue utilizado en el diseño del subsistema de manera general en la definición de las clases de acuerdo con las funcionalidades que deben realizar a partir de la información que manejan.

Alta cohesión: este patrón fue utilizado en el diseño del subsistema de manera general evidenciándose en la agrupación de las clases en dependencia de los requisitos.

Bajo acoplamiento: este patrón se evidencia con la definición de interfaces e implementaciones, como puede ser la interfaz LetraDeCambioFacade y su implementación LetraDeCambioFacadeImpl permitiendo que clases como RegistrarLetraDeCambioSimpleFormController se relacionen únicamente con ellas para realizar sus operaciones, reduciendo el impacto de cambios posteriores en el negocio del sistema, logrando que el código sea más fácil de entender, mantener y reutilizar.

Patrones GOF

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

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.

2.7 Métricas para la evaluación del diseño

Las métricas para aplicaciones informáticas, no son perfectas; muchos expertos argumentan que se necesita más experimentación hasta que se puedan emplear bien las métricas de diseño. Sin embargo, el diseño sin medición, es una alternativa inaceptable (Gonzáles, 2010). Las métricas son una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado. Permiten averiguar cuán bien están definidas las clases y el sistema, lo cual tiene un impacto directo en el mantenimiento del mismo, tanto por la comprensión de lo desarrollado como por la dificultad de modificarlo con éxito. Estas métricas tienen como propósito entender y mejorar la calidad del producto, evaluar la efectividad del proceso y mejorar la calidad del trabajo realizado al nivel del proyecto (Pressman, 2005). En el presente trabajo se aplicará la Métrica de Tamaño Operacional de Clase (TOC).

2.7.1 Métrica de Tamaño Operacional de Clase (TOC)

La métrica TOC se basa esencialmente en la cantidad de funcionalidades contenidas en las clases. A partir de las ellas se determina la afectación que ejerce en el diseño (Pressman, 2005). En la tabla que se muestra a continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica (Ver Tabla 4).

Tabla 4 Afectaciones en el diseño según la métrica TOC.

Tamaño Operacional de Clase (TOC)	
Atributos	**Afectación**
Responsabilidad	El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
Complejidad de Implementación	El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.

Para la evaluación de cada atributo se establece un rango de valores que determinan la complejidad en la aplicación del TOC. A continuación se muestra una tabla con los rangos de valores y la complejidad por cada uno de los atributos (Ver Tabla 5).

Tabla 5 Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

	Categoría	Criterio
Responsabilidad	Baja	< =Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Complejidad implementación	Baja	< = Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio
Reutilización	Baja	> 2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio

Esta métrica fue empleada para verificar la calidad del diseño obtenido del módulo Letra de cambio. Para comprobar la aplicación de la métrica al módulo Pagaré se puede ver los anexos (Ver Anexo 13). A continuación se muestran los resultados obtenidos una vez aplicada dicha métrica (Ver Tabla 6) (Ver Fig. 13, Fig. 14, Fig. 15).

Tabla 6 Clases más significativas con la cantidad de procedimientos que realizan.

No	Módulo	Clase	Cantidad de Procedimientos
1	LetraDeCambio	CargarDatosLetraDeCambioMultiActionController	6
2	LetraDeCambio	RegistrarLetraDeCambioSimpleFormController	2
3	LetraDeCambio	ActualizarLetraDeCambioSimpleFormController	3
4	LetraDeCambio	CosultarLetraDeCambioAbstractComandController	2
5	LetraDeCambio	LetraDeCambioManagerImpl	11
6	LetraDeCambio	LetraDeCambioDaoImpl	5
	Total		29

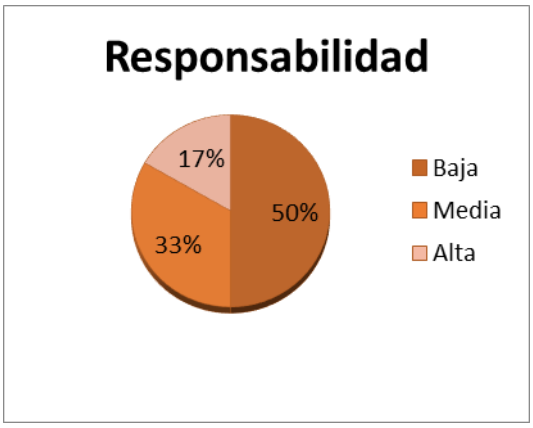


Fig. 13 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

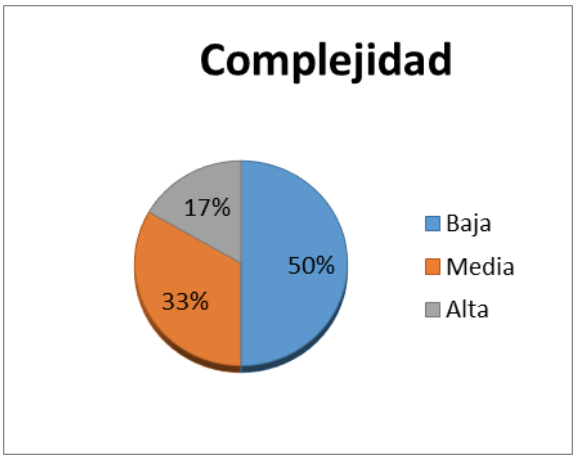


Fig. 14 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

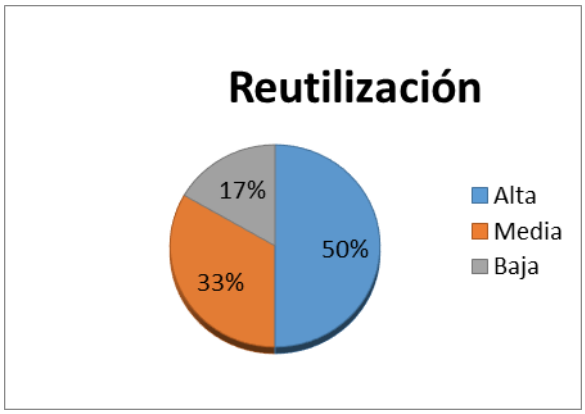


Fig. 15 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel de satisfacción del 50%; de manera que se evidencia cómo se reducen la responsabilidad y complejidad de las clases, además de la elevada reutilización del código (elemento clave en el proceso de desarrollo de software).

2.7.2 Conclusiones Parciales

Al finalizar el capítulo se puede llegar a las siguientes conclusiones:

- Se desarrollaron los flujos de trabajo Análisis y Diseño propuestos en el modelo de desarrollo del centro CEIGE, obteniendo el Modelo de datos, Diagrama de paquetes, Diagrama de clases del diseño y Diagrama de secuencias como principales artefactos del Diseño.

- Con el objetivo de garantizar que los requisitos fueran correctos y cumplieran con las necesidades del cliente, se utilizaron técnicas de validación mediante la construcción de prototipos, revisiones técnicas y confección de casos de pruebas.

- Se describió la arquitectura basada en el patrón arquitectónico MVC y los patrones considerados para el diseño de los casos de uso.

- Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio; de manera que se puede confirmar la elevada reutilización con un 50 % y cómo se reducen la responsabilidad y la complejidad de implementación.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

3.1 Introducción

En el presente capítulo se abordan los temas referentes a la implementación y pruebas del sistema. Se muestran los estándares de codificación utilizados, el diagrama de componentes y de despliegue. Se especifican las pruebas que acreditan el correcto funcionamiento del producto y la validación del sistema. Se valida la solución de la investigación, demostrando que se obtuvieron los resultados esperados, de manera que se da cumplimento al objetivo perseguido con el desarrollo del presente trabajo de diploma.

3.2 Implementación del sistema

La implementación es el principal flujo de trabajo en la fase de construcción. En él se describe cómo los elementos del modelo de diseño se implementan en función de componentes y por ende en piezas más manejables por el lenguaje del programador. Tiene como objetivo llevar a cabo la implementación de cada una de las clases significativas del diseño (Pressman, 2005).

Durante esta fase del desarrollo de software se implementan todas las funcionalidades del sistema, ya sean clases y métodos que son necesarios para lograr que el sistema funcione correctamente. Los módulos Letra de cambio y Pagaré del subsistema Títulos Valores fueron implementados utilizando los estándares de codificación que se utilizan en todo el desarrollo de Quarxo. A continuación se exponen estos estándares, detallándose las convenciones de nomenclatura según el tipo de clase que fue utilizado.

3.2.1 Estándar de Codificación

Los estándares de codificación son guías relacionadas con la generación del código. Los mismos son utilizados para facilitar la legibilidad, comprensión y mantenimiento del código entre distintos programadores. En el desarrollo del sistema Quarxo fueron empleados para lograr una mayor organización de todo el código y del sistema en general. Para ello se utilizó los estándares de codificación definidos en el proyecto banco y del cual se referencia a continuación.

Organización de los ficheros js y css

Todos los ficheros **js** del proyecto estarán en una carpeta con ese mismo nombre dentro de la carpeta **WebContent**, de la misma forma estarán ubicados los **css** y las **imágenes** (Banco, 2011) (Ver Fig. 16).

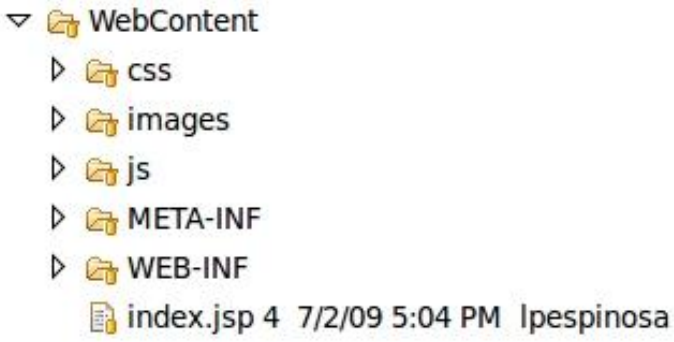


Fig. 16 Organización de los ficheros js y css.

Para una mayor organización, los ficheros ***js*** estarán organizados por subsistemas y módulos, o sea, seguirán el siguiente patrón de carpetas (Banco, 2011) (Ver Fig. 17).

WebContent/js/finixu/subsistema/módulo/*.js.

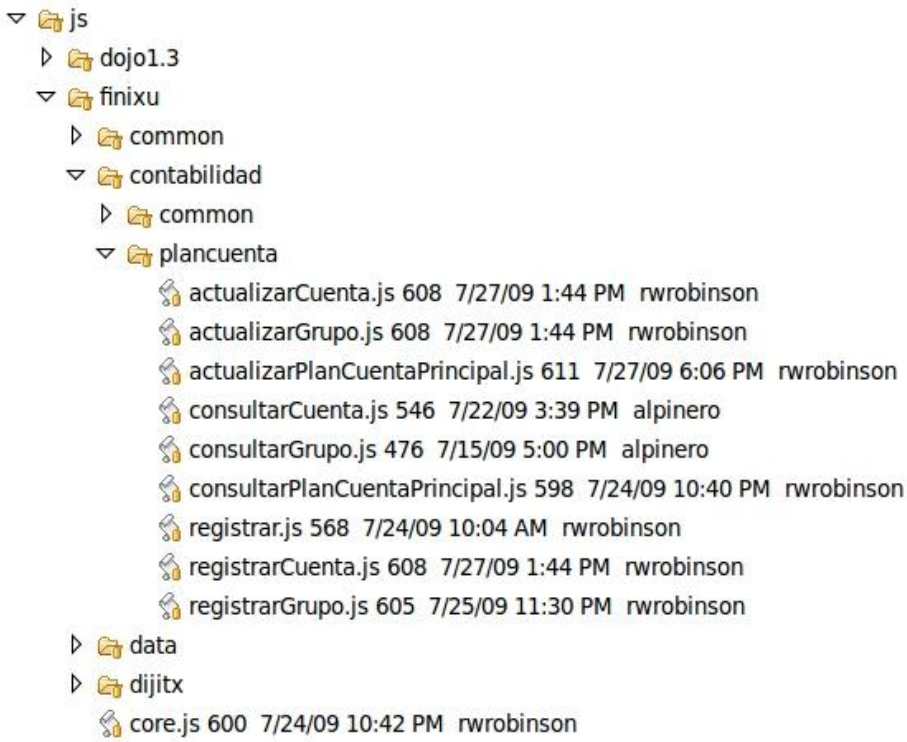


Fig. 17 Organización de los ficheros js.

Nombres de los archivos

Los nombres de archivos deben ser sustantivos y comenzar con letra mayúscula, si está compuesto por más de una palabra se forma el nombre con todas las palabras juntas y la primera letra de cada palabra con letra mayúscula. Solamente se permite el uso de palabras completas, no se pueden usar abreviaturas (Banco, 2011).

Organización del archivo

Cada archivo contiene varias secciones (funciones) y deben separarse con un espacio en blanco entre ellas. Los archivos con más de 1000 líneas de códigos deben ser evitados debido a que dificultarían un posterior mantenimiento (Banco, 2011).

Convenciones de nombres

Todos los nombres que se usen tienen que ser en español, no se admite ningún nombre en otro idioma (Banco, 2011).

Funciones

Las funciones deben ser verbos, comenzar siempre con letra minúscula y en caso de ser varias palabras, se pondrían todas seguidas siendo la primera letra de la primera palabra minúscula y la primera letra de cada una de las restantes palabras sería mayúscula. No se puede usar abreviaturas y el nombre debe describir completamente la acción principal de la función (Banco, 2011).

Variables

Se cumple lo mismo que en las funciones, aunque para el caso que se usen variables temporales se pueden usar abreviaturas (i, j, k) (Banco, 2011).

Constantes

Las constantes se escriben con mayúscula la palabra completa y si son más de una palabra se separan con un guion bajo “_” (Banco, 2011).

3.2.2 Modelo de componentes

Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos del software que entran en la fabricación de aplicaciones informáticas (Jacobson, y otros, 2000). El siguiente diagrama de componentes muestra las relaciones existentes entre los módulos Letra de cambio y Pagaré con el resto de los componentes del sistema Quarxo y una breve descripción de las funciones que realizan cada uno de ellos (Ver Fig. 18).

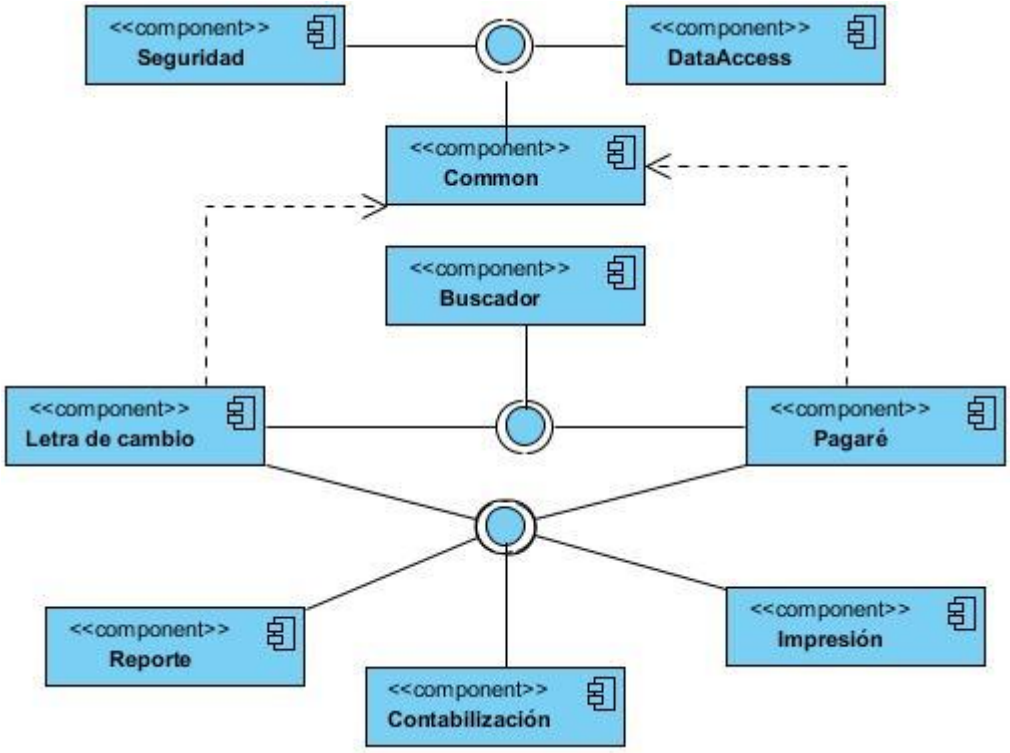


Fig. 18 Diagrama de componentes.

El componente **Common** es el subsistema común para todos los subsistemas. Este tendrá clases que serán utilizados por los demás subsistemas.

El componente **Letra de cambio** es el módulo del subsistema Títulos Valores que contiene todos los datos de las Letras de cambio.

El componente **Pagaré** es el módulo del subsistema Títulos Valores que contiene todos los datos de los Pagarés.

Mediante el componente **Buscador** se ofrecen un grupo funcionalidades y clases que forman en su conjunto el motor de búsqueda del sistema, mediante una interfaz gráfica se solicitan los diferentes conceptos y criterios en dependencia del módulo donde se emplee.

Utilizando el componente **DataAccess** se acceden a las funciones necesarias para llevar a cabo la conexión a la base de datos.

El componente **Impresión** brinda las clases que contienen las funcionalidades mediante las cuales es posible imprimir determinada información.

El componente **Seguridad** es el 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, gestiona la información de los usuarios, roles y permisos necesarios para lograr la confiabilidad requerida en el sistema. Por lo que se hace necesaria la comunicación con este componente para garantizar la seguridad de los módulos.

La comunicación con el componente **Contabilización** es necesaria para obtener la fecha contable del sistema, que se utilizan a la hora de manejar todas las fechas de estos módulos.

El componente **Reporte** contiene todos los reportes que se generan en el sistema.

3.2.3 Diagrama de despliegue

Los diagramas de despliegues muestran los recursos que son necesarios a la hora de poner en marcha el sistema para que todo funcione correctamente. Para desplegar el sistema Quarxo se necesita una PC Cliente que tenga instalada una máquina virtual de Java, que se encuentre conectada a una Impresora mediante USB y al Servidor de aplicación mediante HTTPS. Este servidor de aplicaciones debe tener instalada una máquina virtual de Java, además del Apache Tomcat 6.0, además debe estar conectada mediante el protocolo TCP/IP al servidor de Base de Datos Microsoft SQL Server 2005. La siguiente imagen (Ver Fig. 19) muestra el diagrama de despliegue del sistema de Quarxo desarrollado.

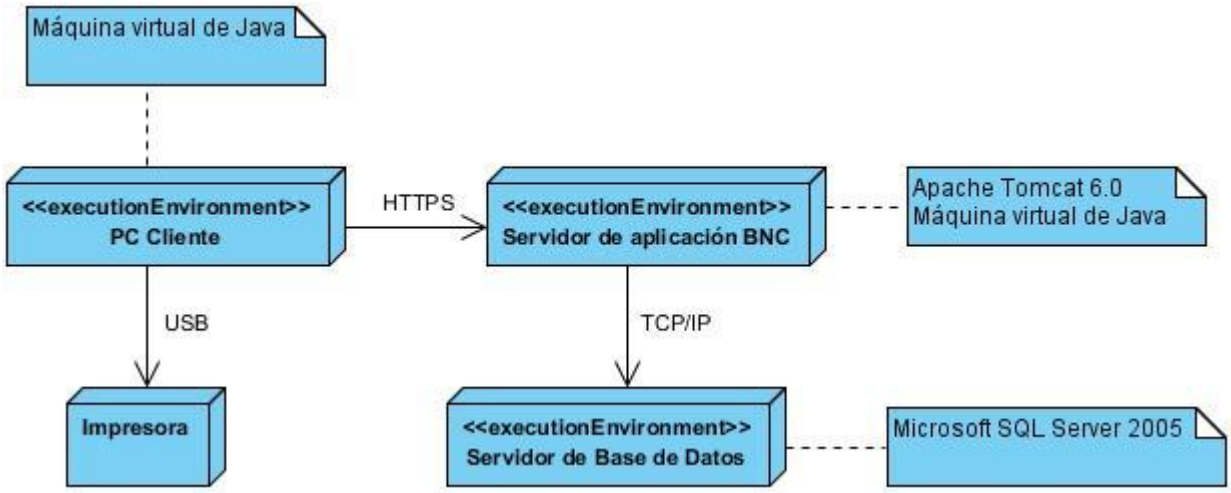


Fig. 19 Diagrama de despliegue.

3.2.4 Descripción de las principales clases a utilizar

A continuación se muestran las clases más importantes desde el punto de vista funcional para el módulo Letra de cambio. Se realiza una descripción de las funcionalidades que se realizan en estas clases (Ver Fig. 20 y Fig. 21) (Ver Tabla 7 y Tabla 8). El resto de las clases aparecen en los anexos (Ver Anexo 14 y Anexo 15).

Fig. 20 Clase CargarDatosLetraDeCambioMultiActionController.

Tabla 7 Descripción de las funcionalidades de la clase CargarDatosLetraDeCambioMultiActionController.

Nombre: CargarDatosLetraDeCambioMultiActionController	
Atributos	**Descripción**
<Property> -attribute:LetraDeCambioFacade	Referencia a la fachada de módulo.

Métodos	Descripción
cargarPais(request :HttpServletRequest, response : HttpServletResponse) : void	Se utiliza para cargar todos los países.
cargarMonedas(request :HttpServletRequest, response : HttpServletResponse) : void	Carga los tipos de monedas (EUR, CUC, USD).
cargarBancos(request :HttpServletRequest, response :HttpServletResponse) : void	Carga todos los bancos (Solo los bancos cubanos).
imprimir(request : HttpServletRequest, response :HttpServletResponse) : void	Imprime el comprobante de registro de la Letra de cambio.
cancelarLetraDeCambio(request : HttpServletRequest, response : HttpServletResponse) : void	Cambia el estado de las Letras de cambio a Cancelada.
pagarLetraDeCambio(request : HttpServletRequest, response : HttpServletResponse) : void	Cambia el estado de las Letras de cambio a Pagada.

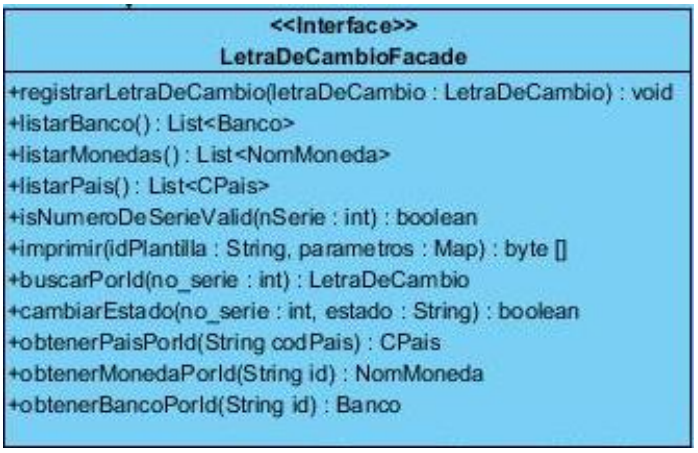


Fig. 21 Clase LetraDeCambioFacadeImpl.

Tabla 8 Descripción de las funcionalidades de la clase LetraDeCambioFacadeImpl.

Nombre: LetraDeCambioFacadeImpl	
Atributos	**Descripción**
<Property> -attribute:LetraDeCambioFacade	Referencia a la fachada de módulo.
Métodos	**Descripción**

registrarLetraDeCambio(letraDeCambio: LetraDeCambio) : void	Registra las Letras de cambio.
listarBanco() : List<Banco>	Devuelve la lista de los bancos.
listarMonedas() : List<NomMoneda>	Devuelve la lista de los nombre de las monedas.
listarPais() : <CPais>	Devuelve la lista de los países.
isNumeroDeSerieValid(nSerie : int) : boolean	Valida el número de serie.
imprimir(idPlantilla:String, parámetros:Map) : byte []	Imprime el comprobante de registro de la Letra de cambio.
buscarPorId(no_serie:int):LetraDeCambio	Busca las Letras de cambio con un número de serie dado.
cambiarEstado(no_serie : int, estado : String) : boolean	Busca la Letras de cambio con un número de serie dado y le cambia el estado.
obtenerMonedaPorId(String id) : NomMoneda	Busca las monedas por su identificador y las devuelve.
obtenerPaisPorId(String codPais) : CLases.CPais	Busca los países por su identificador y los devuelve.
obtenerBancoPorId(String id) : CLases.Banco	Busca los bancos por su identificador y los devuelve.

3.3 Pruebas

El principal objetivo de esta disciplina es evaluar la calidad del producto que se desarrolló. Se realiza a través de diferentes fases mediante la aplicación de pruebas concretas para validar que las suposiciones hechas en el diseño y que los requerimientos se estén cumpliendo satisfactoriamente (Jacobson, y otros, 2000). Se verifica el funcionamiento del producto como se diseñó y que los requerimientos han sido cumplidos satisfactoriamente. Las pruebas son unas de las maneras de validar que lo que se desarrolló cumple con todas las expectativas del cliente. Se verifica si lo que se ha implementado está acorde con el análisis y el diseño obtenido anteriormente. Permite la obtención de errores que pueden ser corregidos antes de que sea desplegado el sistema en la empresa cliente. Además de que es la mejor manera de comprobar que todo funciona correctamente.

3.3.1 Pruebas unitarias

El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores. Las pruebas unitarias son una forma de probar el correcto funcionamiento del código, esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado (Myers, 1983). Las pruebas unitarias realizadas a los servicios del sistema sirven para validar que las salidas son correctas y aseguran al desarrollador que la solución no presenta errores en la lógica de programación y que las respuestas son las correctas ante una entrada de datos determinada. Se aplicarán los métodos de pruebas adaptados a este nivel, como son los métodos de prueba: Caja Blanca, para comprobar los caminos lógicos del software y Caja Negra, para probar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

3.3.2 Pruebas de Caja Blanca

Las pruebas de caja blanca permiten aumentar la calidad y confiabilidad del software. Estas son aplicadas al código logrando como resultado que disminuya el número de errores existentes en el sistema. Se identificaron dos formas de realizar las pruebas de caja blanca (Tahchiev, 2010):

- Pruebas estáticas de caja blanca: es el proceso que cuidadosamente y metódicamente revisa el diseño del software, la arquitectura o el código para encontrar defectos sin necesidad de ejecutar el código. Esto algunas veces se refiere a un análisis estructural.
- Pruebas dinámicas de caja blanca: en estas pruebas se revisa dentro de la “caja”, se examina el código y se observa este mientras se ejecuta. Utiliza la información que se obtiene al observar que hace el código, cómo trabaja, para así determinar que probar, que no probar y cómo aproximarse a las pruebas.

Para probar los módulos implementados se seleccionan las pruebas de caja blanca dinámica. Para esto se utilizó el framework JUnit, el cual brinda un conjunto de librerías que se integran fácilmente al IDE de desarrollo seleccionado: Eclipse. JUnit permite la realización de las pruebas a los métodos de las clases implementadas. Para hacer uso de este framework se definieron los siguientes pasos:

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

- Definir los métodos a probar dentro de cada caso de prueba, incluyendo los parámetros de entrada.
- Realizar pruebas a los métodos (Tahchiev, 2010).

A continuación se evidencia cómo se le realizaron las pruebas de caja blanca a la clase GestionarLetraDeCambioManagerImpl localizada en el módulo LetraDeCambio, demostrándose mediante la aplicación de la prueba al método: CambiarEstado (Ver Fig. 22).

```
public boolean cambiarEstado(int nSerie, String estado) {
    // TODO Auto-generated method stub
    System.out.println("GestionarLetraDeCambioManagerImpl.cambiarEstado()");
    LetraDeCambio letraDeCambio = letraDeCambioDAO.buscarPorId(nSerie);
    if (letraDeCambio == null)
        return false;
    EstadoLetraCambio estadoLetraCambio = letraDeCambio
            .getEstadoLetraCambio();
    if (estadoLetraCambio.getEstado().equals("CANCELADA"))
        return false;
    else if (estadoLetraCambio.getEstado().equals("PAGADA")
            && !estado.equals("CANCELADA"))
        return false;
    else if (estadoLetraCambio.getEstado().equals("EMITIDA")
            && estado.equals("PAGADA")) {
        EstadoSistema estadoSistema = globalFacade.obtenerEstadoSistema();
        Date fechaSistema = estadoSistema.getId().getFecContab();
        if (fechaSistema.after(letraDeCambio.getFechaVencimientoLetra())
                && fechaSistema.before(letraDeCambio
                        .getFechaVencimientoNegociacion())
                || esMismoDia(fechaSistema, letraDeCambio
                        .getFechaVencimientoLetra())
                || esMismoDia(fechaSistema, letraDeCambio
                        .getFechaVencimientoNegociacion())) {
            return letraDeCambioDAO.cambiarEstado(nSerie, estado);
        } else
            return false;
    }

    return letraDeCambioDAO.cambiarEstado(nSerie, estado);
}
```

Fig. 22 Método CambiarEstado.

Para realizar la prueba a dicho método primeramente se crea una clase que herede de la clase TestCase de JUnit, dicha clase se nombra GestionarLetraDeCambioManagerImplTestCase, en la cual inicialmente, se implementa el método setUp() en el cual se define la configuración que va a tener el entorno de prueba, para este caso mediante el archivo titulosvalores-contex.xml se

accederá al contexto de la aplicación para poder hacer la llamada a los métodos de la clase que se quiere probar. Este contexto permitirá mediante la prueba utilizar las referencias a las clases que se encuentran en otras capas de la aplicación (Ver Fig. 23).

```
private GestionarLetraDeCambioManagerImpl manager;
        private int idBuscar, idCambiarEstado, idSerieValido;
        private String estadoLetraCambio;

        public void setUp() throws Exception {
                ApplicationContext context = new ClassPathXmlApplicationContext(
                                "classpath:junit/titulosvalores-context.xml");
                manager = (GestionarLetraDeCambioManagerImpl) context
                                .getBean("gestionarLetraDeCambioManager");
                idBuscar = 698744522;
                idCambiarEstado = 121233122;
                idSerieValido = 365874125;
                estadoLetraCambio = "CANCELADA";
        }
```

Fig. 23 Clase GestionarLetraDeCambioManagerImplTestCase.

Durante la realización de la prueba, se preparan los parámetros de pruebas que le son necesarios al método CambiarEstado. A estos parámetros se les verifica la condición de prueba a través del método assertEquals() el cual comprueba si el resultado de la llamada al método devuelve un valor distinto del que se especifica (Ver Fig. 24).

```
public final void testCambiarEstado() {
                assertEquals(false, manager.cambiarEstado
(idCambiarEstado, estadoLetraCambio));
        }
```

Fig. 24 Prueba realizada con el assertEquals().

Resultado de las pruebas de caja blanca

El resultado de las pruebas realizadas a los métodos de la clase GestionarLetraDeCambioManagerImpl fue satisfactorio (Ver Fig. 25). Evidenciándose por el color verde mostrado en la barra de la parte superior izquierda (en caso de existir fallos la barra tendría color rojo). En dicho escenario se ejecutaron 11 casos de pruebas, donde el 100 % tuvo 0 errores (errors en inglés) y 0 fallos (failures en inglés). En los anexos se puede evidenciar la realización de las pruebas a otros métodos y los resultados arrojados (Ver Anexo 17).

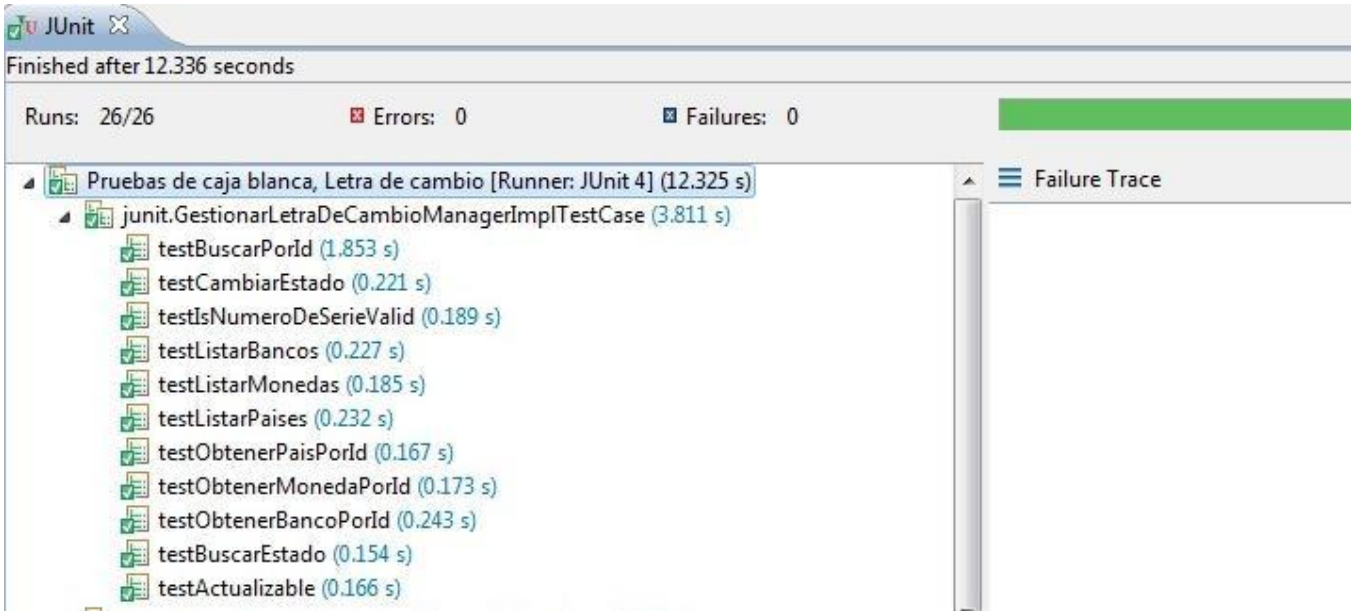


Fig. 25 Resultado de las pruebas de caja blanca.

3.3.3 Pruebas de Caja Negra

Las pruebas de Caja Negra o pruebas Funcionales se realizan sobre la interfaz del software, entendiendo por interfaz las entradas y salidas del mismo. No es necesario conocer su lógica de funcionamiento, únicamente la funcionalidad que debe realizar. También conocidas cómo Pruebas de Comportamiento, estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una "Caja Negra" cuyo comportamiento solo puede ser determinado mediante el estudio de las entradas y salidas obtenidas a partir de ellas. Las técnicas de prueba no se hallan de forma aislada, sino como un conjunto integrado de acciones que combinadas permiten verificar y evaluar la calidad de software. Entre las técnicas para desarrollar la prueba de Caja Negra se encuentran (Pressman, 2005):

- Técnica de la partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del análisis de valores límites: prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de grafos de causa-efecto: permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

Para la realización de las pruebas de caja negra que se le aplicaron a los módulos implementados se utilizó la técnica: partición de equivalencia. Esta técnica es efectiva al permitir comprobar los valores válidos e inválidos de todas las entradas existentes en el software. Además, permite la reducción del número total de casos de prueba que hay que desarrollar. Para verificar que la aplicación se comporta según los requerimientos establecidos por el cliente, se diseñan casos de pruebas usando el método de caja negra (Ver Anexo 18).

Resultado de las pruebas de caja negra

Los resultados obtenidos a través de la realización de las pruebas realizadas, fueron satisfactorios desde el punto de vista interno y funcional, dado que los módulos mantuvieron un correcto comportamiento ante las diferentes situaciones en que se probaron. Estos fueron probados mediante pruebas internas por el equipo de calidad del centro CEIGE, donde fueron sometidos a tres iteraciones de pruebas funcionales, exploratorias y de regresión. En las dos primeras iteraciones se obtuvo un total de 9 no conformidades (Ver Anexo 16).

La primera iteración de prueba fue realizada por uno de los miembros del equipo de calidad: Leandro Reyes, el día 01 de mayo de 2013. En esta iteración se obtuvieron un total de 8 no conformidades, las mismas se muestran en la tabla que se muestra a continuación (Ver Tabla 9). Estas no conformidades fueron corregidas y los módulos fueron sometidos a una segunda iteración de pruebas.

Tabla 9. No conformidades obtenidas en la primera iteración de pruebas por el equipo de calidad interna de CEIGE.

Elemento	No Conformidad	Aspecto correspondiente	Tipo
Documentación / Aplicación	En las condiciones de ejecución del documento 0000053487-CIG-QF2-N-TV-i5101, el paso No 2 hace referencia a: letra de cambio y en la aplicación aparece cono Gestión de Cambio.	Letra de cambio/0000053487-CIG-QF2-N-TV-i5101.Títulos valores /Gestionar letra de cambio	No correspondencia.
Aplicación / Documentación	En el requisito emitir letra de cambio, EP 1.1: Emitir una Letra de cambio correctamente. El campo Referencia original, no acepta caracteres	Títulos valores /Gestionar letra de cambio. Letra de cambio/0000053487-CIG-QF2-	Validación.

	declarados como válidos.	N-TV-i5101	
Aplicación / Documentación	En el requisito emitir letra de cambio EP 1.1: Emitir una Letra de cambio correctamente. El campo Referencia corriente, no acepta caracteres declarados como válidos.	Títulos valores /Gestionar letra de cambio. Letra de cambio/0000053487-CIG-QF2-N-TV-i5101	Validación.
Aplicación / Documentación	En el requisito emitir letra de cambio EP 1.1: Emitir una Letra de cambio correctamente. El campo Importe, no acepta caracteres declarados como válidos.	Títulos valores /Gestionar letra de cambio. Letra de cambio/0000053487-CIG-QF2-N-TV-i5101.	Validación.
Aplicación	En el requisito emitir letra de cambio el campo Librado, acepta caracteres declarados como inválidos.	Letra de cambio/0000053487-CIG-QF2-N-TV-i5101.Títulos valores /Gestionar letra de cambio/Emitir letra de cambio.	No correspondencia.
Aplicación	En el requisito emitir letra de cambio EP 1.3: Emitir una Letra de cambio dejando campos requeridos en blanco. Al aceptar dejando todos los campos vacíos se emite un mensaje que no corresponde con el DCP.	Letra de cambio/0000053487-CIG-QF2-N-TV-i5101.Títulos valores /Gestionar letra de cambio/Emitir letra de cambio.	No correspondencia.
Aplicación	Falta de ortografía en el mensaje de confirmación palabra (está).	Títulos valores/Gestionar letra de cambio/Emitir letra de cambio/Buscar/Buscar/Cancelar.	Ortografía.
Aplicación	En el requisito 0000053487-CIG-QF2-N-TV-i5106 en Pagar Letra de cambio al dar clic en pagar realiza la funcionalidad de cancelar.	Títulos valores/Gestionar letra de cambio/Emitir letra de cambio/Buscar/Buscar/Pagar.	Funcionalidad.

En la segunda iteración se realizó pruebas de regresión las cuales fueron realizadas por: Leandro Reyes (probador del equipo de calidad) de donde se obtuvo una no conformidad (Ver Tabla 10). Esta no conformidad fue corregida. Los módulos fueron sometidos a una tercera iteración de pruebas donde no se obtuvo ninguna no conformidad. Por los resultados satisfactorios obtenidos en la última iteración el equipo de calidad emitió un acta de liberación de software certificando la calidad del producto desarrollado (Ver Anexo 19).

Tabla 10. No conformidades obtenidas en la segunda iteración de pruebas por el equipo de calidad interna de CEIGE.

Elemento	No Conformidad	Aspecto correspondiente	Tipo
Aplicación	En el requisito emitir letra de cambio el campo Librado, no acepta caracteres declarados como válidos.	Letra de cambio/0000053487-CIG-QF2-N-TV-i5101.Títulos valores /Gestionar letra de cambio/Emitir letra de cambio	No correspondencia.

Los módulos fueron probados también por el equipo de calidad de software (Calisoft) de la UCI, donde se le realizaron pruebas de carga y estrés. En estas pruebas fueron encontradas 4 no conformidades en la aplicación, las mismas fueron corregidas obteniéndose al finalizar un total de 0 no conformidades. Por lo que se obtuvo el acta de liberación por parte de este equipo de Calisoft (Ver Anexo 20).

3.4 Validación de la solución

La investigación desarrollada plantea como problema a resolver: la gestión actual de los procesos asociados a los Títulos Valores se ve afectada por las deficiencias de su ejecución en el BNC. Estas deficiencias son provocadas por la realización manual de los medios de pago Letra de cambio y Pagaré. Lo que trae como consecuencia: el aumento del tiempo empleado para la gestión de estos documentos, además del gasto de recursos materiales.

Con la implementación y puesta en ejecución de los módulos Letra de cambio y Pagaré se contribuye a la mejora de la gestión de estos procesos en el BNC, logrando dar solución al problema planteado al inicio de la investigación. De esta manera, se benefició a los

negociadores del departamento obteniendo una mayor eficiencia y rapidez en la labor que desempeñan. Esto se puede evidenciar mediante la disminución considerable del tiempo de realización y búsqueda de estos medios de pagos (Ver Fig. 28 y Fig. 29). Otro aspecto importante es la mejora en cuanto al consumo de recursos materiales de oficina (Ver Fig. 30), ya que al contar con este sistema informático se reducen los errores humanos, obteniendo un rango mínimo de errores (Ver Fig. 26 y Fig. 27) lo que da paso a emplear menos recursos. Por todo lo antes expuesto se puede concluir que los módulos adicionados al subsistema Títulos Valores satisfacen las necesidades del cliente. En las imágenes que aparecen a continuación, se evidencia como el margen de errores que se cometen al trabajar con los Títulos Valores se reducen luego de obtenido los módulos Letra de cambio y Pagaré.

Fig. 26 Comparación del margen de errores que puede cometerse al gestionar las Letras de cambio.

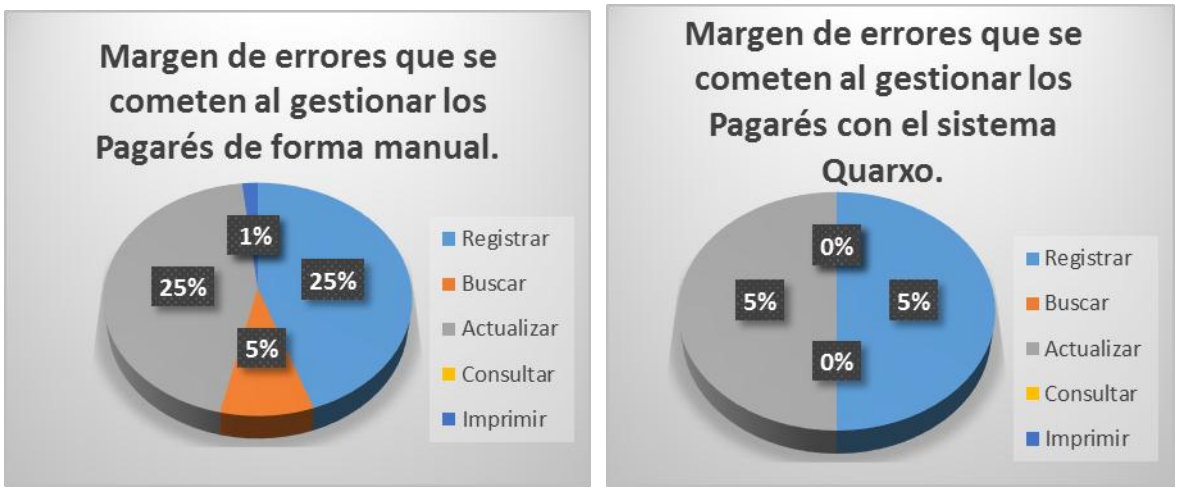


Fig. 27 Comparación del margen de errores que puede cometerse al gestionar los Pagarés.

3.4.1 Disminución de los tiempos de respuesta

El tiempo de respuesta se utiliza haciendo referencia a la demora de las operaciones que se pueden realizar con las Letras de cambio o Pagarés. La disminución de tiempo implica necesariamente mayor eficiencia y productividad, a partir de que se dedica menos tiempo en solucionar estas operaciones y se agilizan los procesos que esperan por su realización. Con la incorporación de los módulos Letra de cambio y Pagaré al subsistema Títulos Valores se reduce considerablemente el tiempo para realizar las operaciones (Ver Fig. 28 y Fig. 29), además de incrementarse las operaciones que se pueden realizar sobre estos medios de pago. A continuación se muestra una comparación que demuestra que el resultado de la solución obtenida satisface las necesidades del cliente.

Fig. 28 Comparación del tiempo de respuesta de la gestión de Letras de cambio.

Fig. 29 Comparación del tiempo de respuesta de la gestión de Pagarés.

3.4.2 Ahorro de recursos materiales

Contar con los recursos materiales adecuados y realizar un uso racional de los mismos es un elemento clave en la gestión de las organizaciones. Debido a los inconvenientes que presenta la elaboración de los medios de pago de forma manual y que estos pueden contener errores humanos al ser elaborados provoca que el gasto de recursos en cuestiones de papel, lapiceros y tinta de impresora sea mayor. El desarrollo de los módulos Letra de cambio y Pagaré de Quarxo fase 2 posibilita una mejor gestión de estos medios de pago, permitiendo crearlos y consultarlos tanto como sea necesario desde el sistema y si contienen errores pueden ser corregidos de inmediato, de esta manera, se imprimirán los documentos que estén correctamente sin necesidad de gastos de papel por causa de errores. A continuación se muestra la reducción del uso de materiales de oficina en el departamento de negociaciones (Ver Fig. 30).

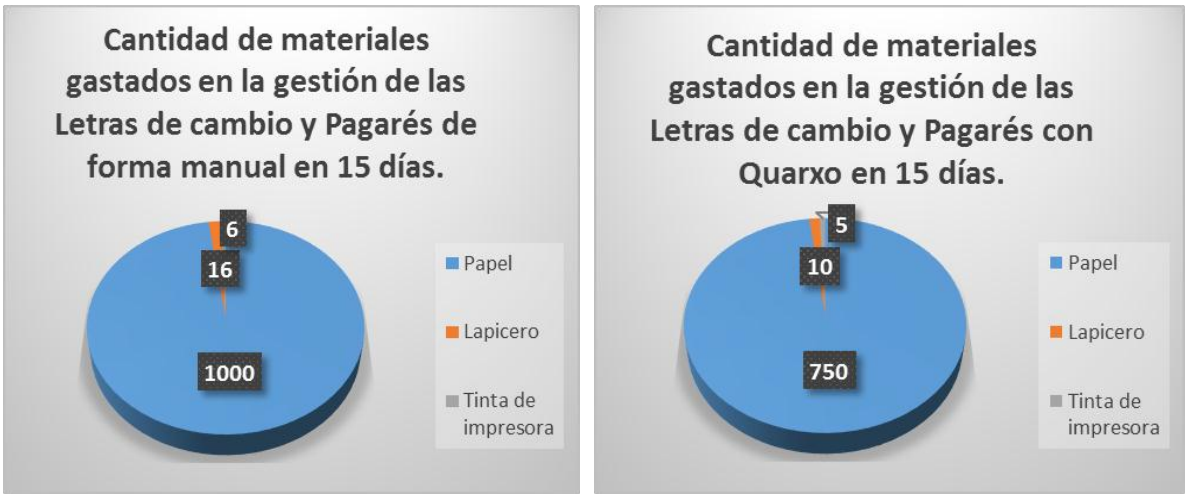


Fig. 30 Comparación de gasto de recursos antes y después de Quarxo.

Por lo expuesto anteriormente, quedan validadas las variables consideradas en la investigación y queda demostrado que la solución desarrollada elimina las deficiencias que existían en el BNC. De esta manera, se reduce el tiempo empleado y el consumo de recursos materiales cuando se requieren gestionar los medios de pagos Letra de cambio y Pagaré. Además, se logra minimizar el rango de errores existente, lo que propicia que el trabajo se desempeñe con mayor eficiencia y calidad.

3.5 Conclusiones Parciales

Al finalizar el capítulo se puede llegar a las siguientes conclusiones:

- Se obtuvieron los módulos de Letra de cambio y Pagaré del subsistema Títulos Valores de Quarxo fase 2, desarrollado con tecnologías libres.
- Los módulos fueron probados mediante pruebas de Caja Blanca y Caja Negra, quedando validados mediante pruebas de aceptación.
- Se probó que la solución obtenida contribuye a la disminución del margen de errores que se cometen en la gestión de las Letras de cambio y Pagarés.
- Quedó demostrado que se reduce el tiempo de respuesta de las operaciones y disminuye el gasto de recursos materiales de oficina.

CONCLUSIONES

Una vez culminado el trabajo se llegó a las siguientes conclusiones:

- Se dio solución al problema planteado al inicio de la investigación, eliminado de esta manera las deficiencias que presentaba la gestión de los procesos asociados a los Títulos Valores.
- Se alcanzó el objetivo general propuesto, al obtenerse los módulos Letra de cambio y Pagaré del subsistema Títulos Valores, contribuyendo a la mejora en el desempeño de estos procesos en el departamento de Negociaciones del BNC.
- El subsistema Títulos Valores permitirá una mejor gestión de la información que maneja al contar con dos módulos nuevos.
- Con el resultado obtenido se cuenta con un registro de todas las Letras de cambio y Pagarés que se elaboran en el departamento de negociaciones, permitiendo llevar un mejor control de estos medios de pago.
- Con el desarrollo de los módulos Letra de cambio y Pagaré se aporta eficiencia en la gestión de estos Títulos Valores, mejorando considerablemente el trabajo de los operadores del banco y minimizando el gasto de tiempo y recursos materiales.

RECOMENDACIONES

Los objetivos propuestos en este trabajo fueron alcanzados satisfactoriamente; sin embargo durante el desarrollo de la investigación surgieron nuevas ideas que serían recomendables tener en cuenta:

- Continuar el desarrollo del subsistema Títulos Valores, implementando un nuevo módulo que permita gestionar las Cartas Remesas, que es otro de los medios de pago que se utiliza actualmente en el BNC.

- Reutilizar los requerimientos funcionales definidos para la gestión de Títulos Valores en el sistema bancario nacional en otras entidades bancarias, aplicando la experiencia adquirida en la obtención de otros sistemas más eficientes y que satisfagan en mayor medida las necesidades de cada cliente.

- Investigar y aplicar la utilización de más Títulos Valores, siendo esta diversificación la que propicie un avance en el comercio tanto nacional como internacional, generando más ingresos y menos movimientos de efectivos entre entidades financieras.

REFERENCIAS BIBLIOGRÁFICAS

Admin. 2013. Grupo Trevenque. *Grupo Trevenque.* [Online] 2013. [Cited: Enero 17, 2013.] http://www.trevenque.es/index.php/Easy-Pagares/85/0/..

Administrador. 2013. Grupo Trevenque. *Grupo Trevenque.* [Online] 2013. [Cited: Enero 17, 2013.] http://www.trevenque.es/index.php/Easy-Letras/84/0/..

Apache. En Línea. Apache tomcat. *Apache tomcat.* [Online] The apache software foudation, En Línea. [Cited: Febrero 19, 2013.] http://tomcat.apache.org/..

Banco. 2011. *Estándares de codificación Proyecto Sistema automatizado de la gestión bancaria.* La Habana : s.n., 2011.

Barboza Parra, Ely Saúl. 1991. *Títulos valores: Principios básicos.* Merida, Venezuela: Universidad de los Andes : s.n., 1991. ISBN 980-221-381-0.

BCC. 2013. Banco Central de Cuba. *Banco Central de Cuba.* [Online] 2013. [Cited: Enero 12, 2013.] webmaster@bc.gov.cu. http://www.bc.gov.cu/Espanol/default.asp.

Beaumont, Ricardo Arturo Callirgos. 2010. *Regulación de la Letra de cambio en la nueva Ley de Títulos Valores: Innovaciones destacables.* 2010. ISBN-34694134.

Bergin, Thomas J. 1989. *The CASE experience.* EEUU : s.n., 1989. pp. págs. 235-244.

Casals, Velmour Muñoz. 2010. *Metodología para el proceso de diagnóstico de la informatización de las instituciones de salud a sus diferentes niveles de atención.* Centro de Informática Médica (CESIM), Universidad de las Ciencias Informáticas (UCI). La Habana : s.n., 2010. pp. 4-5, Proyecto de Investigación.

Christian, Gavin King Bauer. 2005. *Hibernate in Action.* EEUU : Printed in the United States of America, 2005. ISBN 1932394-15-X.

Clara, Ángel Cala Rivero and Rodríguez Alcalde, José Ángel Barroso. 2003. *Reflexiones sobre el Framework de desarrollo del Consejo Superior de Investigaciones Científicas.* 2003.

Clements, Paul C. 2002. *Software Architecture.* Estados Unidos : Software Engineering Institute, 2002. ISBN: 15213-3890.

Dhingar, P. and Swanson, T. 2007. Microsoft SQL Server 2005. 2007.

Dojo. En línea. DOJO TOOLKIT. *DOJO TOOLKIT.* [Online] En línea. [Cited: Febrero 20, 2013.] http://dojotoolkit.org..

Eclipse. En línea. Eclipse. *Eclipse.* [Online] En línea. [Cited: Febrero 16, 2013.] http://help.eclipse.org/galileo/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/int_eclipse.htm.

EcuRed. 2013. EcuRed. *EcuRed.* [Online] 2013. [Cited: Enero 20, 2013.] http://www.ecured.cu/index.php/Banco_Nacional_de_Cuba..

Elmasri, R. y Navathe, S. B. 2002. *Fundamentos de sistemas de bases de datos.* s.l. : Addison-Wesley, 2002. p. 8478290516.

Escalona, María José y Koch, Nora. 2002. *Ingeniería de Requisitos en Aplicaciones para la Web –Un estudio comparativo.* Departamento de Lenguajes y Sistemas Informáticos , Escuela Técnica Superior de Ingeniería Informática, Universidad de Sevilla. Sevilla : s.n., 2002. p. 26.

Francisco, Javier. 2011. abanfin. *abanfin.* [Online] Marzo 28, 2011. [Cited: Enero 17, 2013.] http://www.abanfin.com/?tit=letra-de-cambio&name=Manuales&fid=eh00003.

Gadea, Enrique. 2007. *Los títulos-valor: Lletra de cambio, cheque y pagaré.* s.l. : Librería-Editorial Dykinson, 2007. p. 129. ISBN 8498490022.

Gamma, E. and Helm, R. 1995. *Design Patterns: Elements of Reusable Object-Oriented Software.* s.l. : Addison-Wesley Pub Co, 1995.

Gonzáles, Doria H. 2010. *Las Metricas de Software y su Uso en la Región.* Escuela de Ingeniería. Ecuador, Universidad de las Américas Puebla : Departamento de Ingeniería en Sistemas Computacionales, 2010.

Grau, Xavier Ferré and Segura, María Isabel Sánchez. 2008. *Desarrollo Orientado a Objetos con UML.* s.l. : Facultad de Informática – UPM, 2008.

Iglesias, Adolfo Miguel. 2008. *Propuesta arquitectónica inicial para la Modernización del Sistema Bancario.* La Habana : s.n., 2008.

—. 2009. *QUARXO, PROYECTO. Modernización Sistema del Banco Nacional de Cuba.* La Habana : s.n., 2009.

Jacobson, I. and Booch, G. 2000. *El proceso unificado de desarrollo de software.* Madrid, España : Pearson, 2000.

Java. En Línea. Ciberaula. *Ciberaula.* [Online] En Línea. [Cited: Febrero 16, 2013.] http://java.ciberaula.com/articulo/que_es_java.

Larman, Craig. 1999. *UML y Patrones.* s.l. : Prentice Hall, 1999. ISBN: 970-17-0261-1.

Meneses, Yadira Calimano and Chaviano, Adolfo Miguel Iglesias. 2010. *Sistema Integrador para la Comunicación Financiera para el Banco Nacional de Cuba.* CUJAE. La Habana : s.n., 2010.

Moreno, Henry Rodríguez. 2005. *Apuntes Básicos en Materias de Títulos Valores.* Costa Rica : s.n., 2005. p. 44, Notas relacionadas con el medelo legal costarricense.

Moreno, Juan J. Dans. 2011. *Modelación e implementación del módulo Documentos del subsistema Depósitos de Aduana.* UCI. La Habana : s.n., 2011. p. 88, Trabajo de diploma.

Muñoz, Javier Martínez. 2008. *Definición De Los Requerimientos Funcionales Del Módulo Cuentas de Clientes y Órdenes de Pago Inmediato Del Proyecto Banco Nacional.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2008. p. 119, Trabajo de diploma.

Myers, G. y Filevich, C. y Filecich, A. 1983. *El arte de probar el software.* s.l. : El Ateneo, 1983.

Obregón, William González. 2012. *CEIGE-Modelo de Desarrollo de Software v1.1.* Centro de Informatización de la Gestión de Entidades, Universidad de las Ciencias Informáticas. La Habana : Subdirección de Producción, 2012. pp. 6-12, Modelo de desarrollo de software.

Pantoja, Ernesto Bascón. 2004. *El patrón de diseño Modelo-Vista-Controlador (MVC) y suimplementación en Java Swing.* 2004. Vol. 2, N◦4.

Paradigm, Visual. En Línea. Visual Paradigm. *Visual Paradigm.* [Online] En Línea. [Cited: Enero 16, 2013.] http://www.visual-paradigm.com/.

Perdomo Bello, Yesenia y Pérez Espinosa, Leosvel. 2009. *Análisis y Diseño del Subsistema Títulos valores del Proyecto Modernización del Sistema Bancario Cubano.* La Habana : s.n., 2009. Trabajo de diploma.

Pressman, R. 2005. *Ingeniería del Software: Un Enfoque Práctico.* 2005.

Rincones, José Francisco Martínez. 2003. *La protección penal de los Títulos Valores en el Sistema Jurídico Venezolano.* Venezuela : s.n., 2003. p. 14.

Rodríguez, Andrés Rolando. En Línea. GestioPolis. *GestioPolis.* [Online] Lenguajes, notaciones y herramientas para el modelado y análisis de procesos, En Línea. [Cited: Febrero 16, 2013.] http://www.gestiopolis.com/administracion-estrategia/lenguajes-notaciones-y-herramientas-en-analisis-de-procesos.htm.

—. 2008. GestioPolis. *GestioPolis.* [Online] Junio 16, 2008. [Cited: Enero 18, 2013.] http://www.gestiopolis.com/administracion-estrategia/lenguajes-notaciones-y-herramientas-en-analisis-de-procesos.htm.

Salud. 2003. *Programa de informatización del sector de la Salud.* Dirección de informática, Ministerio de Salud Pública. La Habana : Dirección de informática., 2003. pp. 3-6.

Shalloway, A. y J. Trott. 2004. *Design Patterns Explained: A New Perspective on Object-Oriented Design (Software Patterns Series).* s.l. : Addison-Wesley Professional, 2004.

Sommerville, Ian. 2005. *Ingeniería del software. Séptima edición.* Madrid. : PEARSON EDUCACIÓN. S.A., 2005. ISBN: 84-7829-074-5.

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

Trevenque. 2005. ABCdatos. *ABCdatos.* [Online] Sistemas de Información Trevenque, Julio 6, 2005. [Cited: Enero 17, 2013.] http://www.abcdatos.com/programas/programa/z1980.html.

Veloso, Pedro Hernández. 2004. *Introducción a la Arquitectura de Software.* Buenos Aires : Universidad de Buenos Aires, 2004.

—. 2009. *Uso de Patrones de Arquitectura.* s.l. : UMP, 2009.

GLOSARIO

Beneficiario: Es la persona Natural o Jurídica que realiza el cobro del Título Valor.

Endoso: es lo escrito al dorso de un documento negociable o de otra naturaleza. El endoso es el medio, además de la entrega, por el cual los documentos a la orden pueden negociarse a otra persona.

Letra de Cambio: Título Valor que en sí mismo es un título de crédito, de carácter ejecutivo, documento mercantil, contenido de una orden incondicional de pago. Constituye una promesa de pago.

Librado: Es la persona a la que se da la orden de pago (quien debe pagar), es el destinatario de la orden dada por el librador. Librador Es la persona que emite o da la orden de pago, para cualquier Título Valor.

Librador: Es la persona que emite o da la orden de pago, para cualquier Título Valor.

ORM: (en inglés: Object Relacional Mapping) Persistencia automática y transparente de objetos de una aplicación en una base de datos relacional utilizando metadatos que describen la correspondencia entre el objeto y las tablas de la base de datos.

Pagaré: Título Valor que en sí mismo es un título de crédito, en el cual la persona que lo subscribe reconoce su obligación de abonar una cierta cantidad en tiempo determinado.

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

Servlet: Su función principal es proveer páginas web dinámicas y personalizadas, utilizando para este objetivo el acceso a bases de datos, flujos de trabajo y otros recursos.

Sistema bancario: Conjunto de instituciones que permiten el desarrollo de todas aquellas transacciones (entre personas, empresas y organizaciones) que impliquen el uso de recursos financieros.