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

Título: Diseño e implementación del módulo Letra de Cambio del Proyecto SAGEB.

Autor: Adalberto Lamothe Pellicier

Tutor: Ing. Yesenia Perdomo Bello

Cuidad de La Habana, julio del 2010.

“Año 52 de la Revolución”

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

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

Adalberto Lamothe Pellicier

_____

Firma del Autor

Yesenia Perdomo Bello

_____

Firma del Tutor

"El éxito debe medirse no por la posición a que una persona ha llegado, sino por su esfuerzo por triunfar."

Booker T. Washington

TUTOR: ING. YESENIA PERDOMO BELLO

Correo electrónico: yperdomo@uci.cu

Síntesis del Tutor:

Graduada en la Universidad de las Ciencias Informáticas. Analista del proyecto SAGEB. Adiestrada. Profesora de la asignatura GSW.

Le dedico este trabajo a mi familia, que es mi principal fuente inspiración, que son los que han estado siempre a mi lado dándome su incondicional apoyo.

A mis hermanos, espero que estén orgullosos de mi y que puedan tomarme como un ejemplo a seguir.

A mis hermanos de la UCI Alexei Rivera Acosta y Leonardo García Sagué.

A los Nigthwalkers.

Y por sobre todo a mi mamá, María Regla Pellicier.

Agradezco a todos los que han logrado hacer que la estancia en la UCI sea más agradable, a mis compañeros de grupo de primer año, en especial al piquete de los Mamut (Roberto, Osley, Ángel, Javico, Yandy), con los cuales he compartido los mejores momentos en la escuela, también le agradezco a todos los que de una manera u otra se han ganado mi cariño y forman parte de mi vida, Carlos Chaviano, Yarida, Arasay, Tania, Bello, Daymi, Rita, Adrielito, Darién, Roniel al piquete del 110 Carlos Sanregre, el Deni, Arnoldo, Raydel, Soto, Alfredo, Alberto Ramírez, Leosdeny, a los Nigthwalker que están y estuvieron Vicente, La O, Ambruster, Osvaldo, Los Montoya, Miravet, Leandro, Gustavo y Anglada. También quisiera agradecer a mis compañeros de apartamento los cuales me han brindado su apoyo todo este tiempo y hecho posible la realización de este trabajo, Albert, Esteban, Ale, Guille, Alberto, Yuniel. También quisiera agradecer por su amistad y su apoyo a las niñas de la facultad 6, Yaima, Kalianny, Ludmila, un agradecimiento muy especial también a mi grupo Los Sensuales de la Pista, por tantas alegrías y logros, en especial a su director Damián Falcón. Agradecer también a mi sobrina Yaima, a quien conocí en esta escuela y desde entonces tenemos las mejores relaciones. Quisiera también darle las gracias al tribunal por toda su ayuda por sus críticas constructivas, por su paciencia y por su comprensión durante el desarrollo del trabajo, a mi tutora Yesenia, a mi oponente Lily por ser una gran profesora y una buena amiga además, por haberme guiado y ser incondicional defensora en todo momento, a Javier por ser un tutor más y estar siempre dispuesto a ayudar en lo que fuera necesario, a mi casi compañero de tesis Yuri, por su ayuda y apoyo en la realización de tan ardua labor, solo tú y yo sabemos por lo que hemos pasado.

Quisiera dar estos agradecimientos especiales a las personas que son muy importantes en mi vida, y están desde antes de entrar a la universidad, a mis hermanos Leonardo García Sagué y Alexei Rivera Acosta, los cuales han sido los pilares que me han sostenido durante estos 5 años, han sido mi inspiración, y los que me han dado la fuerza para llegar hasta donde estoy y ser lo que soy actualmente, si ellos no hubieran estado a mi lado desde el comienzo no creo que hubiera tenido fuerzas suficiente para terminar. A mi hermana Rebeca por significado para mí un ejemplo de amor, respeto y cariño, a mi hermano Ordanis Lamothe Pellicier por haber sido el que siempre estuvo a mi lado en todas las circunstancias, a mis padres Adalberto Lamothe y Leonardo Almeida, por haberme inculcado los valores necesarios para forjarme como un hombre, a mi tía Elbita por haberme cuidado como una madre y por su preocupación constante y por último y más importante a mi madre por haberme dado la vida, su corazón

su cariño, su esfuerzo, su amor, su apoyo, durante toda mi vida, gracias a ella soy quien soy y gracias a ella es que he conseguido todo esto por lo que la considero la autora intelectual de todos mis logros.

RESUMEN

La informatización de la sociedad cubana ha hecho posible que la mayoría de las empresas y entidades de Cuba, se vean obligados a perfeccionar la gestión de sus procesos de negocios y automatizar sus funcionalidades para entrar en el mundo competitivo del comercio y los servicios en el mundo. Las entidades bancarias de Cuba se unen a la idea de informatizar sus funcionalidades, y para ello la Universidad de las Ciencias Informáticas, desarrollará un sistema para la gestión de los procesos que se desarrollan en el Banco Nacional de Cuba.

La Letra de Cambio es uno de los medios de pago más utilizados en el intercambio comercial. Actualmente el proceso de gestión de las Letras de Cambio en el Banco Nacional de Cuba puede consumir más tiempo del que debería, ya que este proceso no se encuentra automatizado, trayendo consigo, que las negociaciones se tornen lentas.

El presente trabajo de diploma propone el Diseño e Implementación del módulo Letra de Cambio del Proyecto SAGEB. A partir de la utilización de herramientas y tecnologías y basándose en los requerimientos previamente definidos se pretende alcanzar una implementación de una solución que gestione los procesos de las Letras de Cambio en el Banco Nacional de Cuba.

Palabras Claves:

Letra de Cambio, SAGEB, Diseño, Implementación

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El crecimiento del comercio ha aumentado notablemente en los últimos años, siendo necesario la utilización de medios de pago para subsidiar las deudas que se establecen entre la compra venta de mercancías, bienes o otras negociaciones establecidas entre entidades o empresas nacionales y/o internacionales. Los bancos desempeñan un papel fundamental debido a que ofrecen seguridad y diversidad en la utilización y manipulación de los medios de pago. Las empresas cubanas se preparan para tomar decisiones sobre políticas de venta, cobros y pagos, mediante el vínculo con los bancos y demás intermediarios financieros. En la actividad de cobros y pagos, se aprecian mejorías, pero es imprescindible seguir dando pasos para perfeccionar la gestión y evitar que la cadena de impagos gravite recurrentemente sobre la vida económica del país.

El sistema que está en explotación actualmente en los bancos cubanos en sus diferentes versiones es el SABIC[1]. Este sistema permite el manejo de las funcionalidades contables, lo que facilita integrar el sistema bancario nacional en términos de transacciones contables y flujo monetario. Pero este sistema está desarrollado sobre una tecnología obsoleta que no permite tener un control de la información que se genera producto de las negociaciones o la utilización de los medios de pago para el intercambio mercantil que se establece entre empresas y entidades cubanas.

Por las razones antes planteadas el Banco Nacional de Cuba le solicitó a la Universidad de las Ciencias Informáticas (UCI) el desarrollo de un software que fuera extensible a todos los procesos que se desarrollan dentro de las gerencias del propio banco, permitiendo también la interacción de esos procesos con otros procesos externos que se desarrollan en los demás bancos y entidades bancarias del país. Es de vital importancia la gestión de las Letras de cambio para el pago de las Cartas de Créditos que entran y salen constantemente al Banco Nacional de Cuba. Por la situación anteriormente planteada se obtiene el siguiente problema a resolver, ¿Cómo optimizar la gestión del título valor Letra de Cambio por parte de los funcionarios del Banco Nacional de Cuba? Para darle solución al siguiente problema se tiene como **Objeto de estudio:** Letras de cambio en entidades financieras bancarias, el cual se encuentra dentro del

[1] (Sistema Automatizado para la Banca Internacional de Comercio)

Campo de acción: Gestión de Letras de cambio en el Banco Nacional de Cuba; teniendo como **Objetivo general**, diseñar e implementar el módulo Letra de Cambio del proyecto SAGEB.

Se plantearon los siguientes **objetivos específicos:**

- Diseñar la solución de software para los requisitos relacionados con el módulo Letra de Cambio.
- Implementar el diseño realizado relacionado con el módulo Letra de Cambio.

Tareas a Realizar:

- Caracterización y comprensión de los procesos relacionados con la gestión de las Letras de cambio en el BNC[2], para lograr un mejor entendimiento del negocio.
- Caracterización de las herramientas, lenguajes y notaciones utilizadas por el proyecto con el objetivo de facilitar y comprender el ambiente en que se desarrollará el software.
- Realización del modelo de diseño del módulo Letra de Cambio del proyecto SAGEB para facilitar el desarrollo del software.
- Realización del modelo de componentes del módulo Letra de Cambio del proyecto SAGEB para el desarrollo del software.

Aportes Prácticos

Como aporte práctico, obtiene una gestión eficiente de los procesos de Letra de Cambio en las entidades bancarias cubanas.

El presente documento se estructura en tres capítulos que muestran la investigación realizada para fundamentar el diseño e implementación de la Letra de Cambio del Proyecto SAGEB.

[2] Banco Nacional de Cuba

ESTRUCTURA DEL DOCUMENTO:

Capítulo 1: Se expone el estado del arte, donde se realiza la fundamentación teórica del tema, el objeto de estudio, se explica el funcionamiento del módulo, los procesos fundamentales y otros detalles considerables.

Capítulo 2: Se exponen las herramientas tecnologías utilizadas para el modelado de la aplicación y otras necesarias para su futura implementación. Se describe el negocio, utilizando como herramienta de modelación Visual Paradigm.

Capítulo 3: Se enfoca en la construcción de la solución y estándares de la implementación. Aquí se construyen las funcionalidades que se definieron en el capítulo.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se encuentran definidos los conceptos relacionados con los procesos de la Letra de Cambio. Se definen los principales conceptos relacionados con la Letra de cambio, así como, se analizan las herramientas y lenguajes utilizados.

1.2. Informatización de la sociedad en Cuba.

Nuestro país ha valorado la idea de ser un país competitivo, eficaz y eficiente; esto sólo es posible aplicando la informatización en todas las esferas y procesos de la sociedad cubana. En este sentido, se ha identificado la necesidad de dominar e introducir en la práctica social las Tecnologías de la Información y las Comunicaciones y lograr una cultura digital como una de las características imprescindibles del hombre nuevo, lo que facilitaría a nuestra sociedad acercarse más hacia el objetivo de un desarrollo sostenible. La aplicación de las nuevas tecnologías como parte de la informatización de la sociedad cubana ha traído consigo el procesamiento de la información, la entrada de datos de los materiales en formato papel a una base de datos automatizada, posibilitando una rápida localización de los datos.

El sistema bancario como parte de esta modernización ha centrado la utilización de estos beneficios en satisfacer la demanda de los nuevos tipos de clientes y servicios que se asocia a cada entidad bancaria. Alguno de los impactos que ha tenido el sistema bancario cubano como parte de la utilización de las nuevas tecnologías para la modernización de sus servicios se detalla a continuación:

- Facilidad en el uso de la información. La existencia de software para procesar textos permiten la utilización óptima del tiempo disponible para trabajar con distintos tipos de documento.
- Búsqueda y distribución de información digitalizada, que se envía a los clientes mediante correo electrónico.
- Elaboración de boletines a partir de la recopilación y procesamiento de información proveniente de instituciones cubanas.
- Notibancos (sobre sucesos que involucran a instituciones bancarias extranjeras, como compra de bancos, financiamientos bancarios, movimientos en las cotizaciones de monedas).

- Publicación de un Portal Interbancario que contiene todo la información que es de interés para el país.

La introducción de las nuevas tecnologías en la banca cubana también ha permitido mejorar gradualmente el servicio en las propias sucursales y facilitar las operaciones de turistas y visitantes extranjeros en general, que pueden encontrar ahora un sistema bancario moderno, prácticamente a la altura de sus similares en el exterior. En este sentido, el impacto sería la apertura de las posibilidades de acceso a la información desde el puesto de trabajo y sus características más relevantes serían la rapidez y la efectividad al cumplir con lo que es solicitado.

Por diversas razones, entre ellas el bloqueo económico y financiero impuesto por Estados Unidos a la isla, la aplicación extensiva de la automatización tuvo lugar en la isla con cierto retraso, en comparación con los deseos y necesidades de la nación. Aunque es válido destacar que a pesar de todo esto se obtuvo un impacto adicional en todo el proceso de implantación de nuevas tecnologías en la banca nacional fue la introducción de la red de cajeros automáticos, con los cuales se facilitó la domiciliación del pago de nóminas, el pago a un sector de los jubilados y pensionados y, más recientemente, el pago de servicios públicos como el teléfono, el agua y la electricidad.

La economía moderna se fundamenta en la posibilidad de efectuar actos de comercio a créditos. La agilidad que caracteriza al comercio es lo que ha forzado al legislador a dotar a los comerciantes de nuevos medios de transferir el crédito de los particulares. El comercio se sustenta con documentos de créditos para respaldar sus operaciones de compra y/o venta. Los documentos de créditos requieren de rapidez, flexibilidad y seguridad jurídica para su transferencia. Los títulos valores constituyen documentos de crédito que son utilizados como medio de pago y en ocasiones como instrumento de crédito.

1.3. Sistema Bancario Nacional

Se ha hecho evidente que las numerosas e importantes transformaciones organizativas y normativas efectuadas y por producirse posteriormente en la economía, demandaban una ampliación y diversificación del sistema bancario y financiero de Cuba capaz de enfrentar, además, el establecimiento de una relación diferente con la comunidad internacional en materia comercial y financiera. Consecuentemente, a partir de

1995, se diseñó e implementó, gradualmente, un sistema encaminado a garantizar el funcionamiento de la economía cubana en las nuevas circunstancias y en el marco de la estrategia antes mencionada.

- Banco Financiero Internacional, S.A. (BFI).
  Desde su constitución hasta la fecha ha operado como un banco comercial, dirigido en lo fundamental a prestar servicios a las entidades jurídicas cubanas y extranjeras, asociaciones económicas, empresas mixtas y a personas naturales cubanas .Presta servicios de créditos a las empresas y mantiene relaciones de corresponsalía con una amplia red de bancos en el extranjero.

- Banco Internacional de Comercio, S. A.
  Brinda una amplia gama de servicios bancarios a entidades cubanas, extranjeras y mixtas. Realiza operaciones en moneda libremente convertible, fundamentalmente relacionadas con el comercio exterior, financiamientos y operaciones de compraventa de moneda.

- Banco Metropolitano, S.A.
  Tiene como objetivo fundamental prestar servicios especiales de banca privada y colateralmente operaciones lucrativas relacionadas con el negocio de banca en moneda libremente convertible y moneda nacional, así como operaciones comerciales a través del Banco Internacional de Comercio, S. A. Sus principales clientes lo forman el Cuerpo Diplomático acreditado en Cuba y firmas extranjeras; extranjeros residentes, permanentes o temporales, en el país y particulares cubanos.

1.4. Conceptos Generales

1.4.1. Titulo valor

Los títulos Valores son documentos mercantiles en los que está incorporado un derecho privado patrimonial. En las compraventas que se realizan dentro del tráfico mercantil, está muy extendida la utilización de algunos Títulos Valores como medio de pago y, en ocasiones, como instrumento de crédito. Dentro de los títulos valores más usados están los cheques, letras de cambio y pagarés (2010) (htt1) (htt3)

1.5. Letra de Cambio

La Letra de Cambio es una orden incondicional de pago dirigida por el librador (o girador) al librado (o girado) requiriéndole que pague a la vista o a un tiempo fijo o futuro determinable una suma definida de dinero, a la orden de, o a una persona específica o al portador (beneficiario). (2010) (htt1) (htt3)

1.5.1 Personas que intervienen

- 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): Quien debe pagar la Letra de Cambio cuando llegue la fecha de vencimiento. El librado puede aceptar o no la orden de pago dada por el librador y en caso de que la acepte, quedará obligado a efectuarlo. Beneficiario (el que cobrará): Es la persona que tiene en su poder la Letra de Cambio y a quien se le debe abonar.

1.5.2. Requisitos que debe contener una Letra de Cambio

- La denominación de Letra de Cambio: Esta indicación debe contenerse en la letra. Debe expresarse en el mismo idioma en que esté redactada la letra.
- La orden incondicional de pagar: La orden de pago no está sometida a ninguna condición. Si el importe está en números y letras y ambos son divergentes será válida la cantidad escrita en letra.
- El nombre del librado: El deudor no asumirá la obligación de pagar hasta que declare cambiariamente su aceptación.
- El nombre del beneficiario: El acreedor, legitimado por la posesión. Es el tenedor de la letra.
- La indicación del vencimiento. Es esencial, determina la fecha en la que se va hacer efectivo el pago de la Letra de Cambio.

- La indicación del lugar de pago: Si no se especifica se entenderá que es el domicilio del librado.
- El nombre y la firma del librador: El emisor de la letra tiene que firmar legible.

1.5.3. Aval

El aval cambiario es un acto escrito por el que una persona se compromete a cumplir la obligación de pago que, por razón de la letra, compete a la persona avalada. El aval ha de ponerse en la propia letra o en su suplemento, mediante la expresión de la palabra "por aval" o cualquiera otra similar, e irá firmada por el avalista. El aval no escrito o documentado fuera de la letra no surte efectos como tal, sin perjuicio de que haya que considerarlo como fianza. El aval deberá indicar quién es la persona avalada, pero si no lo indica se entenderá que queda avalado el librado, y en defecto de éste el librador. Se entiende que el aval alcanza a la totalidad del importe de la letra, pero el avalista puede limitar su responsabilidad a una suma inferior al importe de la letra. El avalista que pague la Letra de Cambio adquirirá los derechos derivados de ella contra la persona avalada y contra las que sean responsables cambiariamente respecto de esta última.

1.5.4. Gestión de cobro de letras de cambio.

El cliente (librador o tomador de la letra) presentará al Banco la solicitud de prestación de servicios con no menos de 15 días antes del vencimiento de la letra (si librador y librado operan en el mismo Banco) y con no menos de 30 días de antelación al vencimiento de la letra (cuando librador y librado operen en diferentes Bancos), a fin de que la misma pueda enviarse en término a la sucursal del librado. Las letras que no se presenten a la aceptación o al cobro dentro del término señalado quedan perjudicadas, así como también si no son protestadas oportunamente. Las letras deben presentarse al cobro y pagarse al tenedor el día del vencimiento y en la moneda designada. Las letras deben ser pagadas por los deudores a su vencimiento, de lo contrario serán protestadas dentro de los 8 días hábiles siguientes. El pago de las mismas puede aceptarse en cualquier momento, no obstante, en caso de no pago invariablemente debe estar protestada antes de las 6 pm del 8vo día hábil. Las letras de cambio tienen como todo documento un tiempo establecido para su prescripción, que en este caso prescriben a los 3 años de su vencimiento.

1.5.5. Formas de vencimiento aplicable a la Letra de Cambio

La Letra de Cambio puede ser girada apareciendo como fecha de vencimiento cualquiera de las formas siguientes:

A LA VISTA: Cuando la Letra de Cambio es pagadera "a la vista", esto significa que no existe plazo para su vencimiento, y por lo tanto, esta deberá pagarse a su presentación. También se consideran pagaderas "a la vista", y por el total del valor que representan en su conjunto, las letras con vencimientos sucesivos cuando ha dejado de pagarse una de ellas. Otro caso en que las letras de cambio se consideran pagaderas "a la vista", es cuando no se hace indicación en el documento referente a su fecha de vencimiento.

A CIERTO TIEMPO VISTA: Si la Letra de Cambio es girada "a cierto tiempo visto", se anotaran en el lugar destinado a la fecha de vencimiento expresiones como las siguientes: "a diez días vista", "a 30 días vista", "a 60 días vista" o alguna otra que indique el plazo convenido. Lo anterior quiere decir que el documento deberá pagarse después de los días que se especifique contados desde la fecha de su presentación, y cuando haya transcurrido el tiempo señalado deberá efectuarse el pago.

A CIERTO TIEMPO FECHA: En caso de que la Letra de Cambio sea girada "a cierto tiempo fecha", deberá hacerse la anotación en el documento de "a 30 días", "a 60 días", etc. debiéndose entender que estos plazos comienzan a contarse desde la fecha en que el documento es girado.

A DÍA FIJO: La forma más común de girar una Letra de Cambio es con vencimiento "a día fijo", en este caso se determina en forma exacta la fecha en que deberá ser pagado el documento.

1.5.6. Efectos del pago. Pago parcial

El pago de la letra extingue el crédito cambiario para todos los obligados en la letra (librado, librador, endosante, avalista).

El pago ordinario de la letra está protegido por la acción cambiaria directa contra el librado o sus avalistas, que puede ser ejercitada por el tenedor sin necesidad de protesto. La Ley establece que el portador no

podrá rechazar un pago parcial, pero el librado podrá exigir en ese caso que el pago se haga constar en la letra y se le dé recibo del mismo. Debe tenerse en cuenta, además, que la letra parcialmente satisfecha no conserva las acciones cambiarias si no se protesta por el resto no pagado.

1.5.7. Forma de rellenar una Letra de Cambio

Por último, puede resultar útil analizar el contenido de la letra señalando como han de integrarse sus diferentes elementos sobre la reproducción del anverso y del reverso del modelo oficial de la misma. La Ley Cambiaria no exige que la Letra de Cambio sea emitida en un determinado impreso oficial. Sin embargo, las vigentes disposiciones fiscales condicionan el carácter de título ejecutivo de la Letra de Cambio al hecho de que esté emitida en papel timbrado de cuantía no inferior a la que le corresponda.

1.5.8. Anverso

Lugar de libramiento se indicará la plaza en que la letra se emite por el librador. No tiene que coincidir necesariamente con el domicilio del librador. El Importe se expresará, en cifra, la cantidad cuyo pago ordena el librador y la clase de moneda de que se trate. El importe debe coincidir con la cantidad y signo monetario que figure en el espacio número 6. En caso de discrepancia, prevalecerá lo contenido en dicho espacio. La Fecha de emisión se especificará, en número o en letra, el día, mes y año en que se gira la letra por el librador. La Fecha de Vencimiento cuando la letra se emita a fecha fija, se especificará, en número o en letra, el día, mes y año, en que el tenedor de la letra podrá exigir su pago al librado, en otros supuestos, menos habituales, se expresará "a la vista", "a tantos días de la vista", o "a tantos días desde la fecha". El importe se expresará, en letra, la cantidad cuyo pago ordena el librador y la clase de moneda de que se trate. La Persona o entidad, cuando exista domiciliario, se le identificará por su nombre y apellido o por su razón social. Habitualmente el domiciliario es una entidad bancaria, por lo que, se suele indicar la razón social del Banco en que la letra queda domiciliada. La serie se expresará el número de la cuenta bancaria con cargo a la cual el Banco deberá abonar el importe de la letra. Normalmente, se tratará de una cuenta abierta a nombre del librado. El nombre del librado se identificará por su nombre o razón social a la persona a quien se dirige el mandato de pago, así como la población, calle o plaza y número del domicilio del librado. El pago se hará normalmente en el domicilio que figure junto al nombre del librado, excepto cuando se haya domiciliado su pago, en cuyo caso el domiciliario aparecerá en el

espacio. El nombre del librador se consignará el nombre y apellidos, tratándose de personas físicas o la razón social, tratándose de personas jurídicas.

1.6. Diferencias entre la Letra de Cambio y el Pagaré

1.6.1. Letra de Cambio

Contiene 3 elementos personales (Girador, Tomador, y Beneficiario). Es una orden de pago que implica una acción de regreso para el girador. No Genera Intereses.

1.6.2. Pagaré

Titulo Cambiario, con 2 elementos (Suscriptor y beneficiario). Contiene una Orden de pago que implica la obligación directa de la promesa de pago, estipula Intereses.

1.7. Sistemas existentes para la Gestión de la Letra de Cambio

1.7.1. Letras de Cambio 2002 v2.05:

Desarrollada para la gestión y emisión de letras de cambio totalmente configurable y adaptada al formato oficial en vigor. Ofrece la 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 (estos software no son de utilidad para nuestro país por su condición de “software propietarios”). (Bello, 2008)

1.7.2. Letras de Cambio 2007

Descripción: Letras de Cambio para Windows es una aplicación creada para poder rellenar e imprimir de la forma más fácil posible Letras de Cambio oficiales en España, usando para ellos solamente este programa y una impresora conectada al equipo. Fecha de actualización: 20-08-2008. Permite trabajar con una base de datos personalizada, en la cual se carga la información necesaria de clientes y vencimientos, e imprimirla en el momento exacto. Es muy útil para evitar perder tiempo rellenándola manualmente. Esta aplicación no crea letras de cambio, sino que sirve para rellenar los datos de las mismas. Las letras de cambio oficiales se compran solamente en entidades autorizadas a tal fin.

Valoración

Al no existir en nuestro país un sistema con características similares a los mostrados anteriormente hace que la gestión de procesos en la economía cubana se haga decadente e ineficiente. Como no sería factible el uso de estos sistemas debido a su condición de ser software propietarios, que traería como consecuencias el gasto en licencias y mantenimientos del mismo; la solución sería crear un software propio.

1.8. Herramientas y Tecnologías

1.8.1. Herramientas de desarrollo de software

Las HDS[3] desempeñan un Importante papel en el desarrollo de aplicaciones. Como parte de la ingeniería de Software, las HDS han experimentado cambios en los últimos años. Aún ante la existencia de numerosas herramientas, es necesario abordar temas genéricos sobre las HDS, identificar tópicos relevantes y sus relaciones para facilitar su comprensión y análisis. Esta investigación en progreso, establece la construcción de una ontología que describe conceptos y relaciones para la investigación en HDS. Parte de las HDS en el contexto de la IS[4]: las disciplinas, las áreas de conocimiento y las herramientas como una capa de la IS, para dar lugar a una exploración mayor sobre las definiciones de HDS, las características que definen sus beneficios y las taxonomías disponibles. Ello procura establecer conexiones y motivar una visión de conjunto para su análisis.

1.8.2. Herramientas CASE

Las herramientas CASE[5] aplicaciones que ayudan a economizar tiempo y dinero en los procesos de desarrollo del software. Para el uso eficiente de estas herramientas es aconsejable tener una correcta organización y una planificada metodología de trabajo. Soporta el desarrollo de prototipos de sistemas, en

[3] herramientas para el desarrollo de software

[4] Ingeniería de software

[5] (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, Ingeniería de Software Asistida por Ordenador)

general, el desarrollo de prototipos de aplicaciones toma varias formas. En ocasiones se desarrollan diseños para pantallas y reportes con la finalidad de mostrar organización y composición de los datos, encabezados y mensajes. Los ajustes necesarios al diseño se hacen con rapidez para alterar la presentación y las características de la interface. Sin embargo, no se prepara el código fuente, de naturaleza orientada hacia procedimientos, como una parte del prototipo. (Alonso, 2010)

1.8.3. Lenguaje de Modelado (UML)

UML[6] es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Se puede aplicar en el desarrollo de software entregando gran variedad de formas para dar soporte a una metodología de desarrollo de software (tal como el Proceso Unificado Racional o RUP), pero no especifica en sí mismo qué metodología o proceso usar. La utilización de UML constituye una ventaja ya que es una notación resultante de la evolución de las notaciones previas en ingeniería de software. Toma los aspectos fuertes de tres metodologías anteriores: OMT, Booch y OOSE. La notación UML se fundamenta en principios de modelado, lo cual es importante para toda implementación de un sistema de información. (Design shop (Danny_Meker), 2010)

1.8.4. Visual Paradigm for UML 2.3

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Visual Paradigm es una herramienta CASE concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas entre otras opciones. Constituye una herramienta software libre de probada utilidad para el analista. Dentro de sus características se aprecia que soporta BPMN y UML versión 2.1. (charting, 2010)

[6] Unified Modeling Language

Las razones por las cuales se decidió desarrollar el sistema con la herramienta visual paradigm son porque ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite modelar diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

1.8.5. Metodología de desarrollo RUP

El proyecto decidió utilizar como metodología de desarrollo RUP[7], para aplicar durante toda la vida de desarrollo del software. RUP incluye artefactos (productos tangibles del proceso) y roles (papel que desempeña una persona en un determinado momento).

RUP es una metodología de desarrollo que tiene como objetivo administrar el ciclo de vida de un proyecto minimizando así los riesgos. La gestión de los requerimientos de un proyecto es sustentada a través del análisis y diseño de Casos de Uso y el modelado es llevado a cabo usando, RUP es un proceso que se caracteriza por:

- Dirigido por casos de uso. Los casos de uso describen los requisitos funcionales del sistema desde la perspectiva del usuario y se usan para determinar el alcance de cada iteración y el contenido de trabajo de cada persona del equipo de desarrollo.
- Centrado en la arquitectura. La arquitectura permite ganar control sobre el proyecto para manejar su complejidad y controlar su integridad. Hace posible la reutilización a gran escala y provee una base para la gestión del proyecto.
- Iterativo e incremental. Se divide en 4 fases: Inicio, Elaboración, Construcción y Transición, y cada una de ellas se divide en iteraciones. En cada iteración se trabaja en un número de disciplinas haciendo énfasis en algunas de ellas.

Además de estas características principales cabe destacar las siguientes:

- Desarrollo basado en componentes: La creación de sistemas intensivos en software requiere dividir el sistema en componentes con interfaces bien definidas, que

[7] (**R**ational **U**nified **P**rocess)

posteriormente serán ensamblados para generar el sistema. Esta característica en un proceso de desarrollo permite que el sistema se vaya creando a medida que se obtienen o que se desarrollan y maduran sus componentes.

- Utilización de un único lenguaje de modelado: UML es adoptado como único lenguaje de modelado para el desarrollo de todos los modelos.
- Proceso Integrado: Se establece una estructura que abarque los ciclos, fases, flujos de trabajo, mitigación de riesgos, control de calidad, gestión del proyecto y control de configuración. El proceso unificado establece una estructura que integra todas estas facetas. Además, esta estructura cubre a los vendedores y desarrolladores de herramientas para soportar la automatización del proceso, soportar flujos individuales de trabajo, para construir los diferentes modelos e integrar el trabajo a través del ciclo de vida y a través de todos los modelos. (Alonso, 2010)

1.9. Patrones del Diseño

Los patrones del diseño tratan los problemas del diseño que se repiten y que se presentan en situaciones particulares del diseño, con el fin de proponer soluciones a ellas. Son una solución probada para un problema general de diseño. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes. A continuación, sé mencionaran un conjunto de ellos que contribuyen a que los autores realicen el diseño de la solución de forma rápida y eficaz. (diseño)

1.9.1. Patrones de Asignación de Responsabilidades. (GRASP[8])

Son un conjunto de patrones GRASP, que permite la asignación de responsabilidad en parámetros útiles para el diseño del producto.

Entre los más útiles se encuentran:

[8] (General Responsabilty Assignment Software Patterns)

Patrón Experto: Asignar una responsabilidad al experto en la información, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. Esto provee un bajo nivel de acoplamiento. Es fácil de entender, mantener y manipular.

Patrón Creador: Guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Brinda apoyo a un bajo acoplamiento, lo cual supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento.

Patrón Controlador: El controlador es un intermediario entre la interfaz de usuario y el núcleo de las clases donde reside la lógica de la aplicación. El controlador no realiza mucho trabajo por sí mismo; más bien coordina la actividad de otros objetos. Asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase que represente alguna de las siguientes opciones:

- El sistema global.
- La empresa u organización global.
- Algo activo en el mundo real que pueda participar en la tarea.
- Un manejador artificial de todos los eventos del sistema de un caso de uso.

Patrón Bajo Acoplamiento: Una clase con bajo acoplamiento no depende de “muchas otras” clases. Las clases con alto acoplamiento recurren a muchas clases y no es conveniente. Son más difíciles de mantener, entender y reutilizar. Asigna una responsabilidad para mantener bajo el acoplamiento. Tener la mínima conexión entre los módulos.

Patrón Alta Cohesión: Es la meta principal que ha de buscarse en todo momento. Asigna una responsabilidad para mantener alta la cohesión. Cuan relacionadas y enfocadas están las responsabilidades de un módulo. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de

operaciones, también simplifica el mantenimiento y los mejoramientos. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización. (Guerrero, 2010)

1.9.2. Patrones Estructurales

Facade[9]: El objetivo de la utilización de este patrón es proveer un intermediario reduciendo el número de objetos con los que interactúa un cliente y una interfaz, proporcionando un bajo acoplamiento, además de emitir cambios en el la lógica implementada sin afectar al cliente que la utiliza ya que la misma esta oculta tras la Facade. (HD Lorean, 2010)

1.9.3. Patrones de Comportamiento

Command: El objetivo de utilizar este patrón es tener parametrizados los objetos por las acciones que realizan. Este patrón permite especificar, administrar y ejecutar solicitudes en tiempos distintos. El objeto Command puede guardar un estado que permita deshacer la ejecución del comando. Soporta la capacidad de generar bitácoras que permitan la recuperación del estado en caso de que el sistema falle.

1.9.4. Patrón de Acceso a Datos

DAO[10]: Centraliza todo el acceso a datos en una capa independiente, aislándolo del resto de la aplicación. Desacoplando la lógica de negocio de la lógica de acceso a datos, de manera que se pueda cambiar la fuente de datos fácilmente Sus principales beneficios son que reduce la complejidad de los objetos de negocio al abstraerlos de la implementación real de la comunicación con la fuente de datos. Cada DAO tiene que implementar los métodos declarados en la interfaz DAO correspondiente.

1.9.5. Patrón de Presentación

Composite View (Vistas Compuestas): Un objeto vista que está compuesto de otros objetos vista. Por ejemplo, una página JSP que incluye otras páginas JSP y HTML usando la directiva include

[9] fachada

[10] Data Access Object

MVC[11]: 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 se encarga de presentar la interfaz al usuario, en sistemas web, esto es típicamente HTML, aunque pueden existir otro tipo de formatos. En la vista sólo se deben de hacer operaciones simples y el controlador 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. Cuando se aplica este patrón los accesos a la base de datos se hacen en el modelo, la vista y el controlador no deben de saber si se usa o no una base de datos. El controlador es el que decide que vista se debe de imprimir y que información es la que se envía. (Confuso_Modelo_Vista_Controlador, 2010)

1.10. Ambiente de Desarrollo

1.10.1. Plataforma J2EE

La plataforma J2EE: es un conjunto de herramientas que crean un escenario ideal para el desarrollo y despliegue de aplicaciones escalables en la Web, y que cuentan con las siguientes características:

- Escalable, si tu empresa ve incrementado su número de clientes, nada más sencillo que añadir nuevos componentes J2EE a una aplicación Web para soportar al aumento de clientes, sin tener que reescribir todo el código de nuevo.
- Altamente Soportada, prácticamente cualquier gran empresa de software tiene un contenedor de componentes (o servidor de aplicaciones) Web compatibles con J2EE, entre ellas IBM (Websphere), BEA (WebLogic), Apache (Tomcat), la propia Sun con su nuevo servidor de aplicaciones iPlanet, MacroMedia con (JRun), etc.
- Segura, mientras que otros modelos de aplicaciones empresariales requieren medidas de seguridad específicas en cada aplicación, el entorno de seguridad de la plataforma J2EE permite que se definan unas restricciones de seguridad en el momento de despliegue de la aplicación, aislando así las aplicaciones de la complejidad de las implementaciones de

[11] Modelo Vista Controlador

seguridad, la plataforma J2EE hace portables una gran complejidad de implementaciones de seguridad. (plataforma)

1.10.2. Lenguaje Java

Java: es un lenguaje de programación orientado a objetos, simple, Distribuido, Robusto, Dinámico, Portable y fácil de aprender. Es un lenguaje multiplataforma de código abierto, distribuido, que proporciona un conjunto de clases para su uso en aplicaciones de red, permitiendo abrir sockets y establecer conexiones con servidores o clientes remotos. Además de ser poderoso manejando threads y excepciones. Java fue diseñado para crear software altamente fiable.

1.10.3. Contenedor Web (Tomcat)

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

1.10.4. Base de Datos

SQL Server 2005: provee herramientas sólidas y conocidas a los profesionales de IT, así como también a trabajadores de la información, reduciendo la complejidad de la creación, despliegue, administración y uso de aplicaciones analíticas y de datos empresariales en plataformas que van desde los dispositivos móviles hasta los sistemas de datos empresariales. A través de un conjunto global de características, la interoperabilidad con sistemas existentes y la automatización de tareas rutinarias; ofrece una solución completa de datos para empresas de todos los tamaños.

1.10.5. Eclipse 3.3 IDE

Eclipse: es un IDE[12] para Java muy potente. Fue creado por la IBM bajo la filosofía de software libre. Se está convirtiendo en el estándar de puntera de los entornos de desarrollo para Java. Es Eclipse un marco

[12] (Integrated Development Environment) (Entorno de Desarrollo Integrado)

de trabajo que está compuesto por componentes que se pueden o no incluir en dependencia de las necesidades del desarrollador, a estos complementos se les llama (plugins). De hecho, existen complementos que permiten usar Eclipse para programar en otros lenguajes aparte del Java como son PHP, Perl, Python, C/C++.

1.11. Frameworks y Componentes

1.11.1. Frameworks

Un "Software Framework": es un diseño reusable de un sistema (o subsistema). Está expresado por un conjunto de clases abstractas y el modo en que sus instancias colaboran para un tipo específico de software. Todos los frameworks de software son diseños orientados a objetos" (Degiovan, 2010)

1.11.2. Spring MVC

Se utilizará Spring MVC: para atender las peticiones web simples con navegación lineal. Spring MVC brinda una jerarquía de clases “Controller” para recibir dichas peticiones. Spring MVC ofrece una división limpia entre Controllers, Models (JavaBeans) y Views. Es muy flexible, ya que implementa toda su estructura mediante interfaces no como Struts que obliga a heredar de clases concretas tanto en sus Actions como en sus Forms. Además, todas las partes del framework son configurables vía plugin en la interface, aunque Spring provee clases concretas como opción de implementación. Spring MVC provee interceptores también como controllers que permiten factorizar el comportamiento común en el manejo de múltiples requests. (Bello, 2008)

1.11.3. Spring WebFlow

Spring WebFlow: es un framework que permite manejar la navegación de la aplicación Web. Surge como una respuesta a la funcionalidad limitada del flujo de página ofrecida por los frameworks MVC clásicos. Los flujos Web en este framework son diseñados para ser auto-controlados, dando la posibilidad de definir reglas de las navegaciones (múltiples y complejas). En Spring WebFlow un flujo maneja la conversación (ámbito nuevo que define el vacío entre Sesión y Petición) completa, desde que inicia hasta que culmina y en este punto limpia la memoria automáticamente siempre y cuando se termine el flujo, lo cual es una

mejoría ya que el usuario no tiene que preocuparse por borrar los recursos en memoria que no estén siendo utilizados. Spring WebFlow es un complemento a Spring MVC. Siempre y cuando el caso de uso que se desarrolle presente un flujo complejo y/o no lineal se debe utilizar Spring WebFlow. Cuando existen funcionalidades con flujos complejos y estos son utilizados más de una vez por otras funcionalidades entonces se puede valorar la creación de flujos independientes para que sean reutilizados. (Bello, 2008)

1.11.4. Hibernate

Es un poderoso framework de persistencia de objetos en bases relacionales para la plataforma java. Permite desarrollar un modelo de datos utilizando todo el poder de un lenguaje orientado a objetos incluyendo asociación, herencia, polimorfismo, composición y el framework de colección de java. Hibernate es utilizado para el trabajo con la base de datos. Este framework es considerado un Object Relation Mapping (ORM). Una de las características principales es que abstrae elegantemente el trabajo con la base de datos, soportando la manipulación de los datos persistentes objetualmente, a través de ficheros que mapean clases Java contra tablas. Hibernate es recomendable utilizarlo fundamentalmente en la interacción directa con las tablas de la base de datos. (Bello, 2008)

1.11.5. Dojo Toolkit

Dojo Toolkit es un framework Javascript que permite el desarrollo de aplicaciones web enriquecidas en el cliente y Ajax. Es popular porque está integrada en numerosos IDEs y otros frameworks para desarrollo de webs. Contiene un sistema de empaquetado inteligente, los efectos de UI, drag and drop APIs, widget APIs, abstracción de eventos, almacenamiento de APIs en el cliente, e interacción de APIs con AJAX. Dojo proporciona variadas opciones en una sola biblioteca haciendo un mejor trabajo que sustenta los nuevos y viejos Browsers, resolviendo los problemas de compatibilidad entre los navegadores. Tiene múltiples puntos de entrada, es independiente del intérprete y unifica estándares de codificación. (Sánchez, 2010)

1.12. Conclusiones Parciales

En el presente capítulo se ha tratado de forma general y resumida los diferentes conceptos y procesos que abarcan la gestión de la Letra de Cambio, determinando las ventajas, desventajas y factibilidad de su

utilización en las entidades cubanas. Además, se realizó un estudio de las herramientas, tecnologías y la metodología propuesta por la dirección de arquitectura del proyecto, permitiendo sentar las bases para el desarrollo.

CAPÍTULO 2. DISEÑO DE LA SOLUCIÓN

2.1. Introducción

En este capítulo se define la arquitectura del sistema SAGEB, la cual responderá a las necesidades planteadas en el proyecto de modernización del sistema bancario nacional. Como se había mencionado anteriormente el sistema se desarrolla bajo la tecnología de Java Enterprise Edition, donde el framework que se utiliza como núcleo de la aplicación es Spring Framework. Para la persistencia de datos el framework Hibernate y el almacenamiento se realiza con SQLServer.

La arquitectura que se propuso en el proyecto y sobre la que se está realizando el desarrollo del software está compuesta por una arquitectura de tres capas:

- *capa de presentación.*
- *capa de negocio.*
- *capa de acceso a datos.*

Y por una capa transversal a las otras capas con los objetos del dominio. Esta será para aquellos objetos del dominio que están presentes en resto de las capas, como lo son el módulo de seguridad, el módulo de auditoría, el manejo de transacciones. Dicha arquitectura fue basada en la complejidad de los procesos y la relación entre ellos, por lo que se organizó de forma jerárquica por subsistemas, módulos y componentes; quedando estructurado de la siguiente forma:

Subsistema: Conjunto de módulos relacionados con los procesos que ejecutan.

Módulo: Conjunto de Casos de Uso relacionados con uno o más procesos bancarios.

Componentes: Conjunto de funcionalidades comunes que serán reutilizados por el resto de los módulos del sistema.

2.2. Arquitectura de un módulo

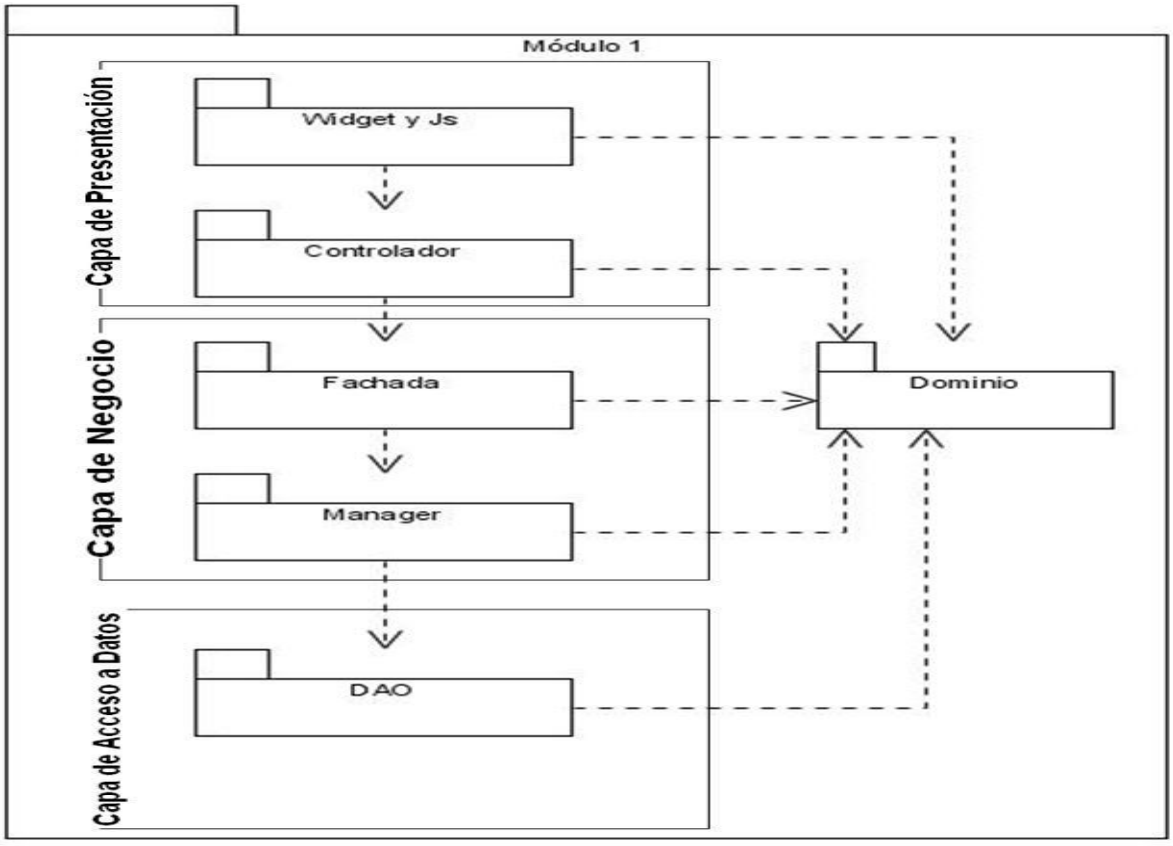

Figura 1. Arquitectura de un módulo del SAGEB.

2.3. Arquitectura de la capa de presentación.

En esta capa se desarrolla la lógica de presentación, en la cual para recibir, controlar y enviar una respuesta desde el cliente se utiliza el Spring MVC. Para hacer menos complejos los flujos de presentación y lograr una buena reutilización de ellos en los diferentes módulos se utiliza Spring WebFlow con el objetivo de representarlos y controlarlos. Y para la generación de las interfaces de las interfaces que interactúan con el usuario se usa la librería Dojo. Esta capa interactúa con la Capa de Negocio y la Capa de Dominio.

2.4. Arquitectura de la capa de negocio.

La capa de negocio del SAGEB está compuesta por dos capas. La subcapa Fachada que será el puente intermediario entre la capa de presentación y la del negocio, donde su función es agrupar las funcionalidades que serán invocadas desde la capa de presentación, sin realizar la lógica del negocio. Y la subcapa Manager donde se realizara la lógica del negocio conteniendo la jerarquía de clases indicadas para su implementación. Esta subcapa hace uso de la capa de acceso a datos para el trabajo con la persistencia de datos y de la capa de dominio para la generación de los objetos del dominio. La siguiente imagen muestra la estructura de la capa de negocio.

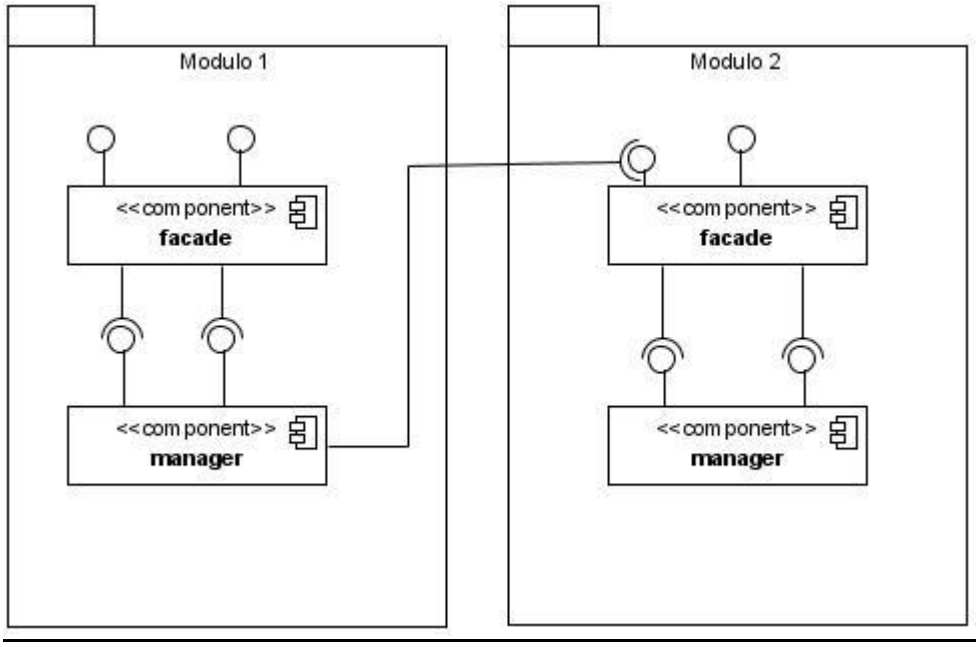

Figura 2. Arquitectura de la capa de negocio.

Nota: Si el paquete facade del módulo 1 necesita funcionalidades que brinda el módulo 2, entonces la invocación de las funcionalidades del módulo 2 se hará mediante su paquete facade.

Estructura de la capa de negocio

Esta capa estará dividida en dos paquetes de clases. Uno será para las clases de la subcapa Facade y el otro para las clases de la subcapa Manager. En el facade estarán todas las interfaces que el módulo

necesite y un subpaquete adicional dentro del paquete facade nombrado impl que contendrá todas las implementaciones de las interfaces declaradas en el facade. De este mismo modo el paquete manager tendrá las interfaces que necesite el modulo y un subpaquete impl con la implementación de dichas interfaces. A continuación se muestra un ejemplo de la estructura de clases dentro de los paquetes mencionados.

2.5. Arquitectura de la capa de acceso a datos

Las operaciones que permitirán el acceso a los datos de la aplicación se encontraran en esta capa, la cual ejerce la conexión con el gestor de base de datos, permitiendo así la persistencia y el acceso a la información. La interacción con la capa de negocio se realizará a través de interfaces. Se utilizará el patrón DAO para el desarrollo de la capa.

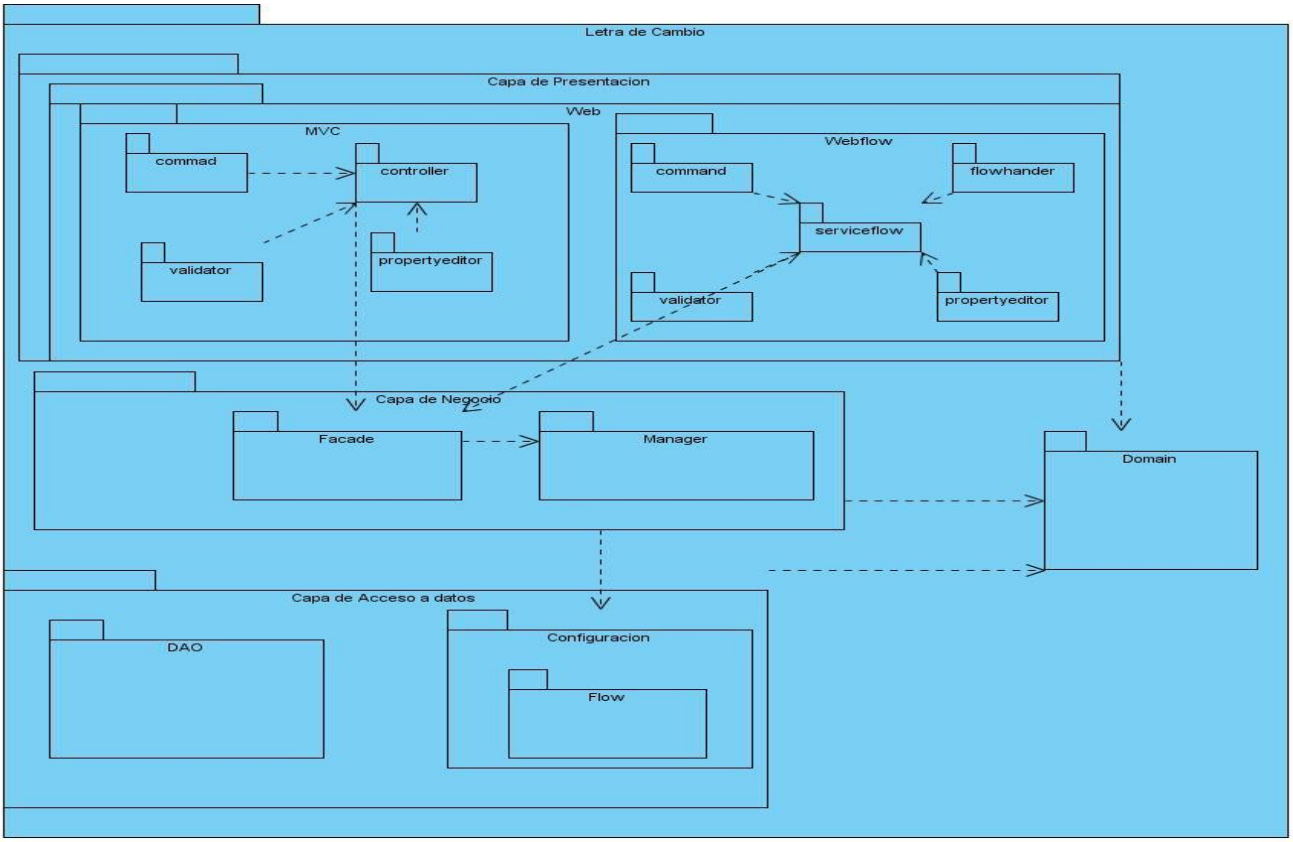

Figura 3. Arquitectura del módulo Letra de Cambio.

Organizar por criterios en paquetes las clases y ficheros de un módulo ayuda a la fácil comprensión de la aplicación, en el caso de Letra de Cambio las mismas se agruparon según la función que cumplen. En breve se mencionan las clases que componen cada paquete.

Paquete configuration: en este paquete estarán los ficheros XML de configuración de los diferentes contextos de Spring:

- dataaccess.xml: contexto de acceso a datos.
- business.xml: contexto de negocio.
- webflow.xml: contexto de Spring WebFlow.
- servlet.xml: contexto de Spring.

Paquete flow: en este paquete estarán los flujos.xml del módulo correspondiente.

Paquete dao: en este paquete estarán las interfaces y las clases de implementen el acceso a datos del módulo, además de los ficheros de mapeos de Hibernate.

Paquete manager: en este paquete se encontrarán las interfaces e implementaciones del negocio del módulo correspondiente.

Paquete facade: en este paquete se encontrarán la interface y la implementación de las funcionalidades que se le brindaran a la presentación.

Paquete domain: en este paquete estarán las clases del dominio del módulo.

Paquete web: este paquete será el contenedor de los paquetes mvc y webFlow.

Paquete mvc: en este estarán los paquetes que contendrán la lógica de presentación en el servidor para SpringMVC.

Paquete commad: contiene clases que representan objetos a manipular en los formularios.

Paquete validator: contiene las clases encargadas de validar los datos.

Paquete propertyEditor: clases para convertir objetos.

Paquete controller: contiene las diferentes clases que heredan de los controladores de Spring.

Paquete webFlow: en este paquete estarán los paquetes que contendrán la lógica de presentación en el servidor para SpringWebFlow.

Paquete serviceFlow: contiene las diferentes clases que median entre el flujo y la facade.

2.6. Diseño de la capa de presentación

En la capa de presentación se hace uso de SpringMVC y SpringWebFlow. En dependencia de la complejidad de cada módulo, se aplicó el framework correspondiente. Donde SpringMVC se utilizó para darle solución a los requisitos que no contemplan la contabilización, como es el caso de:

- Registrar Letra de Cambio.
- Actualizar Letra de Cambio.
- Crear a*sientos contables.*
- Mostrar asientos contables.
- Extraer asientos contables.
- Cobrar comisión.
- Confirmar pago de Letra de Cambio.
- Consultar Letra de Cambio.

Por otra parte SpringWebFlow fue utilizado para darle solución al resto de los casos de usos, que si contemplan la contabilización que es un proceso con flujos complejos:

- Buscar Letra de Cambio.

A continuación se muestra el diseño de la capa de presentación para SpringMVC.

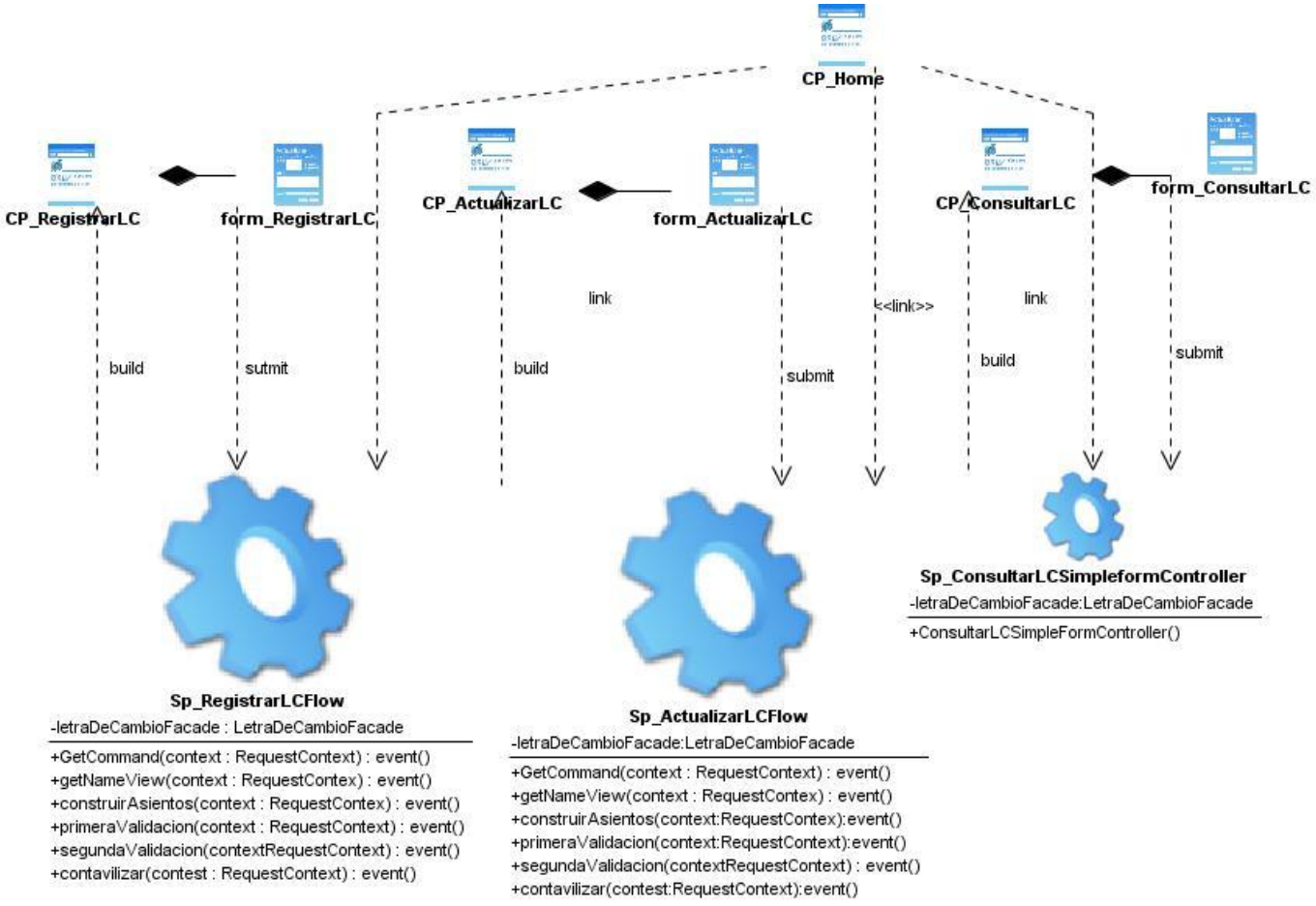

Figura 4. Diseño de la capa de presentación del módulo Letra de Cambio (SpringWebflow).

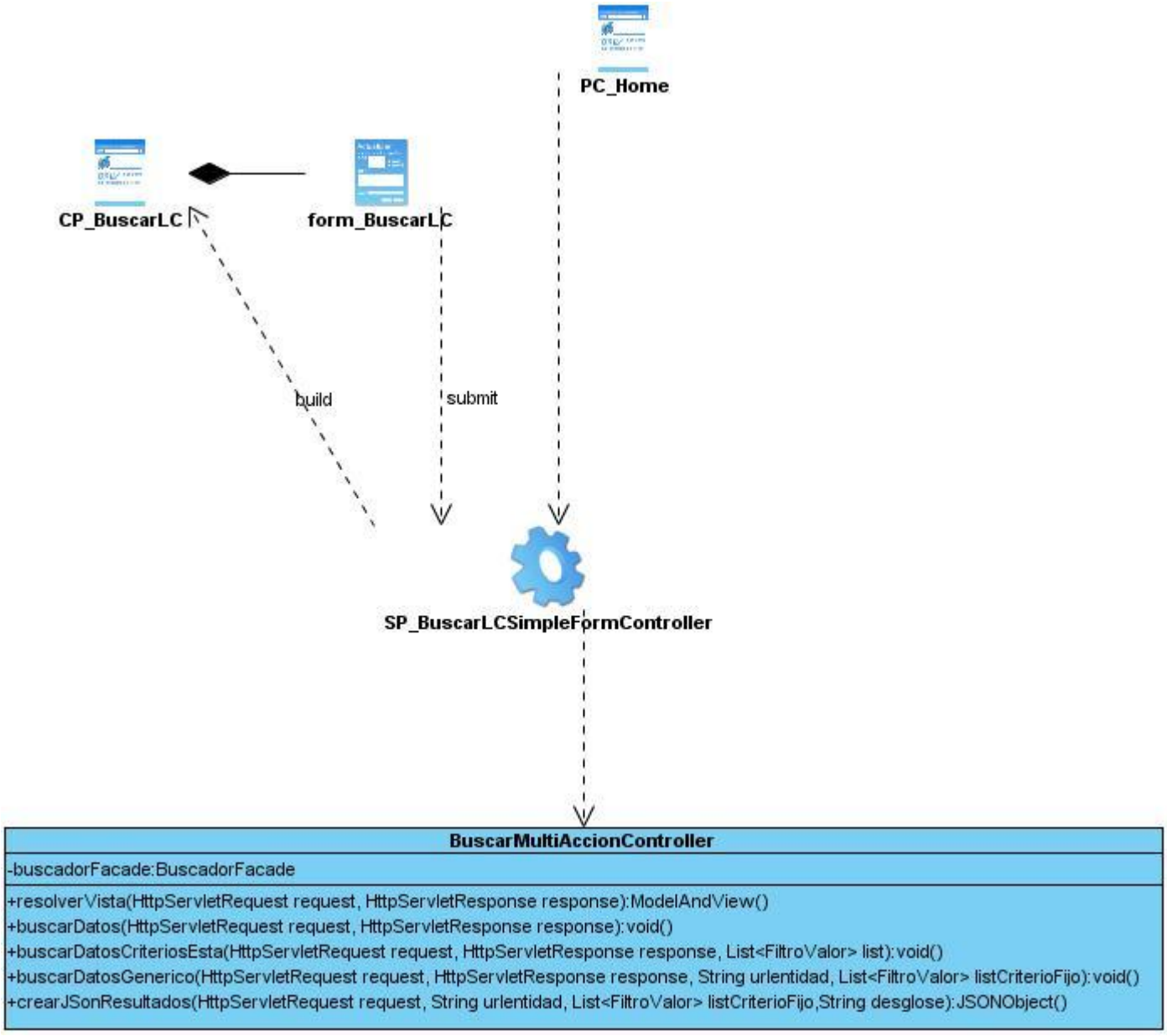

Figura 5. Diseño de la capa de presentación del módulo Letra de Cambio (SpringMVC).

2.7. Diseño de la capa de Acceso a Datos.

En el siguiente diagrama de clases correspondientes a la capa de Acceso a Datos estarán declaradas e implementadas las funcionalidades encargadas de interactuar con la base de datos. Se usara el patrón

DAO así como el componente DAO genérico utilizado en la arquitectura del proyecto, compuesto por la interface y la clase BaseDAO y AbstractBaseDAO respectivamente, las mismas brindan un conjunto de funcionalidades básicas a realizar con las clases persistentes (save, delete, findAll, findById)

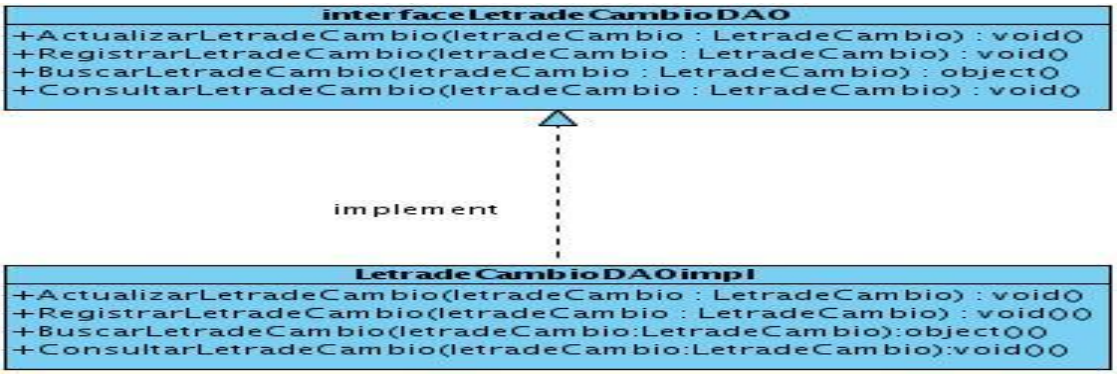

Figura 6. Modelo de Acceso a datos del módulo Letra de Cambio.

2.8 Diagramas de secuencia

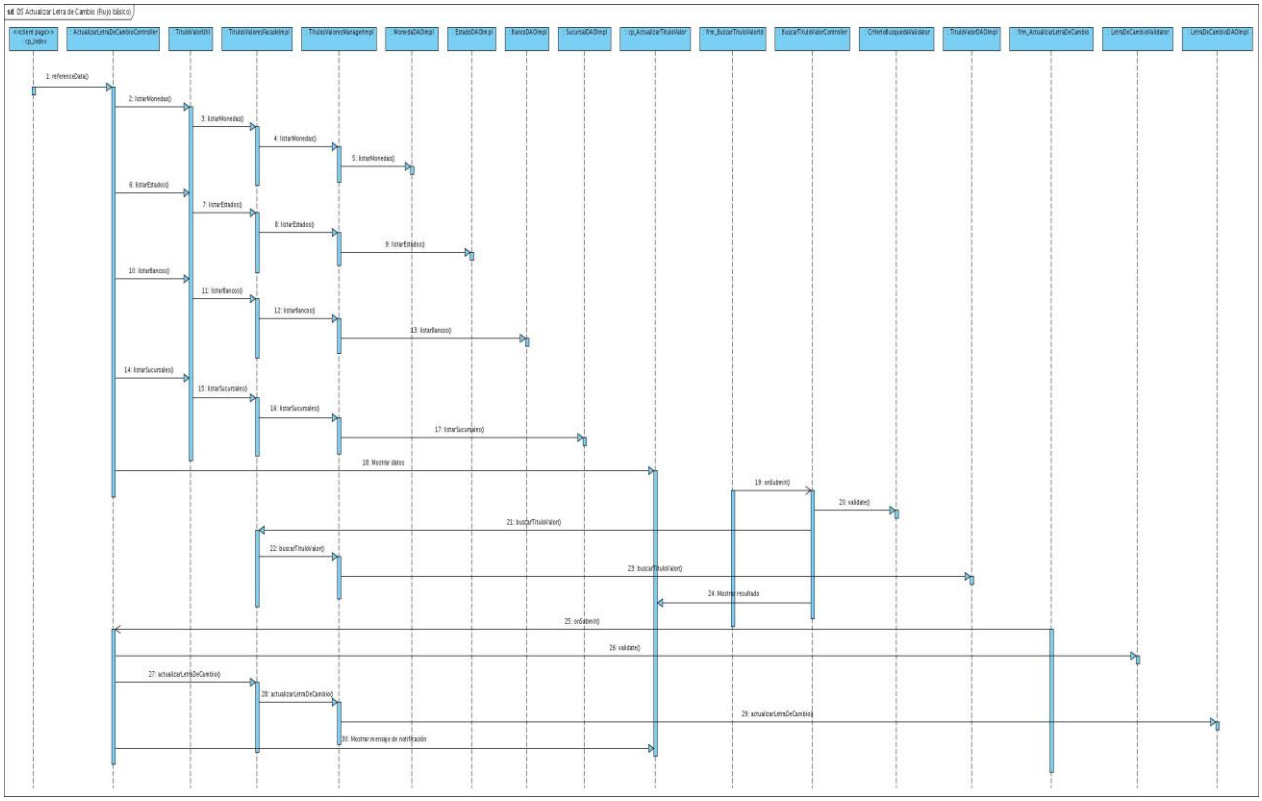

Figura 7. Diagrama Secuencia Actualizar Letra de Cambio (Flujo básico).

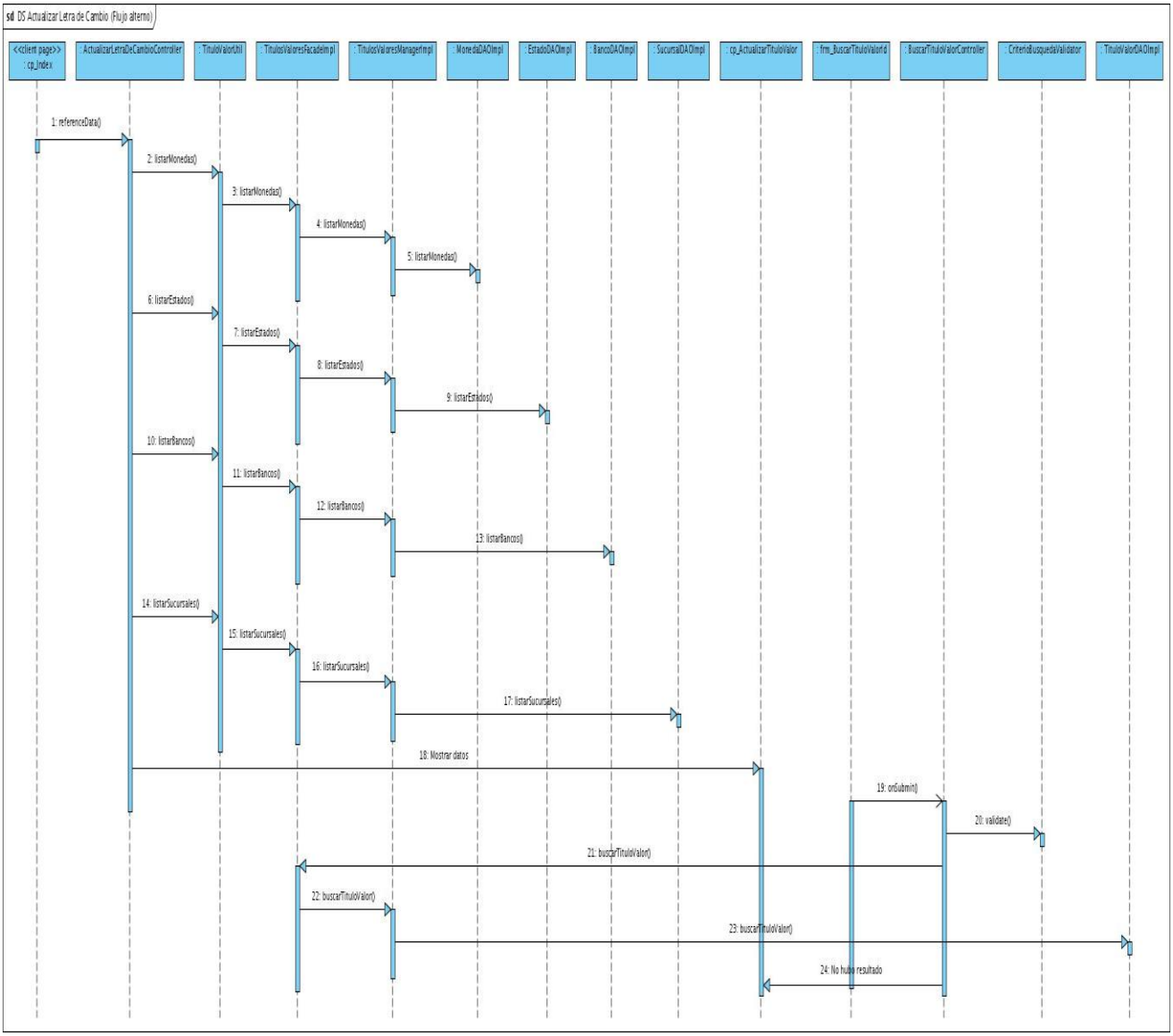

Figura 8. Diagrama Secuencia Actualizar Letra de Cambio (Flujo alterno).

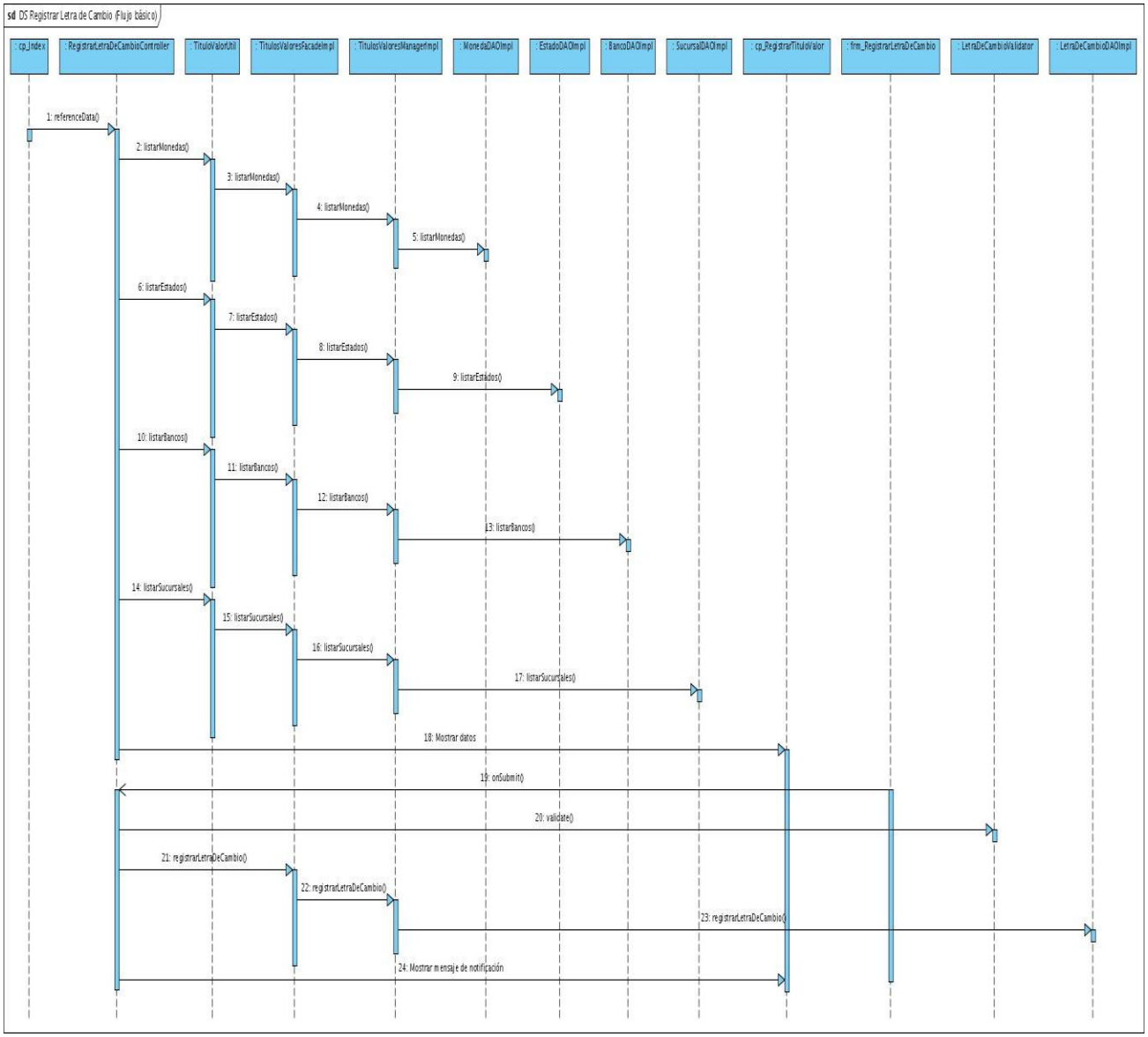

Figura 9. Diagrama Secuencia Registrar Letra de Cambio (Flujo básico).

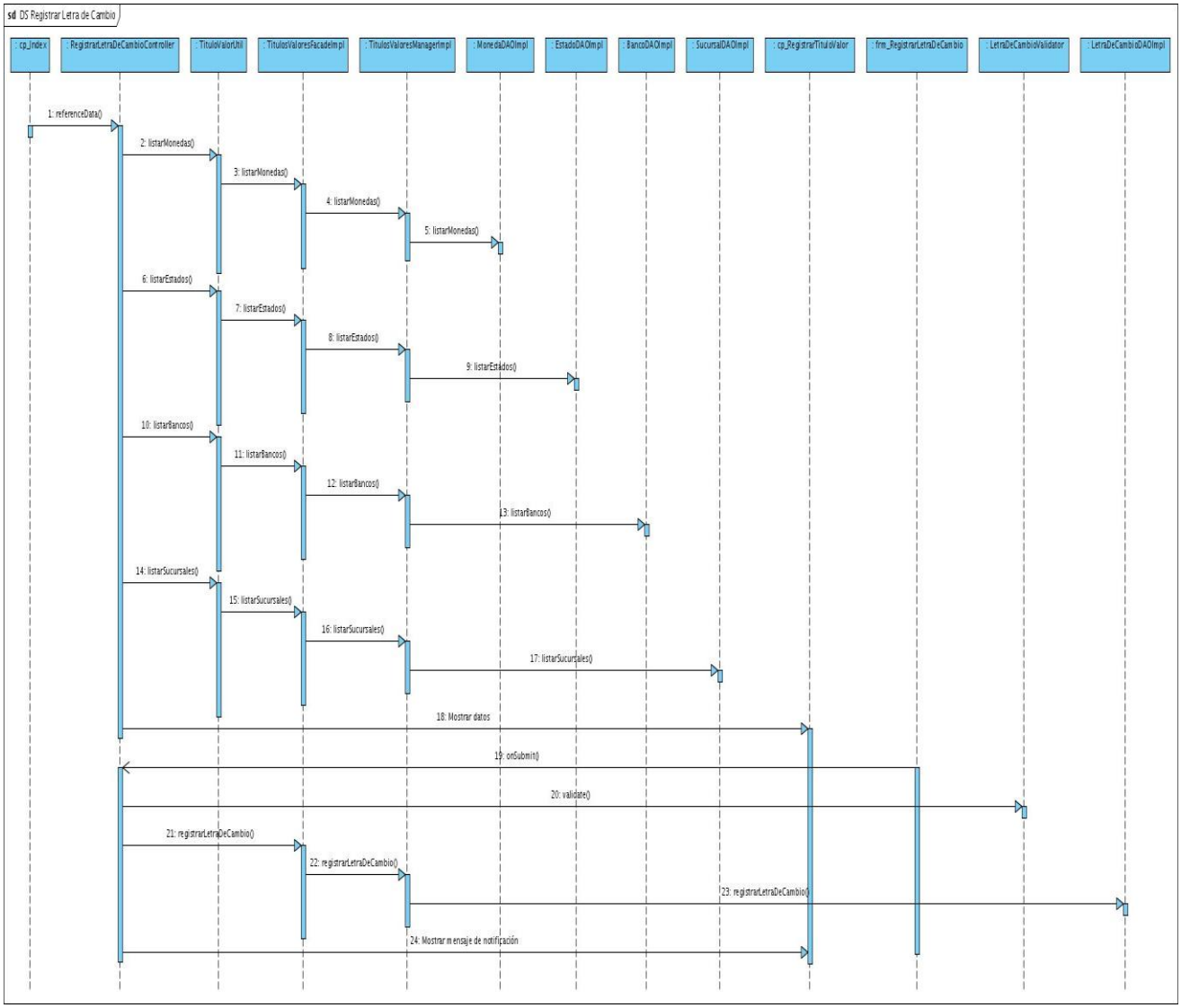

Figura 10. Diagrama de Secuencia Registrar Letra de Cambio (Flujo alterno).

2.9. Modelo de Estados de la Letra de Cambio

La gestión de una Letra de Cambio transita por varios estados, donde una vez emitida ya toma su primer estado hasta llegar a ser pagada o cancelada. En el siguiente diagrama se explica cómo puede ser la vida de una Letra de Cambio.

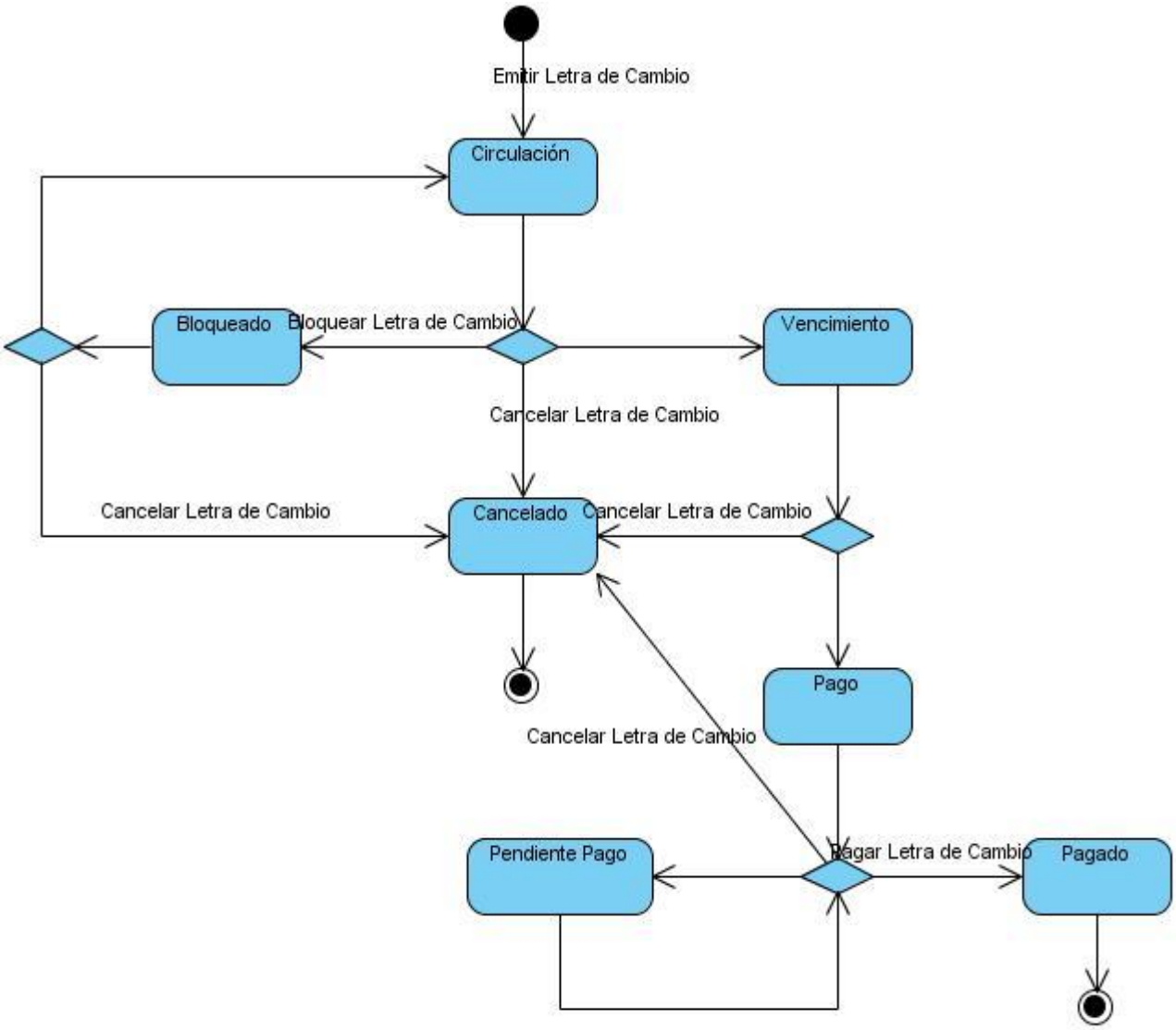

Figura 11. Diagrama de ME del módulo Letra de Cambio.

2.10. Conclusiones Parciales

La panorámica de la arquitectura del proyecto SAGEB dada en este capítulo y el diseño del módulo Letra de Cambio realizado en el mismo, entre otros aspectos facilitó:

- La comprensión de la estructura de paquetes existentes en el módulo.
- Un mejor entendimiento del diseño del módulo lo que implica una uniformidad y organización a la hora de implementar la solución propuesta.

CAPITULO 3. IMPLEMENTACIÓN DEL SISTEMA

3.1. Introducción

En este capítulo se implementa la solución general de gestión de la Letra de Cambio. Se presenta el diagrama de componentes del módulo. Finalmente, se muestran las conclusiones del trabajo, las recomendaciones propuestas, la bibliografía, los anexos y el glosario de términos.

3.2. Utilización de Spring WebFlow Framework

En el capítulo anterior se explica acerca del uso de SpringWebFlow, por las ventajas que este brinda. En el módulo se logra realizar el diseño e implementación de un flujo capaz de responder brevemente el funcionamiento del mismo a partir de los estados más importantes a vistas diferentes e iniciar con diferentes objetos (command). De esta forma, a partir de la declaración de un flujo común, este es capaz de darle respuestas a un conjunto de casos de uso. A continuación se describe en el caso de uso, registrar Letra de Cambio.

Al seleccionar en el menú la opción, “registrar”;

Figura 12: Muestra el menú de Letra de Cambio.

Este botón tiene un vínculo con el método:

```
id: "m54",
 label: "Registrar",
 labelNodeClass: "Level2",
_showBullet: true,
doAction: "true",
action: "finixubnc.core.request({url:'../LetraDeCambio/registrarLetraDeCambio-flow.htm', method:'GET'})"
```

Figura 13: Método del menú registrar Letra de Cambio.

Que a su vez hace una llamada al flujo (“registrarLetraDeCambio “) donde se encuentra el método:

```
    <transition on="aceptar"  validate="false" to="primeraValidacion">
    </transition>
    <transition on="cancelar" validate="false"  bind="false" to="end"/>
    <transition on="verAsientos" to="verAsientosState"/>
    <transition on="actualizarDatosComisiones" validate="false" bind="false" to="registrarLetraDeCambio">
        <evaluate expression="comisionesMultiAction.actualizarDatosComisiones" />
    </transition>
    <transition on="gestionarComision" validate="false" to="gestionarComisionSubflow">
        <evaluate expression="comisionesMultiAction.registrarCallMetodo" />
    </transition>
</view-state>

<subflow-state id="gestionarComisionSubflow" subflow="gestionarComision-flow">
    <transition on="end" to="registrarLetraDeCambio">
        <evaluate expression="comisionesMultiAction.registrarCallMetodo" />
    </transition>
</subflow-state>

<action-state id="primeraValidacion">
  <evaluate expression="RegistrarLetraDeCambioMultiAction.primeraValidacion"/>
  <transition on="success"  to="resultadoPrimeraValidacion" />
</action-state>

<decision-state id="resultadoPrimeraValidacion">
     <if test="flowScope.codigoMensaje=='m0'" then="segundavalidacion" else="registrarLetraDeCambio"/>
</decision-state>

<action-state id="segundavalidacion">
  <evaluate expression="RegistrarLetraDeCambioMultiAction.segundaValidacion"/>
   <transition on="success"  to="resultadoSegundaValidacion" />
</action-state>
```

Figura 14: Método en el cual se levanta la vista.

El cuál pasará desde una primera validación contable hasta lograr una correcta contabilidad, luego se ejecutara el método que levantara la vista como se muestra en la Figura 14, mostrándole al usuario los campos para que pueda proceder al llenado de la Letra de Cambio, como se muestra en la figura 14.

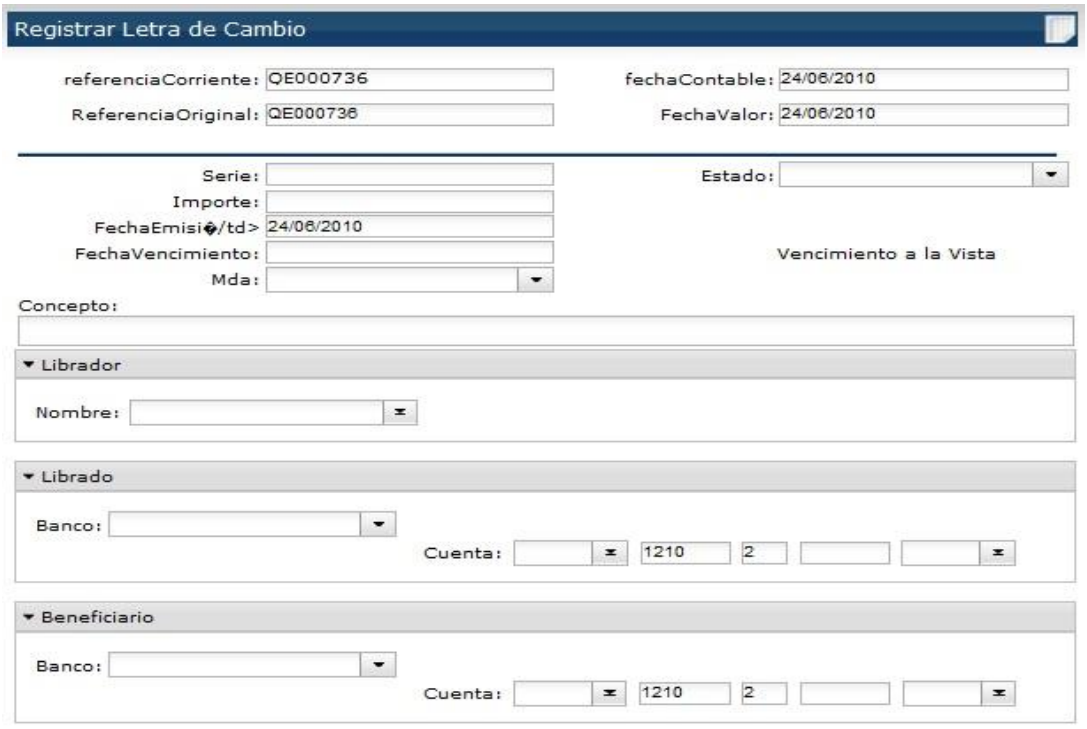

Figura 15: Vista Registrar Letra de Cambio.

Aquí el usuario llenara los campos de la Letra de Cambio con los datos necesarios; una vez que estén llenos todos los campos el usuario podrá ejecutar la opción aceptar donde se llamara al siguiente vínculo:

Una vez registrador los datos el usuario puede elegir la opción que desea realizar, ya sea actualizar, consultar o cancelar.

Figura 16.Opciones que tiene el usuario de elegir.

<transition on=”aceptar” valídate=”false” to=”persistirletraDeCambio”>, en el cual se hará otro vínculo al método, “persistirletraDeCambio”, el cual estar vinculado con el método:

```
<bean id="RegistrarLetraDeCambioMultiAction"
    class="cu.uci.finixubnc.titulosvalores.LetraDeCambio.web.webflow.serviceflow.RegistrarLetraDeCambioMultiAction"
    <property name="letraDeCambioFacade" ref="letraDeCambioFacade"></property>
</bean>
```

Figura 17: Método donde se llama a la clase registrarLetradeCambio.

Antes de llamar al método persistirletraDeCambio" el flujo pasa por 2 métodos de validación y 1 para contabilizar. Es decir, al usuario presionar en el botón aceptar se hacen las llamadas a los métodos de validación los cuales están enlazados entre si y dan paso al de contabilización, si no hay errores en estos. Luego de realizar la contabilización entonces se llama al método persistir y se guarda el objeto en la base de datos.

```
public Event persistirletraDeCambio(RequestContext contex) throws Exception
{
     System.out.println("Entro - RegistrarChequeMultiAction.persistirCheque()");
     String cuentaLibrados=contex.getRequestParameters().get("cuentaLibrados");
     String cuentabeneficiario=contex.getRequestParameters().get("cuentabeneficiario");
     String cuentaavalado=contex.getRequestParameters().get("cuentaavalado");
     letreDeCambioComand letraCambioComand = (letreDeCambioComand) contex.getFlowScope().get("letraDeCambioComand")
     letraCambioComand.setLibrado_cuenta(cuentaLibrados);
     letraCambioComand.setAvalado_cuenta(cuentaavalado);
     letraCambioComand.setBeneficiario_cuenta(cuentabeneficiario);
     LetraDeCambio letraDeCambio =   letraDeCambioFacade.conformarletraDeCambio(letraCambioComand);
     System.out.println(" ya convirtio la dichosa letra cambio" +letraDeCambio.getId_letra_cambio());
     letraDeCambioFacade.registrarLetraDeCambio(letraDeCambio);
     System.out.println("Salio - RegistrarChequeMultiAction.persistirCheque()");
     return success();
 }
```

Figura 17. Método por el cual se registrará una Letra de Cambio.

Y final mente quedará registrada la Letra de Cambio en la base de datos.

Al terminar la ejecución de este flujo se regresa al estado de vista inicial. Descripción de las clases y funcionalidades del subsistema. Aquí se muestran sólo los métodos más significativos:

Tabla 1. Clase contabilizarLetraDeCambioMultiAction.

Nombre: ActualizarLetraDeCambio	
Tipo de clase: Controladora	
Atributo	Tipo
letraDeCambioFacade	LetraDeCambioFacade
Para cada responsabilidad:	

Nombre:	Descripción:
RegistrarEntradaFlujo(RequestContext contex)	Busca la Letra de Cambio que contenga el parámetro de búsqueda.
construirAsientos(RequestContext contex)	Construye los asientos en función del caso de uso, dados los parámetros del mismo.
Primeravalidación(RequestContext contex)	Valida los asientos construidos y notifica los errores en caso de tenerlos.
Segundavalidación(RequestContext contex)	Realiza una segunda validación a los asientos contables y notifica los errores encontrados.
FormarMensajes(List<String> mensajes)	Devuelve una Lista de mensajes
Contabilizar(RequestContext contex)	Contabiliza los asientos listados y notifica los errores encontrados.
esOperacionContable(RequestContext contex)	Realiza el proceso de contabilización
persistirCCredito(RequestContext contex)	Crea la entidad Letra de Cambio en la tabla de la base de datos

Tabla 2. Clase modelo Letra de Cambio.

Nombre: LetraDeCambio	
Tipo de clase: Modelo	
Atributo	Tipo
id	String
fechaValor	Java.util.Date
fechaContable	Java.util.Date
importe	Bigdesimal
referenciaOrigina	String

moneda	nomMoneda
estado	nomMonedaLetradecambioPagare
FechaEvmicion	Date
fechaVencimiento	Date
concepto	String
librador	Banco
libradoBanco	Banco
libradoCuenta	Cuenta
beneficiariocuenta	string
beneficiarioBanco	Banco
avalado	String
avaladoBanco	Banco
avaladoCuenta	String
observaciones	String
cuentaAcreditar	String
contabilizar	Boolean

3.3. Diagrama de componentes

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes. Los componentes físicos incluyen archivos, cabeceras, librerías compartidas, módulos, ejecutables, o paquetes.

Los Diagramas de Componentes ilustran las piezas del software, controladores embebidos, etc. que conformarán un sistema. Un diagrama de Componentes tiene un nivel más alto de abstracción que un diagrama de clase usualmente un componente se implementa por una o más clases (u objetos) en tiempo de ejecución. Estos son bloques de construcción, como eventualmente un componente puede comprender una gran porción de un sistema. (2010)

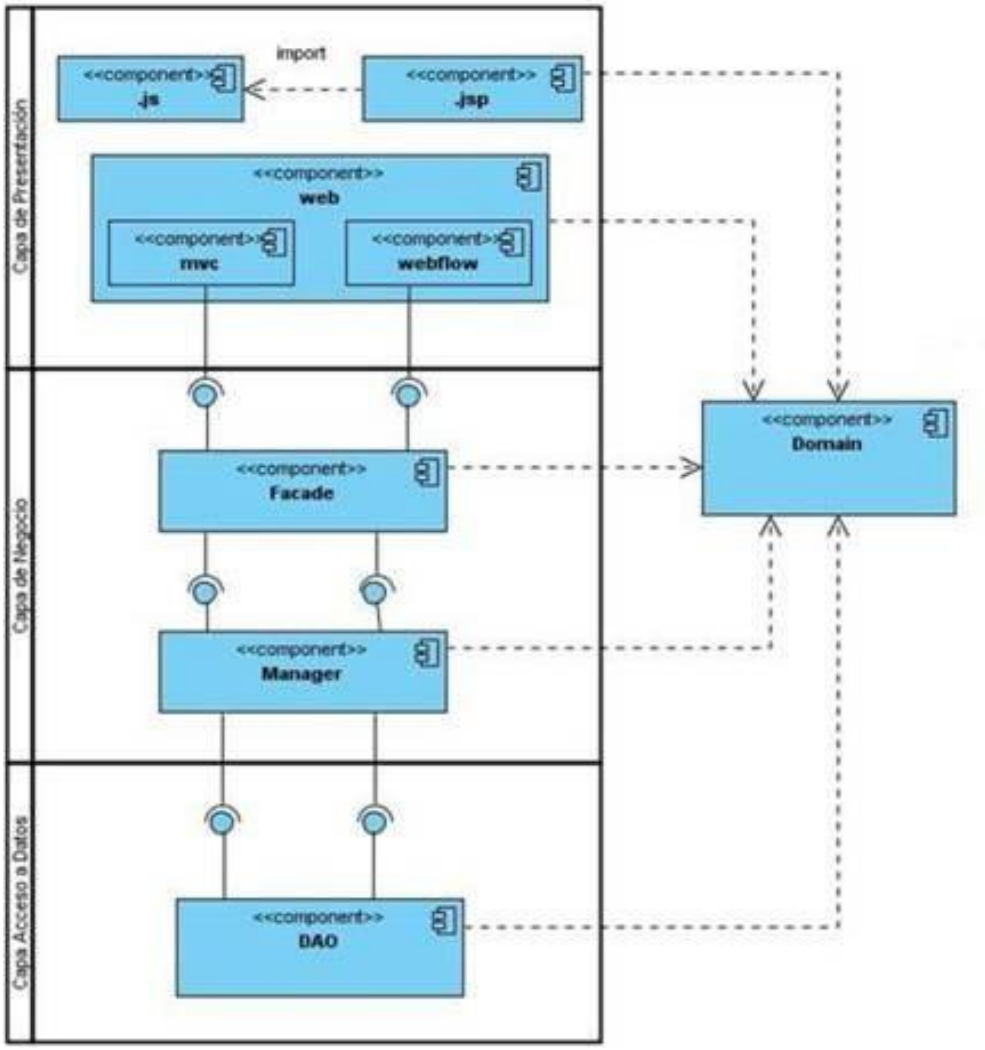

Figura 18. Diagrama de componentes del módulo Letra de Cambio.

3.4. Conclusiones parciales

Con el análisis del caso de uso descrito anteriormente se evidenció el trabajo con sprinWebflow y la importancia que presenta el mismo para el sistema, además se plantearon los principios para el diseño, el tratamiento de errores y los estándares de la implementación.

CONCLUSIONES

Con el presente trabajo se obtuvo como resultado el diseño y la implementación del módulo Letra de Cambio, además el mismo será integrado al SAGEB para así ir completando el software que pasara a ser el próximo sistema contable usado en los bancos de nuestro país. Brindando así una mejor calidad y eficiencia y seguridad en el trabajo de los funcionarios del Banco nacional de Cuba.

RECOMENDACIONES

Se recomienda que se le realicen les pruebas pertinentes a este módulo; la realización de nuevas versiones para ampliar las funcionalidades de la solución propuesta y que se integre a sistemas con iguales características.

REFERENCIA BIBLIOGRÁFICA

2010Cota A. 1994 "Ingeniería de Software". Soluciones Avanzadas. Julio de 1994. pp. 5-13”.

2010buenmaster.com

2010benjaminstrahs*benjaminstrahs*http://descargar.benjaminstrahs.com/d-d-view87584/LETRAS%2BDE%2BCAMBIO/es.htm.

2010ciberaula*ciberaula*http://www.ciberaula.com/curso/java2/que_es/

2010ciberaula.com*ciberaula.com*http://java.ciberaula.com/articulo/diseno_patrones_j2ee/.

2010desarrolloweb.com*desarrolloweb.com*http://www.desarrolloweb.com/wiki/dojo.html

Design shop(Danny_Meker)2009taringa*taringa*http://www.taringa.net/posts/ebooks-tutoriales/3598122/Ultra-Pack-Para--Aprender-a-Programar-en-Todos-Los-Lenguaje.html

2009dsi.uclm.es*dsi.uclm.es*http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf

2006forums.visual-paradigm.com*forums.visual-paradigm.com*http://forums.visual-paradigm.com/posts/list/1712.html

2007freedownloadmanager*freedownloadmanager*http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%5Bcuenta_de_Plataforma_de_Java_14715_p/

HD Lorean 2008wikidot*wikidot*http://hdlorean.wikidot.com/doc:patrones-diseno

2010hibernate.org*hibernate.org*http://www.hibernate.org/

Hista Internacional S.A2007histaintl.com*histaintl.com*http://www.histaintl.com/servicios/consulting/rup.php

2010in.relation.to*in.relation.to*http://in.relation.to/Bloggers/HibernateSearch32ReleasedMappingMassIndexingClustering

2007javahispano*javahispano*http://www.javahispano.org/contenidos/es/comparativa_de_frameworks_web/#recurso-1

2007manual-java.com*manual-java.com*manual-java.com/manualjava/caracteristicas-java.html

Microsoft2005microsoft.com*microsoft.com*http://www.microsoft.com/spain/sql/productinfo/overview/default.mspx

2009monografias*monografias*http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml

2010*MVC*2010

2010sparxsystems.com*sparxsystems.com*http://www.sparxsystems.com.ar/resources/tutorial/uml2_componentdiagram.html

2010sqlmag.com*sqlmag.com*http://www.sqlmag.com/

2010visual-paradigm.com*visual-paradigm.com*http://www.visual-paradigm.com/aboutus/newsreleases/archivedevents.jsp

2008*y Diseño del Subsistema Títulos Valores del Proyecto Modernización del Sistema Bancario Cubano* Ciudad de La Habana2008

ANEXOS

Figura 18. Registrar Letra de Cambio.

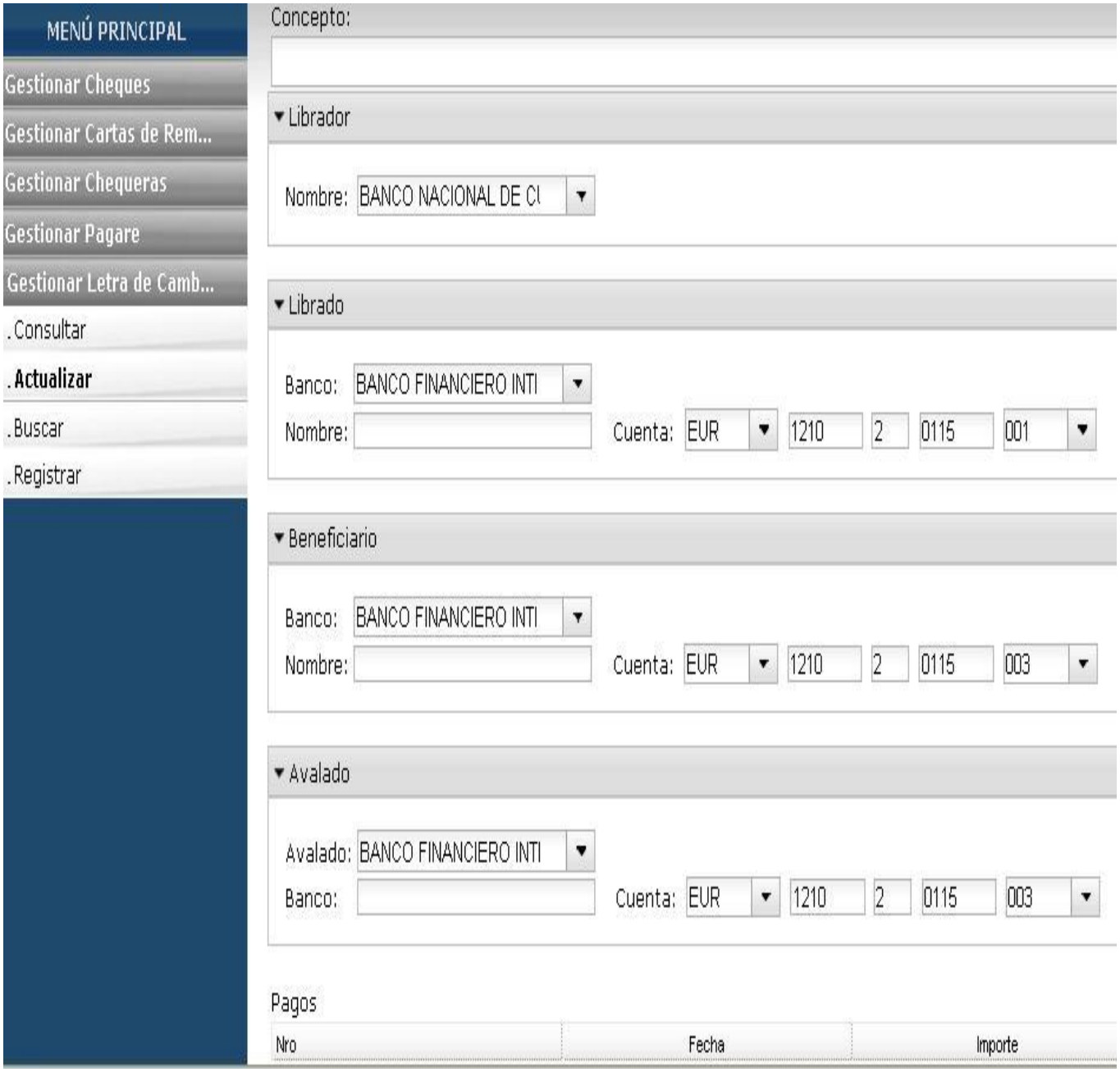

Figura 19. Actualizar Letra de Cambio.

Figura 20. Adicionar comisiones

Figura 21. Actualizar comisiones

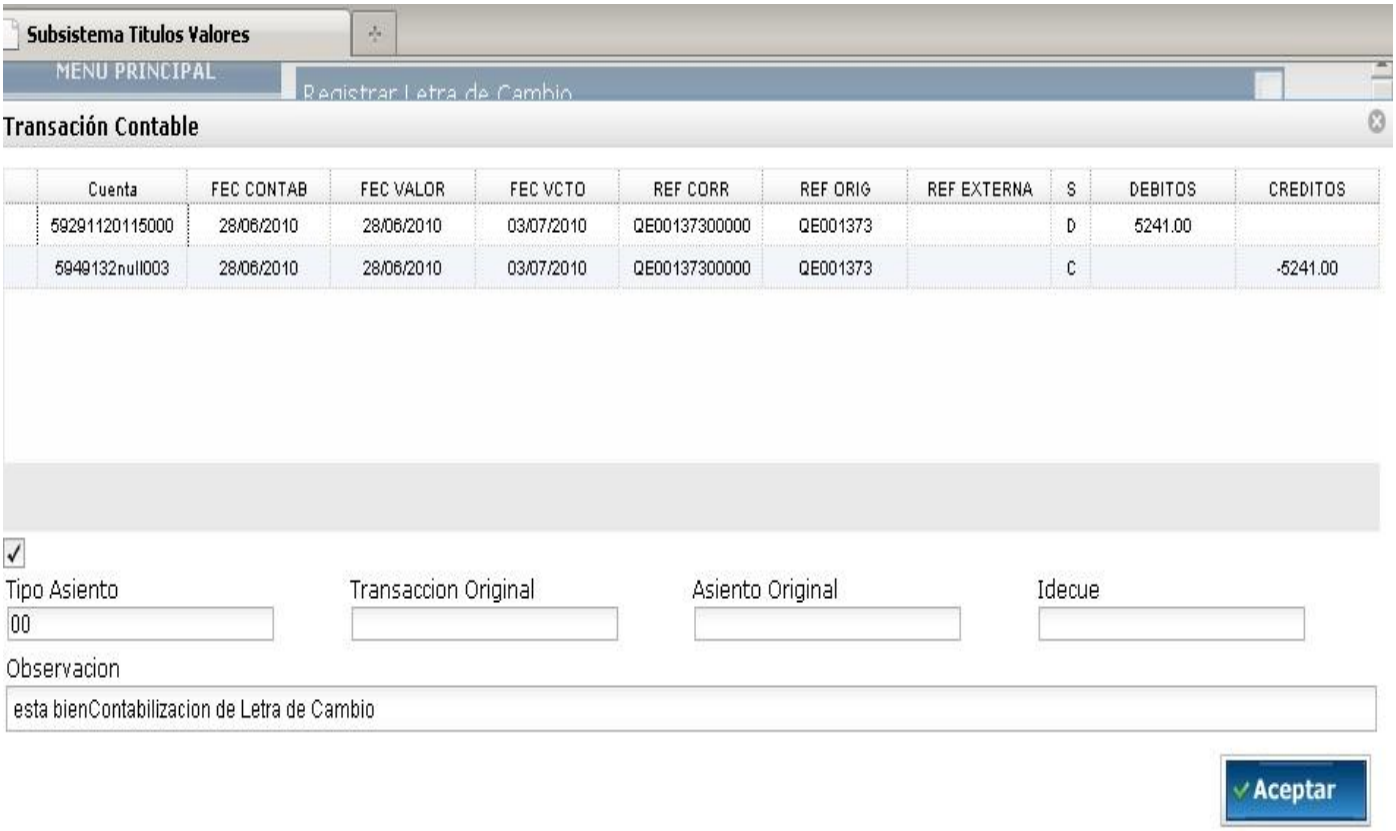

Figura 22. Asientos contables

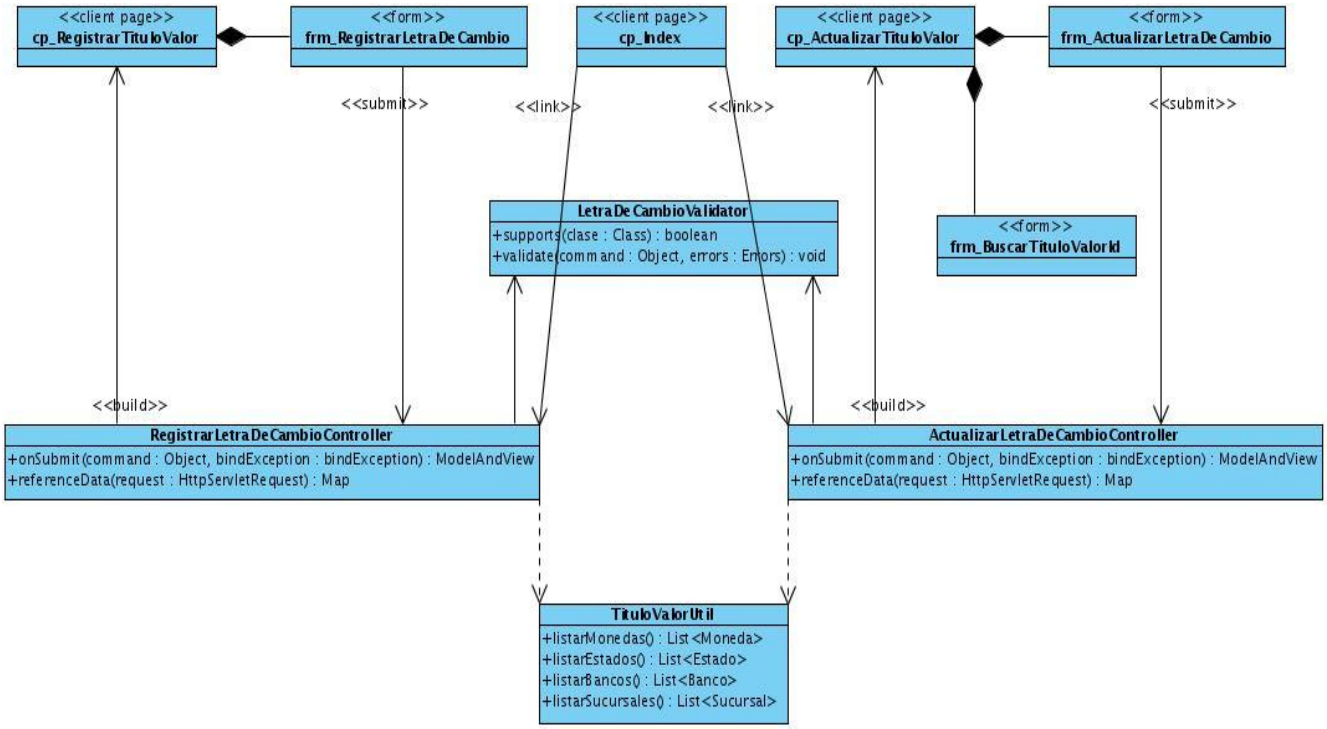

Figura 23. Subcapa de presentación del módulo Letra de Cambio.

GLOSARIO DE TÉRMINOS

Command: Objeto que representa los campos de una vista.

CRUD: Es la forma a referirse a las operaciones básicas en la base de datos (Create, Retrieve, Update y Delete) en inglés.

Plugin: es un módulo de hardware o software que añade una característica o un servicio específico a un sistema más grande.

JavaServer Page (JSP): Tecnología orientada a crear páginas web con programación en Java.