Universidad de las Ciencias Informáticas

Facultad 3

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

Título: Diseño e Implementación de la nueva versión del Módulo Banco del Sistema Integral de Gestión Cedrux.

Autor:

Jorge Carlos Driggs Vélez

Tutores:

Ing. Enrique Chaviano Gómez

Ing. Pedro Antonio Torres Salas

Ciudad de La Habana, __ de _____ del 2011.

"Año 53 de la Revolución"

Declaración de autoría

Declaro que soy el único autor del trabajo: Diseño e Implementación de la nueva versión del Módulo Banco del Sistema Integral de Gestión Cedrux y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste, firmo la presente a los ____ días del mes de _____ del año 2011.

_____
Jorge Carlos Driggs Vélez
Autor

_____
Ing. Enrique Chaviano Gómez
Tutor

_____
Ing. Pedro Antonio Torres Salas
Tutor

"…todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el hombre nuevo que se vislumbra en el horizonte…"

Agradecimientos

A mis tutores por ayudarme en todo lo que estaba a su alcance y sin los cuales no hubiera podido hacer mi tesis.

A mi familia por apoyarme y darme aliento para seguir.

A mis amigos Álvaro, Rolando y Asdrubal por estar ahí para molestarme cada vez que podían.

A mis compañeros de cuarto por soportarme y aguantarme durante tanto tiempo.

A todos mis demás amigos que de una forma u otra me ayudaron en estos 5 años.

A mi equipo del proyecto con los que trabajé mano a mano y contribuyeron en la realización de esta tesis.

A la Revolución y a Fidel sin los cuales no hubiera podido estudiar esta carrera.

A todos…

¡Gracias!

Dedicatoria

A mi abuelo Luis, no llegó a verme graduado de ingeniero y aunque ya hoy no está conmigo sé que era uno de sus principales anhelos.

A mis padres por ser mis guías y estar siempre ahí cuando los he necesitado, en cualquier circunstancia y porque me enseñaron a ser como soy y sin los cuales no hubiera logrado llegar hasta aquí.

A mi hermana por ser la persona que más quiero en el mundo.

A mis abuelos Germán y Marcelina que siempre me inculcaron los mejores valores y me dieron las mejores enseñanzas.

A mi abuela Celinda por apoyarme siempre y ser la abuela más linda del mundo.

Resumen

En Cuba se emplean sistemas de gestión empresarial que mejoran el funcionamiento de las actividades empresariales, sin embargo, estos sistemas no cumplen con la totalidad de los requisitos funcionales para las operaciones bancarias y la independencia tecnológica que necesita el país.

Los sistemas de gestión empresarial permiten gestionar recursos bancarios dentro de la actividad financiera mediante el control de las cuentas bancarias, la emisión de los documentos necesarios para operar sobre las cuentas y las diferentes operaciones contables generadas a partir de la realización de los estados de cuentas y las conciliaciones.

Actualmente se desarrolla el Sistema Integral de Gestión Cedrux, que integra la gestión de actividades bancarias pero presenta deficiencias en cuanto a la gestión funcional de operaciones como la gestión de las cuentas bancarias, la gestión de la configuración, el reconocimiento de cobros y pagos anticipados, el establecimiento de contratos de respaldo para los instrumentos de pago y la liquidación de derechos fiscales y de cobro.

En el presente trabajo se realiza el diseño e implementación de nuevas funcionalidades que solucionen las deficiencias detectadas en la etapa de prueba del módulo para mejorar la solución. El diseño está regido por el modelo de desarrollo orientado a componentes establecido y para la implementación se emplean las herramientas y tecnologías establecidas para el sistema Cedrux.

Como resultado se obtiene el módulo Banco como un producto configurable, totalmente funcional que cumple con las necesidades del cliente, permite el control de los procesos bancarios y se adapta a las legislaciones vigentes referentes a la actividad financiera en las empresas cubanas, permitiendo así que mejore el funcionamiento en general del módulo Banco perteneciente a Cedrux.

Palabras claves: Cedrux, cuenta bancaria, cobros anticipados, derechos fiscales, gestión bancaria, pagos anticipados, sistemas de gestión empresarial.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

En el afán por incrementar la productividad, rendimiento y eficiencia de las empresas, se han buscado vías para mejorar la gestión económica empresarial. Por esto actualmente existe gran flujo de datos que requieren, debido a su extensión, un alto consumo de tiempo y recursos para su gestión. Como consecuencia, con gran influencia en la toma de decisiones y un alto desarrollo en el campo de la informática y las comunicaciones, han surgido sistemas de Planificación de Recursos Empresariales, ERP por sus siglas en inglés “Enterprise Resource Planning”, en los cuales se integran e informatizan procesos para lograr tener una correcta administración que posibilite maximizar beneficios y minimizar costos. Estos sistemas brindan además una solución práctica e integral a problemas reales que se dan a diario, y mejoran la forma de hacer negocios en la empresa con la unificación de las diferentes áreas de la entidad.

Cuba en particular, en medio de numerosas dificultades y contradicciones que afectan a todas las esferas de la vida social, se encuentra en un intenso proceso de búsqueda de soluciones a los problemas que plantean la necesidad del afianzamiento de las conquistas del socialismo y su perfeccionamiento. Para ello, se emplean sistemas de gestión financiera como VERSAT-Sarasola, RODAS XXI y SISCONT5, pero existe la necesidad de crear un sistema que estandarice la gestión de los procesos en las entidades cubanas y se rija por los requerimientos necesarios al desempeño financiero del país. A partir de las políticas económicas establecidas por el Ministerio de Finanzas y Precios, se desarrolla en la Universidad de las Ciencias Informáticas en coordinación con el Ministerio de Finanzas y Precios, las entidades DESOFT y TEICO Villa Clara, el Sistema Integral de Gestión Cedrux está implementado con tecnologías de Software Libre. Se abarcan áreas como el capital humano, la contabilidad, la logística, costos y procesos y las finanzas.

En el sistema Cedrux, el área financiera contiene, entre otras, la actividad de gestión bancaria. Esta consta de operaciones monetarias en las cuales se pueden cometer errores como movimientos de entrada y salida de dinero sin tener efectivo que las respalden.

Actualmente el módulo Banco presenta dificultades en las actividades de gestión de instrumentos bancarios, sobregiros bancarios, tratamiento de las letras de cambio, tratamiento de cheques y en las conciliaciones bancarias se emplea el método de Saldo Ajustado, el cual es poco usado. Se incluyen además un conjunto de incidencias detectadas que dificultan la usabilidad del mismo; entre ellas se pueden mencionar problemas en la generación de reportes, gestión de las cuentas bancarias, gestión de la configuración, configuración de los talonarios de instrumentos por cuentas

bancarias y los instrumentos asociados a estos talonarios, reconocimientos de cobros y pagos anticipados, reconocimiento de números de contrato y la liquidación de derechos fiscales y de cobro.

Para dar solución a estos inconvenientes se determinaron las nuevas funcionalidades que se le agregarán al sistema para mejorar la usabilidad del mismo.

A partir de la problemática planteada anteriormente, se define como **problema a resolver**: Las funcionalidades informatizadas en el módulo banco del Sistema Integral de Gestión Cedrux presentan deficiencias que limitan el correcto funcionamiento del control financiero en las entidades cubanas.

En este contexto se tiene como **objeto de estudio** sistemas informáticos de gestión bancaria usados en las entidades cubanas, centrando como **campo de acción** el diseño e implementación de los procesos de gestión bancaria dentro de la actividad financiera del Sistema Integral de Gestión Cedrux.

Se plantea como **objetivo general** el diseño e implementación de las funcionalidades necesarias para mejorar el funcionamiento del módulo Banco del Sistema Integral de Gestión Cedrux.

Como **objetivos específicos** para dar solución al objetivo propuesto es necesario:

- Definir el marco teórico de la investigación.
- Diseñar las nuevas funcionalidades del sistema a partir de los artefactos entregados por los analistas.
- Implementar las nuevas funcionalidades del sistema a partir del diseño realizado.
- Validar el diseño y la implementación realizados.

Para dar cumplimiento a los objetivos específicos se trazaron las siguientes **tareas investigativas**:

- Sistematización de los fundamentos teóricos de la investigación a partir de los procesos bancarios de una entidad.
- Caracterización de principales tecnologías y patrones a utilizar para el desarrollo de software del Sistema Integral de Gestión Cedrux.
- Análisis de la arquitectura del sistema y principales estándares definidos para el diseño e implementación del subsistema finanzas.
- Diseño de nuevas funcionalidades para la configuración, conciliación y movimientos de cuentas bancarias.
- Diseño de componentes a partir del análisis y levantamiento de requisitos entregados por los analistas.

- Implementación de la lógica de negocio de los componentes propuestos a partir del diseño elaborado.
- Resolución de no conformidades de la primera versión del módulo banco detectadas por el grupo de calidad.
- Evaluación por el equipo de análisis del comportamiento de las nuevas funcionalidades implementadas.
- Elaboración de pruebas para la evaluación de los componentes.

Se tiene como **idea a defender** que para el correcto funcionamiento del control financiero en las entidades cubanas se requiere del diseño y la implementación de las nuevas funcionalidades del módulo banco de Cedrux.

Para cumplir con las distintas tareas investigativas se emplearon los siguientes **métodos de investigación**:

Métodos Teóricos

- **Histórico – Lógico:** Para sistematizar las tendencias históricas y actuales en el desarrollo de los procesos bancarios y determinar su influencia en el problema actual de la investigación.
- **Analítico:** Para realizar un análisis de la información empleada para la investigación, así como las especificaciones de los diferentes procesos financieros y bancarios centrándolos en la problemática.
- **Sintético:** Para realizar una valoración de las diferentes características de los diferentes procesos financieros y poder obtener la solución centrada en los objetivos de la investigación.
- **Modelación:** Para modelar los nuevos 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:** Para realizar el estudio de los procesos actuales que se desarrollan en el sistema bancario y financiero y sus características y determinar los problemas que presenta la aplicación.
- **Observación:** Para determinar cómo quedaría el nuevo desarrollo y su influencia en la

integración con el actual sistema.

- **Experimento:** Para establecer las diferentes pruebas de calidad con el propósito de determinar la fiabilidad del sistema y el mejoramiento de la usabilidad del mismo, a través de la detección de errores.

A partir de los diferentes métodos de investigación, se espera obtener como **posible resultado** una versión del módulo Banco del Sistema Integral de Gestión Cedrux, que mejore el funcionamiento de la gestión bancaria en las entidades cubanas y cumpla con los nuevos requisitos funcionales descritos.

El trabajo de diploma cuenta con Introducción, Tres Capítulos, Conclusiones, Recomendaciones, Bibliografía y Glosario de Términos.

Capítulo 1: "Fundamentación Teórica". Se menciona el estudio del estado del arte donde se fundamentan teóricamente la gestión de los procesos bancarios. Paralelamente se describe el campo de acción así como una breve descripción de los sistemas actuales que implementan estos procesos. Por último, se justifica el uso de las diferentes tecnologías empleadas para el desarrollo de la aplicación.

Capítulo 2: "Diseño e Implementación". Se realiza un análisis de la arquitectura propuesta así como los artefactos de las nuevas funcionalidades entregados por los analistas. Se diseñan los nuevos componentes, estructuras de datos, se valoran los estándares de codificación establecidos y se exponen las nuevas clases implementadas.

Capítulo 3: "Validación de la Solución". Se evalúan los diseños propuestos a partir de las métricas de diseño y técnicas de prueba para la implementación. Se exponen los criterios de aplicación de las pruebas de calidad, así como un análisis de los métodos aplicados para la evaluación de satisfacción de los clientes. Se mencionan además los avales de implantación y calidad para el futuro despliegue del módulo.

Capítulo I: Fundamentación Teórica

1.1. Introducción.

Con el desarrollo empresarial y el incremento de los procesos que influyen en la competitividad de las empresas se dificulta el control eficiente de las actividades financieras. Es necesario por tanto el mejoramiento de las tecnologías que permitan aumentar la eficiencia de estas actividades.

En el presente capítulo se realiza un análisis de los procesos financieros referentes a la actividad bancaria en las empresas, además de un estudio crítico de diferentes sistemas de gestión que se emplean en el mundo así como en Cuba, centrándose en el Sistema de Gestión Integral Cedrux y las actividades financieras que en él se gestionan. El marco de trabajo y el modelo de desarrollo establecidos por el Departamento de Tecnologías y la dirección del Centro para la Informatización de Gestión de Entidades CEIGE, definen los patrones de diseño y arquitectónicos así como las herramientas y tecnologías empleadas para el diseño y la implementación, de acuerdo con las peculiaridades económicas del país y las tendencias actuales del desarrollo de software.

1.2. Software libre y aplicaciones web.

Las nuevas formas de administración y entrega de servicios para satisfacer las necesidades de los clientes tienen gran influencia en la industria del software, de ahí que las tendencias a mejorar las tecnologías y la forma de desarrollar se desplacen a modelos orientados a servicios, proporcionando flexibilidad y agilidad. En este ámbito, el desarrollo de aplicaciones personalizadas a diferentes sectores de la economía se hace complejo, pues se intentan usar diferentes técnicas y métodos para dar solución a las necesidades de los clientes. La determinación del modelo a emplear para satisfacer las demandas del mercado se hace cada vez más difícil, sin embargo, son adaptables a la solución dependiendo de las necesidades del cliente, las tendencias y las tecnologías.

La utilización de software, tecnologías y herramientas libres, ha garantizado el ahorro de dinero en la adquisición de licencias, la eliminación de barreras presupuestarias y garantiza la independencia tecnológica. El producto final se entrega al cliente con la posibilidad de conocer el código fuente, lo que permite conocer mejor la estructura y agiliza el mantenimiento y corrección del mismo. Existe una independencia de plataforma por lo que no presenta las limitaciones de requisitos no funcionales referentes al Sistema Operativo. Todo esto garantiza que el usuario no dependa del autor del software, otorgando cierta libertad.

Teniendo en cuenta que el mercado se extiende a nivel de internet, el empleo del software libre para el desarrollo de aplicaciones web y el uso de la red como un medio de comunicación, socialización y gestión de la información se ha convertido en uno de los temas más tratados en la actualidad. La querencia de software que permitan satisfacer las necesidades de las empresas a través del uso del internet es una vía rápida para la gestión y oferta de servicios. El desarrollo de aplicaciones web es casi indispensable para las entidades, empresas y organizaciones, porque son multiplataforma e independientes del sistema operativo. Permiten hacer cambios en la lógica del negocio sin afectar al cliente. Presentan una gran adaptabilidad y por lo general, no es necesario consumir recursos en la capacitación del personal que accede a la misma, por el bajo nivel de complejidad que poseen. Además, pueden ser utilizadas en tiendas virtuales, diarios digitales, portales de internet y la gestión de recursos empresariales. (1)

1.3. Gestión de los procesos bancarios en la actividad financiera empresarial.

La gestión de los procesos contables tiene por objetivo la captación, medición, registro, valoración y control de la circulación interna de los valores económicos de la empresa. En los procesos contables se incluye la gestión bancaria, encargada del control de todos los movimientos bancarios de entrada y salida de efectivo en la entidad. Incluye la gestión de las cuentas bancarias, los talonarios de instrumentos asociados a ellas, así como los instrumentos emitidos para realizar cada una de las operaciones de anticipos, reembolsos, liquidación de obligaciones y liquidación de derechos.

La gestión bancaria controla además el flujo del capital contable de la entidad y la contabilización de los procesos a través de los estados de cuentas, las conciliaciones bancarias y cierres de períodos contables.

Los procesos bancarios en una entidad permiten:

- Gestionar las cuentas bancarias por tipos y monedas.
- Gestionar por cada cuenta bancaria los talonarios de los diferentes instrumentos de pagos (cheques, letras de cambio, pagaré, etc.).
- Emitir y registrar instrumentos de pago.
- Procesar los estados de cuentas de banco recibidos teniendo en cuenta las operaciones de cobros y pagos automáticos, así como las registradas con anterioridad por la entidad.
- Gestionar las conciliaciones bancarias por cada una de las cuentas bancarias permitiendo escoger el método a utilizar.

- Emitir reportes por moneda como son: registro de cheques emitidos, registro de cobros, registro de pagos, confirmación de saldos, registro de disponibilidad, etc.
- Realizar cierre diario, de período contable y de ejercicio para cada cuenta bancaria. (2)

1.4. Sistemas informáticos de gestión bancaria empleados en Cuba.

En Cuba se emplean varios sistemas que informatizan los procesos de Gestión Bancaria, algunos de factoría nacional y otros desarrollados por empresas extranjeras. En las entidades nacionales cubanas la implantación de estas tecnologías, que en su mayoría gestionan los procesos centrados en un sector específico de la economía o son implementados sobre otras bases económicas, se usan para la gestión de actividades comunes derivadas de la forma de administración inherente al país.

A continuación se mencionan algunos de los sistemas empleados por las empresas cubanas para la gestión de procesos bancarios.

1.4.1 Sistemas de creación nacional.

SISCONT-5: Sistema desarrollado por la empresa de Tecnologías de la Información, Automática y las Comunicaciones (TECNOMATICA), atendiendo a los requerimientos del Ministerio de la Industria Básica aunque puede ser utilizado en otras entidades nacionales.

SISCONT-5 permite trabajar de forma multiusuario o monousuaria o por internet, dependiendo del software elegido que trabaja sobre el sistema operativo Windows XP o mayor. Es altamente configurable en las opciones de seguridad, en el cual el acceso depende del nivel que presente el usuario autenticado. Es un software contable que emplea la doble moneda y emplea un potente motor de base de datos que le permite gestionar documentos en línea desde cualquier sistema.

Gestión bancaria en SISCONT-5:

SISCONT-5 contiene varios módulos integrados con diversas soluciones para su contabilidad y finanzas, entre ellos el módulo de tesorería que permite controlar y programar las cuentas por pagar, actualizar en línea el flujo de caja y emitir cheques y cuentas contables en forma directa. También ingresa un asiento contable actualizando automáticamente todos los libros de diario, mayor, libro banco, compras y ventas. (3).

Como se puede evidenciar dentro del módulo de tesorería se gestionan varios procesos bancarios necesarios para la adecuada gestión financiera en las entidades, pero según las descripciones no hay evidencias de que además de emitir cheques se puedan recibir cheques a través de este módulo, ni especificar las conciliaciones en los libros bancarios por varios métodos.

RODAS XXI: El Sistema Integral Económico Administrativo RODAS XXI fue desarrollado por la empresa CITMATEL para la gestión los procesos económicos de las entidades presupuestadas cubanas. Es un sistema multiempresa que permite el intercambio automático de los comprobantes generados por cada módulo con el de Contabilidad, trabaja con doble moneda, crea reportes fácilmente, permitiendo su visualización, y de forma opcional, imprimirlos.

RODAS XXI cuenta con varios módulos que pueden emplearse en su totalidad o en cualquier combinación, estos son los de Contabilidad, Activos Fijos, Nóminas, Inventario, Facturación y Finanzas.

Gestión bancaria en RODAS XXI:

En el módulo Finanzas se puede realizar el registro de cheques tanto emitidos, como recibidos, llevar el registro de otros instrumentos de pago y efectuar las operaciones de cobros y de pagos. Además, permite tener un control de dietas, reembolsos, vales para pagos menores y el registro de ingresos, llevar los submayores bancarios, realizar la conciliación bancaria y tener todo el control de caja con la posibilidad de realizar el arqueo correspondiente. Incluye la generación de reportes y listados de cheques e instrumentos de pago y registros de las operaciones.

En el módulo Finanzas se gestionan todos los procesos bancarios necesarios según el objeto de estudio definido aunque no hay evidencias de que se gestionen las conciliaciones en los libros bancarios por varios métodos. Aunque el sistema satisface la gestión de los procesos bancarios según la necesidad del objeto de estudio tiene como inconveniente que no se integra con Cedrux porque emplea como gestor de base de datos el Microsoft SQL Server por lo que solo funciona sobre plataformas Windows y en términos legales se deben incurrir en gastos por conceptos de pago de licencias por el uso de estas tecnologías propietarias.

VERSAT-Sarasola: Fue desarrollado por la Empresa de Información y el Conocimiento del Ministerio del Azúcar, Casa del Software de Villa Clara (UEB TEICO), para automatizar la actividad económica, contable y financiera del Ministerio del Azúcar (MINAZ) en el año 1998. Se distingue por ser el primer sistema de contabilidad cubano certificado, según las nuevas normativas establecidas por el Ministerio de Finanzas y Precios y el Ministerio de la Informática y las Comunicaciones para este tipo de software.

VERSAT es una herramienta para la planificación económica, el control y el análisis de gestión, está diseñado para ser empleado en cualquier tipo de entidad empresarial o presupuestada. Permite llevar el control y registro contable individual de todos los hechos económicos que se

originan en las estructuras internas de las entidades, así como exponer el estado financiero y toda la información económica y contable. Logra establecer un proceso de interacción usuario-sistema y presenta rapidez y fiabilidad, a partir de la configuración del proceso de contabilización de los documentos primarios y de las propias posibilidades de trabajo contenidas en cada subsistema.

Es un sistema económico integrado que presenta 10 módulos que incluyen configuración y seguridad, contabilidad general y de gastos, costos y procesos, finanzas, caja y banco.

Gestión bancaria en VERSAT-Sarasola:

El módulo de finanzas, caja y banco es de los más abarcadores dentro del VERSAT, pues dentro de las 5 actividades que posee, están recogidas casi la totalidad de las operaciones financieras que pueden generarse en cualquier entidad. Una vez instalada esta aplicación, puede utilizarse una herramienta que se ha elaborado para incorporar una configuración predeterminada que incluye todos los tipos de documentos que existen en el país, los tipos de conceptos de contrapartidas más usados y en qué actividades se utilizan, más otros aspectos necesarios a configurar previo a la explotación del software. (4).

Aunque el sistema satisface la gestión de los procesos bancarios según la necesidad del objeto de estudio tiene como inconveniente que al estar implementado en el lenguaje Delphi y usar como gestor de base de datos SQL Server solo funciona sobre plataformas Windows y en términos legales se deben incurrir en gastos por conceptos de pago de licencias por el uso de estas tecnologías propietarias.

Cedrux: El Sistema Integral de Gestión Cedrux es desarrollado por la Universidad de las Ciencias Informáticas en coordinación con el Ministerio de Finanzas y Precios, las entidades DESOFT y la UEB TEICO Villa Clara. Es un sistema integral que gestiona las áreas de capital humano, contabilidad, logística, costos y procesos y finanzas.

En el área de las finanzas y la tesorería se gestionan los procesos de Cobros y Pagos, Caja y Banco.

Gestión bancaria en Cedrux:

El módulo banco actualmente lleva el control de los Instrumentos Bancarios que se generan de las operaciones de Anticipos, Reembolsos y Liquidaciones. Se realiza además el balance de Carga Inicial, la gestión de Cuentas Bancarias y su configuración, el control de los Estados de Cuentas y Recuperaciones.

El módulo de Banco del sistema Cedrux satisface las necesidades del objeto de estudio, gestionando parcialmente los procesos bancarios. En la gestión de los procesos bancarios existen

actualmente deficiencias como que solamente se realizan movimientos de salida en los instrumentos, las conciliaciones de cuentas se realizan por un único método de tres métodos posibles, no se realizan cancelaciones de operaciones antiguas, así como no se gestiona correctamente los talonarios para números de instrumentos para cheques a emitir y recibir.
Para corregir estas deficiencias es necesario implementar las nuevas funcionalidades que cubran las deficiencias detectadas.

1.4.2 Sistemas extranjeros.

ASSETS-NS: Fue distribuido en Cuba en el año 1997 por INFOMASTER, entidad informática perteneciente a la Empresa Nacional de Producción y Servicios a la Educación Superior del MES. Es un Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de compras, ventas, producción, taller, inventario, finanzas, contabilidad, presupuesto, activos fijos, útiles y herramientas y recursos humanos.
ASSETS mejora los procesos de administración y planificación de inventarios, es un sistema flexible, amigable, funciona en ambiente multiusuario y estaciones remotas. Presenta opciones de seguridad a partir del perfil del usuario que se encuentre registrado.
Gestión bancaria en ASSETS-NS:
Genera automáticamente los asientos de diario a la contabilidad por cada una de las transacciones contempladas en el sistema, generación automática del catálogo de cuentas, procesos de caja, conciliación entre módulos, movimientos masivos de activos fijos, aperturas de módulos, conciliación bancaria, registro y anulación de comprobantes de operaciones por lotes, visualización de períodos contables abiertos y el estado de cierre del resto de los módulos, reporte de conciliación bancaria, gestión de instrumentos bancarios, estados de cierre, y la generación de reportes (5).
Como se puede evidenciar en el sistema se gestionan varios procesos bancarios necesarios para la adecuada gestión financiera en las entidades (conciliación bancaria, reporte de conciliación bancaria, gestión de instrumentos bancarios), pero según las descripciones no hay evidencias de que además de emitir instrumentos se puedan recibir también, ni especificar las conciliaciones en los libros bancarios por varios métodos.
SAP ERP: Fue desarrollado por la multinacional del software SAP (Sistemas, Aplicaciones y Productos), creada en 1972 en Alemania.
SAP ERP posee una plataforma de tecnología abierta que puede aprovechar e integrar diversos sistemas, tanto de SAP como de terceros. Proporciona funcionalidad de software end-to-end para el

manejo de la empresa, además de soporte para la gestión de sistemas. El software ERP de SAP comprende cuatro soluciones independientes que brindan soporte a procesos de negocio a través de su sistema ERP específico: SAP ERP Financials, SAP ERP Human Capital Management, SAP ERP Operations y SAP ERP Corporate Services.

Gestión bancaria en SAP ERP:

SAP ERP Financials le ofrece control e integración de toda la información financiera y empresarial esencial para la toma de decisiones estratégicas y operativas. Mejora la gestión de los controles internos, simplifica el análisis financiero y permite llevar a cabo procesos de negocio adaptables. Esto le permite transformar las finanzas de una simple función administrativa en una parte estratégica de su empresa centrada en el valor.

La aplicación de SAP para bancos, ofrece un entorno sólido para manejar todos los aspectos de organización, brindando una variedad de beneficios empresariales. La gestión bancaria necesita controlar los costos, consolidar las operaciones y el negocio, y administrar las operaciones globales al mismo tiempo que se adaptan a las nuevas tecnologías. (6).

No obstante, el beneficio de SAP ERP es intangible, ya que se tiene que valorar cuanto le cuesta a la empresa tener o no la información. Además, su infraestructura puede ser muy alta si no se cumple desde un principio con los requerimientos del SAP. Se necesita una persona al 100% con el conocimiento de todo el negocio para que pueda hacerse cargo de administrar el SAP y dar solución a los problemas de forma rápida. Requiere de mantenimiento, actualización y fuertes inversiones en módulos complementarios. Es muy costoso, muchas empresas no pueden pagar los precios aunque necesitan de las prestaciones de un ERP. (7).

Openbravo: Sistema de planificación de recursos empresariales (ERP) desarrollado por la compañía Tecnicia (actualmente conocida como Openbravo), que está preparado para implantarse en entornos multinacionales y multicliente.

Openbravo ofrece un sistema ERP totalmente integrado, adaptado a las necesidades de cada empresa, independientemente de su tamaño o su sector de actividad. Brinda una amplia gama de funcionalidades que abarcan los procesos de negocio, tales como: gestión de las relaciones con el cliente, facturación, aprovisionamiento, inventario, gestión de proyectos, gestión económica financiera e inteligencia de negocio. También ayuda a las empresas a administrar sus operaciones diarias, optimizar los procesos de negocios, lograr mayor satisfacción del cliente e incrementar la rentabilidad empresarial.

Gestión bancaria en Openbravo:

Dentro de la gestión económico-financiera Openbravo gestiona el plan de cuentas, las cuentas contables, los impuestos, las cuentas a pagar, las cuentas a cobrar y la contabilidad bancaria; incluye la realización de liquidaciones y emisión de informes de banco. (8).

Aunque el sistema satisface la gestión de los procesos bancarios según la necesidad del objeto de estudio no hay evidencias de la gestión de conciliaciones bancarias por varios métodos, elemento muy importante a tener en cuenta en las entidades cubanas de las Fuerzas Armadas, y aunque es distribuido bajo licencia de software libre, al estar implementado con la tecnología J2EE se hace difícil el acceso y se violan términos legales relacionados con el bloqueo. Este elemento se evidencia en el Informe de Cuba sobre la resolución 62/3 de la Asamblea de la ONU: "(…) el sector de la informática y las comunicaciones se limita el acceso a las tecnologías de punta (…) el acceso a las nuevas versiones del motor de base de datos en software libre para prestaciones medias de mayor difusión en el mundo, MySQL se mantuvo limitado, como también sucede con Java, al ser adquirido este producto por la firma norteamericana Sun Microsystems. Este sistema que se descargaba gratuitamente en Internet desde Cuba, era ampliamente usado en el país en una gran variedad de aplicaciones (…)". (9)

Análisis de los sistemas estudiados.

A partir del estudio de los sistemas de gestión que se encuentran implantados en las entidades del país y que de cierta manera incluyen la gestión de los procesos bancarios, se llega a la conclusión de que no existe un sistema que integre todos los procesos de las empresas y que a su vez gestione de manera eficiente y estándar las actividades bancarias.

El empleo de sistemas de carácter extranjero como SAP ERP generan al país gran cantidad de gastos por pago de licencias y mantenimiento, que requieren trabajos de adaptación al sistema cubano, porque están basados en sistemas económicos extranjeros. Esto genera dependencia tecnológica porque al ser software privativo no permiten la modificación del código interno.

En el caso de las aplicaciones nacionales, son creadas para sectores específicos de la economía, por lo que su uso no se puede generalizar a todas las entidades, estas no gestionan los procesos de forma estándar, aunque es válido resaltar que se le han hecho modificaciones para emplearlos en otras entidades, pero generan consumo de recursos en la adaptación. Se puede señalar por ejemplo el RODAS XXI, creado para gestionar los procesos del CITMA y entidades presupuestadas y en el caso del VERSAT-Sarasola, creado para el Ministerio de la Industria Azucarera y algunas entidades productoras, demostrando que en estas se gestionan los procesos de manera diferente.

De ahí la necesidad de informatizar los nuevos requerimientos del sistema, referentes a la gestión bancaria en las entidades, regidos por las políticas económicas del país. Con el desarrollo y la implantación del módulo Banco del Sistema Integral de Gestión Cedrux y la sustitución de los sistemas actuales que se emplean en el país, se solucionarán los problemas de estandarización y adaptabilidad en la gestión de los procesos bancarios, se eliminará el necesario pago de licencias en los sistemas de factoría internacional, así como su continuo mantenimiento generado por las constantes actualizaciones realizadas para adaptarse al proceso de gestión que se lleva en cada una de las entidades que los emplean, erradicando así la dependencia tecnológica. Esto ahorrará gran cantidad de recursos y dinero a la economía nacional y aumentará la rentabilidad de las empresas.

1.5 Modelo de desarrollo.

Para la implementación del módulo se emplearon las pautas arquitectónicas y el modelo de desarrollo establecido por el centro CEIGE que proponen un modelo estandarizado, así como la definición clara de cada uno de los roles involucrados en el proceso de desarrollo de la solución.

Con el objetivo de producir software de alta calidad y lograr la salida de un producto eficiente que cumpla con los requisitos funcionales y se adapte a las necesidades reales del sistema económico cubano, el modelo de desarrollo usado presenta las siguientes características:

- **Centrado en la arquitectura:** La arquitectura guía los casos de uso y determina la línea base y los elementos del software. Debe estar estructurada para dar soporte a los requerimientos del sistema y admitir nuevos cambios cada vez que se requiera. Diseña y controla la integración de los componentes, dando un nivel de independencia entre ellos para su implementación individual y acoplamiento posterior, además de determinar las prioridades en el desarrollo.
- **Orientado a componentes:** Las iteraciones son realizadas dado el peso arquitectónico de los componentes en los cuales se descompone el sistema. Estos representan una abstracción de los procesos de la lógica del negocio y poseen interfaces que permiten la fácil integración y comunicación entre ellos.
- **Iterativo e incremental:** El equipo de arquitectura, la dirección del proyecto y los clientes, determinan las iteraciones así como los objetivos a cumplir por cada una de ellas. En cada iteración se determinan los riesgos, los componentes a desarrollar y su integración con el sistema, permitiendo el desarrollo incremental del producto.

- **Ágil y adaptable al cambio:** El desarrollo de las partes formaliza las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados. (10)

1.6. Patrones de diseño y arquitectura.

Para desarrollar aplicaciones de buena calidad es necesario establecer patrones que permitan agilizar el proceso a través de la reutilización y la aplicación de esquemas de solución que se apliquen a diferentes tipos de problemas.

1.6.1 Patrones de diseño.

Los patrones de diseño son descripciones de clases cuyas instancias colaboran entre sí. Cada patrón es adecuado para ser adaptado a un cierto tipo de problema. Representa un esquema o microarquitectura que supone una solución a problemas (dominios de aplicación) semejantes; una estructura común que tienen aplicaciones semejantes. (11).

Los patrones de diseño brindan una solución generalmente ya probada y documentada a problemas que se dan durante el proceso de desarrollo de software. Emplean un conjunto de buenas prácticas que facilitan el trabajo, definen una estructura de clases que da respuesta a uno o varios problemas en particular y presentan la ventaja de que son fáciles de comprender, además de que no dependen del lenguaje, haciéndolos genéricos. Lo complejo es cuando se tiene que decidir cuál usar, pues presentan diferentes soluciones, ya sea a través del empleo de uno u otro, o la combinación de varios. De ahí la importancia de conocer y estudiar los diferentes patrones que existen para poder determinar su uso.

A continuación se explican los patrones de diseño seleccionados para su aplicación en el módulo.

Patrones GRASP: Patrones que asignan responsabilidades a través de la descripción de los principios fundamentales del diseño y la solución a un problema. Entre ellos existen 5 patrones fundamentales:

- **Experto:** Asigna responsabilidades a la clase que se encarga de gestionar la información referente a un objeto determinado y es capaz de cumplir con la actividad asignada conservando el encapsulamiento y promoviendo clases sencillas fáciles de comprender y mantener.

- **Creador:** Tiene la responsabilidad de asignar la creación de una clase a otra. Esta asignación solo puede realizarse cuando agrega o contiene una instancia de ella, contiene instancias de sus objetos y contiene los datos de inicialización que serán transmitidos a la nueva clase.
- **Controlador:** Gestiona los eventos de entrada generado por actores externos, asignando responsabilidades a otras clases de manera independiente que permitan el bajo acoplamiento y la alta cohesión.
- **Bajo acoplamiento:** Es la medida en que cada una de las clases realiza actividades independientes, además de poseer un conocimiento de las actividades que realizan las otras clases del sistema permitiendo la reutilización.
- **Alta cohesión:** La cohesión es la relación que existe entre las clases y la medida en que éstas realizan labores únicas dentro del sistema, pero que están estrechamente relacionadas entre sí.

Patrones Gang of Four (GoF): Patrones de diseño que definen una estructura de clases que tratan una situación en particular.

- **Fachada:** Proporciona una interfaz única que simplifica los servicios generales del sistema, define un comportamiento independientemente al lugar de uso, siendo esta más fácil de utilizar. (12)

1.6.2 Patrones de arquitectura.

La Arquitectura de Software es la organización fundamental de un sistema enmarcada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución. (13)

A continuación se describen los patrones arquitectónicos definidos para la estructura del módulo.

- **Arquitectura orientada a componentes:** un componente de software es una unidad de composición con interfaces especificadas contractualmente y dependencias explícitas de contexto. Un componente de software puede ser desplegado de manera independiente y es objeto de composición por terceras partes. (14)

  Un componente es una unidad independiente dentro de un sistema de software. Satisface funcionalidades específicas con interfaces bien definidas que puede ser ensamblado con otros componentes a través de una interfaz.

Todas las funcionalidades contenidas en al menos un componente, y las distintas interacciones entre estos, originan funcionalmente la presencia de módulos y subsistemas.

- **Patrón MVC:** Patrón de arquitectura en el que todo proceso está distribuido en tres componentes de aplicación que separan la interfaz de usuario, el control de la lógica del negocio y el acceso a los datos en componentes distintos:
- **La Vista:** Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación web, la vista es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

  La vista se encarga de interactuar con el usuario, presentando los datos de forma visual a través de una interfaz gráfica interpretada por un navegador web.
- **El Controlador:** Es el encargado de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el modelo en caso de ser necesario.

  El controlador se encargada de recibir las peticiones del usuario enviadas por la vista y enviárselas al modelo para recibir respuesta de este y transmitírselas a la vista.
- **El Modelo:** Es la representación de la información que maneja la aplicación. El modelo en si son los datos puros que puesto en el contexto del sistema, proveen de información al usuario o a la aplicación misma.

  El modelo incorpora el dominio de la aplicación así como el acceso y la transformación de los datos persistentes, ya sean bases de datos, archivos XML, texto, registros, entre otros. (15)

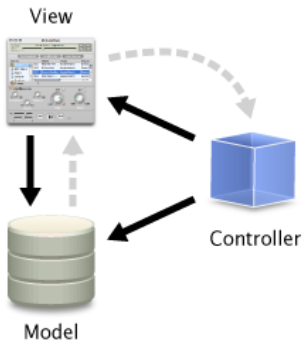

Figura 1: Modelo Vista-Controlador.

1.7. Lenguaje de modelado.

UML (Unified Modeling Language) 2.1: El Lenguaje Unificado de Modelado prescribe no solo la estructura de la aplicación, el comportamiento y la arquitectura, sino también procesos de negocio y estructura de datos. UML, también proporciona una base para el modelo que unifica las etapas de

desarrollo e integración de modelos de negocio, a través de modelos arquitectónicos y la aplicación, para el desarrollo, implementación, mantenimiento y la evolución. (16)

El Lenguaje Unificado de Modelado es un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, presenta una vista esencial de la estructura del proyecto, con lo que se visualiza, especifica, construye y documenta el software.

1.8. Lenguajes de programación.

Un Lenguaje de Programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora. Cada lenguaje posee sus propias sintaxis. También se puede decir que un programa es un conjunto de órdenes o instrucciones que resuelven un problema específico basado en un Lenguaje de Programación. (17)

Un lenguaje de programación es un idioma artificial diseñado para que las computadoras puedan interpretar las órdenes dadas por el hombre. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones.

1.8.1 En el servidor.

Los lenguajes de programación en el servidor se interpretan y ejecutan en el propio servidor web y son los encargados de recibir las peticiones generadas por los usuarios a través de las páginas clientes, y enviarles una respuesta a su petición. Realizan operaciones de acceso a bases de datos, acceso a redes, lógica de negocio, entre otras.

- **PHP 5.2.6:** Es un lenguaje de programación interpretado, diseñado para la creación de páginas web dinámicas. Es gratuito, independiente de la plataforma y puede ser usado en cualquier sistema operativo. Posee una gran librería de funciones y documentación. Es de fácil aprendizaje debido a su gran semejanza a otros lenguajes. Presenta alta compatibilidad con los gestores de bases de datos más comunes como PostgreSQL, Oracle y MySQL, entre otros.

1.8.2 En el cliente.

Los lenguajes de programación en el cliente son ejecutados e interpretados en un navegador web. Son independientes de la plataforma y no requieren de un sistema operativo en específico. Se encargan de realizar las peticiones al servidor a través de las páginas clientes.

- **HTML:** El Lenguaje de Marcado de Hipertexto es el lenguaje que se utiliza para la creación de páginas web. Se compone por una serie de comandos que son interpretados por el navegador. El navegador ejecuta todas las órdenes contenidas en el código HTML, de forma que puede estar capacitado para brindar prestaciones. (18)

  HTML se usa para describir la estructura y el contenido en forma de texto. Se describe en forma de etiquetas. Puede incluir scripts, los cuales pueden afectar el comportamiento del navegador web. Puede incluir el tratamiento de imágenes y multimedia para acompañar el texto.

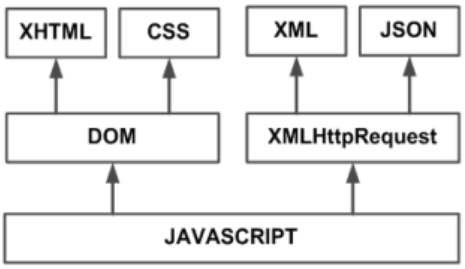

Figura 2: Tecnologías bajo el concepto de AJAX.

1.9. Otras tecnologías utilizadas

Para el desarrollo del producto se emplearon otras tecnologías, las cuales se describen a continuación:

- **XML:** Lenguaje de Etiquetado Extensible muy simple, pero estricto que desempeña un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a HTML pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

  El Lenguaje Extensible de Etiquetado es un metalenguaje extensible de etiquetas, desarrollado por la World Wide Web Consortium (W3C), permite definir la gramática de lenguajes específicos, se propone como un estándar para el intercambio de

información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y otros. (19)

- **AJAX:** El término AJAX es un acrónimo de *Asynchronous JavaScript + XML*, que se puede traducir como "JavaScript asíncrono + XML". AJAX es la combinación de varias tecnologías independientes.

  Las tecnologías que forman AJAX son XHTML y CSS, para crear una presentación basada en estándares; DOM, para la interacción y manipulación dinámica de la presentación; XML, XSLT y JSON, para el intercambio y la manipulación de información; XMLHttpRequest, para el intercambio asíncrono de información; JavaScript, para unir todas las demás tecnologías. (20).

1.10. Frameworks.

Los frameworks son estructuras de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que se le puede añadir las últimas piezas para construir una aplicación concreta. (21)

Los frameworks son diseñados para mejorar el desarrollo de software. Son el conjunto de procesos y tecnologías en el cual varios objetos son integrados para dar solución a un problema complejo, en él se estandarizan conceptos, prácticas y criterios para resolver un tipo de problemática en particular. Los frameworks permiten acelerar el proceso, reutilizar código ya existente y promover las buenas prácticas en el uso de patrones.

- **Sauxe 2.0:** Sauxe es un Marco de Trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. (22)

  Sauxe utiliza en la capa de acceso a datos el Lenguaje de Consulta de Datos (DQL) que implementa Doctrine. La documentación de éste tiene todas las características necesarias para ser funcional en casi cualquier proyecto. (22)

  Doctrine ORM para PHP 5.2.3 y posterior permite convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos, y el utilizado en una base de datos relacional**.** (23)

Sauxe utiliza ExtJs en la capa de presentación, por la gran gama de componentes que se pueden reutilizar para agilizar el proceso de desarrollo y mostrarle al usuario una interfaz más amigable y funcional. (22)

ExtJs es una librería JavaScript que permite construir aplicaciones complejas. Incluye componentes UI (User Interface) de alto rendimiento y personalizables, modelo de componentes extensibles, una API fácil de usar y licencias open source y comerciales.

Es soportado por varios navegadores web como Internet Explorer, Firefox, Safari y Opera. (24)

Sauxe es el resultado de una importante mezcla entre: la extensión de algunos componentes de un potente Marco de Trabajo, un sistema ORM y una muy popular librería, dando lugar a un novedoso Marco de Trabajo, con funcionalidades y prestaciones específicas pero también fácilmente extensibles a cualquier aplicación web de gestión. (25)

Sauxe emplea 8 componentes de ZendFramework que conforman un potente y extensible Marco de Trabajo de aplicaciones web. A partir de la extensión de algunos componentes de ZendFramework surge ZendExt, desarrollado por el Departamento de Tecnología y el UCID, con el objetivo de crear un Marco de Trabajo extensible y configurable centrando el desarrollo de las aplicaciones, en la lógica del negocio, en las interfaces de usuario, alejando a los programadores de los detalles arquitectónicos, con soporte para entornos multientidad y para una arquitectura de sistema orientada a componentes. (22)

1.11 Estructura del Framework Sauce.

El marco de trabajo definido está enfocado en realizar una distribución acorde a los subsistemas, módulos y componentes diseñados, así como la complejidad de los mimos y el flujo de información que en ellos se gestiona, con lo que se logra el desarrollo de una arquitectura web.

La estructura física del sistema se realizó mediante el empaquetamiento lógico de funcionalidades.

El sistema, dentro de su carpeta raíz cuenta con las carpetas ***apps*** y ***web,*** que contienen respectivamente la lógica del negocio e interfaces de los módulos referentes a cada uno de los subsistemas del proyecto. En cada una de ellas se encuentra una carpeta por subsistema y específicamente dentro del subsistema ***Finanzas*** se encuentra el módulo ***Banco*** y sus componentes***.***

La especificación de las configuraciones y los recursos asociados a la aplicación así como los ficheros xml de cada uno de los componentes se encuentran situados en el paquete ***común***, ubicado en la carpeta apps al mismo nivel que los demás subsistemas. La carpeta ***común*** dentro

de Finanzas contiene los recursos comunes al subsistema, y para el desarrollo de los componentes instrumento, talonario y notificaciones, está la carpeta ***comunfinanzas.***

Como se describe a continuación, el módulo Banco está estructurado con una carpeta para cada componente:

- **Controlles:** Contiene las clases controladoras correspondientes al componente. Está encargada de gestionar las funcionalidades del componente y el flujo de información entre las vistas y las modelos. Comunica las interfaces de usuario con la lógica del negocio.
- **Models:** Contiene la programación de toda la lógica del negocio del componente. En las carpetas ***bussines*** y ***domain*** se realiza el control y acceso a la información física de la base de datos.

  En la carpeta ***bussines*** están contenidas las clases modelo, en las que se programa la lógica del negocio del componente y las acciones de modificación a los datos persistentes como insertar, actualizar y eliminar.

  En la carpeta ***domain*** se guardan los archivos generados por el framework Doctrine. Incluye las clases encargadas del acceso por consultas a la base de datos. Estas clases heredan de clases contenidas dentro de la carpeta ***generated*** que son las encargadas de mapear del modelo de objetos al modelo relacional.
- **Services:** Contiene las funcionalidades que brinda el componente a los componentes externos que las soliciten, con lo que controla el flujo de información y la integración entre componentes.
- **Validators:** Contiene las clases que realizan las validaciones del componente, como las precondiciones a tener en cuenta para la ejecución de algún método. Se implementan aparte para separarlas de la lógica del negocio y se ejecutan a partir del mecanismo AOP (programación orientada a aspectos) del framework, con lo que se garantiza configurar las validaciones que se deseen en cada acción.
- **Views:** Contiene los ficheros que gestionan la capa de presentación, agrupados en las carpetas ***idioma*** y ***scripts***

  La carpeta ***idioma*** contiene ficheros de tipo json que recopilan etiquetas para la gestión de los mensajes vinculados a la presentación. La carpeta ***scripts*** contiene todas las vistas, para lo que se crea una carpeta para cada clase controladora y dentro se incluye la vista o script, archivos de extensión phtml que especifican el título de la página que se gestiona y se carga el archivo js que mostrará la presentación.

Ver la distribución del marco de trabajo en el Anexo 1 de la versión digital del documento.

1.12. Herramientas y tecnologías de desarrollo.

Las herramientas y tecnologías de desarrollo son aquellas que se encargan de realizar toda la gestión, y desarrollo del ciclo de vida del software. Este proceso incluye desde la etapa de planificación del proyecto, pasa por la captura de requisitos y llega hasta la etapa de pruebas y despliegue de la aplicación.

A continuación se describen las herramientas y tecnologías de desarrollo definidas para la elaboración del sistema.

1.12.1 Herramientas CASE.

La Ingeniería de Software Asistida por Computación, CASE por sus siglas en inglés (Computer Aided Software Engineering), define la aplicación de técnicas y métodos por medios de programas, métodos y documentación para comprender y explotar las capacidades de los ordenadores.

Las herramientas CASE son las que se encargan de apoyar o automatizar todo el proceso del ciclo de vida de un proyecto de software, siendo imprescindible su uso para la organización y manejo de la información del mismo. Permiten a los administradores del proyecto llevarlo a cabo de forma eficaz y eficiente. (26)

- **Visual Paradigm 6.4:** Visual Paradigm for UML (VP-UML) es una herramienta de diseño UML y herramienta CASE UML diseñada para la ayuda al desarrollo de software. VP-UML soporta los principales estándares de la industria tales como Lenguaje de Modelado Unificado (UML), SysML, BPMN y XM. Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para los requisitos de la captura, software de planificación, la planificación de controles, el modelado de clases y el modelado de datos, modelado de objetos, modelado de componentes, diagrama de despliegue, diagrama de casos de uso y diagrama de actividades.

  Entre las ventajas que presenta se encuentran la navegación intuitiva entre el código y el modelo, demanda en tiempo real, modelo incremental de viaje redondo y sincronización de código fuente, soporte a varios lenguajes en generación de código e ingeniería inversa a través de plataformas Java, como C++, CORBA IDL, PHP, XML y Python. (27)

1.12.2 Herramientas de desarrollo colaborativo.

Las herramientas de desarrollo colaborativo son un conjunto de componentes que se encargan de llevar un control de versiones de código y presentan elementos básicos como el correo y herramientas de mensajería instantánea que en su conjunto atienden a las necesidades de los integrantes como grupos de desarrollo que pueden o no estar ubicados físicamente en la misma área, es decir, grupos de trabajo compuestos por integrantes trabajando a distancia o en forma remota. (28).

- **Control de versiones:** El control de versiones tiene la capacidad de recordar todos los cambios que se hacen tanto en la estructura de directorios como en el contenido de los ficheros. Cuando más de una persona trabaja con los mismos archivos, e inclusive cuando es una sola persona, mantiene cierto control sobre los cambios que se realizan determinando el quién, cuándo y qué. Permite además, tomar decisiones sobre la forma final de un archivo cuando los cambios son realizados por dos personas y facilita recuperar versiones anteriores cuando es necesario por deficiencias detectadas en la última versión. (29)

  Es el empleo de métodos y herramientas para controlar los cambios que se realicen sobre los elementos de un producto o la configuración del mismo. Los sistemas de control de versiones facilitan la administración de las diferentes versiones. El principal objetivo es editar de forma colaborativa y compartir información, además de ayudar en la comunicación entre los desarrolladores, manejo de los lanzamientos, administración de fallos, estabilidad entre el código y los esfuerzos de desarrollo experimental y atribución y autorización en los cambios de los desarrolladores, mejorando considerablemente la dirección del proyecto.
- **Subversion TortoiseSVN 1.4.5:** TortoiseSVN es una herramienta SMC (software de control de código fuente de Microsoft Windows). Está implementado como una extensión de shell de Windows, lo que hace que la integran perfectamente en el explorador de Windows. (30)

  Es un cliente gratuito de código abierto para el sistema de control de versiones Subversión. Maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un repositorio central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio. (31)

1.12.3 IDE de programación.

Los Entornos de Desarrollo Integrado, IDE por sus siglas en inglés (Integrated Development Environment), son un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (32)

Ele IDE seleccionados para el trabajo con los lenguajes de programación para la realización del proyecto es:

- **Netbeans 6.9:** Entorno de desarrollo integrado (IDE por sus siglas en inglés), modular, de base estándar, escrito en el lenguaje de programación Java. Se puede instalar en Windows, MacOS, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicación, las cuales pueden ser usadas como una estructura de soporte general para compilar cualquier tipo de aplicación. Con este IDE se pueden crear aplicaciones que sean soportadas por la plataforma Java como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails, Groovy and Grails y C/C++. (33)

1.12.4 Servidor de aplicaciones.

Un servidor de aplicaciones web es un software que provee la infraestructura necesaria para las aplicaciones web. (34)

Un servidor de aplicaciones web gestiona el procesamiento de páginas que contienen scripts o etiquetas.

- **Apache 2.0:** Servidor web de tecnologías open source que presenta gran robustez, configurabilidad y estabilidad más usado en internet. Es un servidor multiplataforma y corre sobre múltiples sistemas operativos. Presenta entre otras características como base de datos de autenticación y el negociado de contenido.

  Apache es un servidor de diseño modular que trabaja con gran cantidad de lenguajes de script como Perl, PHP y otros, teniendo todo el soporte que se necesita para páginas dinámicas. (35)

1.12.5 Servidor de base de datos.

Es un software que controla la organización, almacenamiento, recuperación, seguridad e integridad de los datos en una base de datos. Acepta pedidos de datos desde un programa de aplicación y le ordena al sistema operativo transferir los datos apropiados. (36)

Un servidor de base de datos es un ordenador encargado de gestionar los datos, ya sea de manera local o a través de la red, implementado sobre el patrón arquitectónico cliente-servidor. Puede estar conectado a otros servidores de bases de datos y realizar la gestión conjunta de los datos o puede también estar conectado a servidores de servicios.

- **PostgreSQL 8.3:** Sistema de gestión de bases de datos de objetos relacionales que utiliza modelo cliente servidor y multiprocesos en vez de multihilos, lo que propicia que el fallo en uno de sus procesos no afectará el sistema por lo que seguirá funcionando.
  Entre sus características se encuentran la integridad referencial, replicación asincrónica, copias de seguridad en caliente, juego de caracteres internacionales, accesos encriptados, múltiple documentación y es multiplataforma. (37).
  Es un producto open source sin pago de licencias, que le reporta a las entidades el ahorro de costos significativos pues tienen libertad de usarlo, modificarlo y distribuirlos en productos comerciales y no comerciales.

1.12.6 Navegador web.

Un navegador web es el instrumento que permite a los usuarios de internet navegar o surfear entre las distintas páginas de sus sitios web preferidos. Se trata de un software que posee una interfaz gráfica compuesta básicamente de botones de navegación, una barra de dirección, una barra de estado (generalmente, en la parte inferior de la ventana) y la mayor parte, en el centro, que sirve para mostrar las páginas web a las que se accede. (38)

Los navegadores web son aplicaciones que interpretan órdenes recibidas en código HTML y las convierten en páginas web. Permiten el envío de información solicitada por el usuario a los servidores web e interpretan las respuestas para mostrárselas al usuario.

- **Mozilla Firefox 2.0.17:** Navegador web de código abierto, considerado uno de los más usados en la actualidad. Es multiplataforma y está disponible en varias versiones: Microsoft Windows, Mac OS X y GNU/Linux. Incluye navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas, navegación privada, navegación con georreferencia y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario.

1.13. Pruebas de software.

Las pruebas de software son los procesos que permiten verificar y revelar la calidad de un producto de software. Se emplean para determinar los posibles errores que puedan existir, tales como fallos de implementación, calidad, cumplimiento de requisitos funcionales o usabilidad. Las pruebas deben estar incluidas en todo el ciclo de vida del proyecto, aunque su aplicación está determinada por el tipo de metodología que se emplee y el nivel de complejidad y profundidad que contenga el proyecto. Así mismo las pruebas no aseguran la ausencia de errores, solo pueden demostrar que existen defectos en el software.

1.13.1 Prueba de caja negra.

Las pruebas de caja negra se encargan de realizar pruebas de funcionalidad al software verificando que las funciones del mismo son operativas. Se realizan sobre la interfaz sin comprobar la estructura interna del componente que se prueba.

En este tipo de prueba se pretende demostrar que:

- Dado una entrada se tiene como resultado la salida esperada.
- Las funciones del software son operativas.
- La entrada se acepta de forma correcta.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

Los posibles errores que se pueden detectar son:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación. (39)

1.14. Conclusiones del capítulo.

Luego de analizar los diferentes procesos bancarios y los sistemas empleados en Cuba para su gestión, se llegó a la conclusión de que estos sistemas presentan deficiencias que limitan el control adecuado y estandarizado de la gestión bancaria. Existe, por tanto, la necesidad del diseño y la implementación de la nueva versión del módulo Banco del Sistema Integral del Gestión Cedrux.

Se realizó la descripción de las herramientas y tecnologías, establecidas por la Línea de Arquitectura y la dirección del proyecto, para el desarrollo del sistema. El uso de tecnologías de software libre y el empleo de herramientas de esta índole brindan rapidez y evitan el pago de licencias por el empleo de tecnologías que requieren de un mantenimiento constante, lo cual acarrearía un alto costo en el pago de ellas. Finalmente, se analizaron los tipos de pruebas establecidos para comprobar la calidad y usabilidad del módulo.

Capítulo II: Diseño e Implementación de la Solución.

2.1 Introducción.

Para informatizar los procesos bancarios en las entidades del país es necesario cumplir con las legislaciones establecidas referentes a la gestión de actividades económicas, de manera que exista estandarización, eficiencia y control. Es necesario además, realizar el diseño y la implementación de estos procesos que se definen a partir de los diferentes requisitos funcionales descritos.

En el presente capítulo se describe la solución informática de la nueva versión del módulo Banco del sistema Cedrux. Se realiza un análisis de los artefactos obtenidos durante el proceso de análisis, a partir de los cuales se obtiene el diseño arquitectónico regido por los requisitos identificados por los analistas y los patrones de diseño. Se definen los componentes del sistema con las nuevas funcionalidades y se describen las clases implementadas y los estándares de codificación.

2.2 Modelo de diseño.

El diseño es el proceso de aplicar distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles como para permitir su realización física. Con el diseño se pretende construir un sistema que satisfaga determinada especificación del sistema, se ajuste a las limitaciones impuestas por el medio de destino y respete requisitos sobre forma, rendimiento utilización de recursos y costo.

El diseño es el primer paso en la etapa de desarrollo. Es el proceso de establecer la arquitectura, componentes, módulos y datos de un sistema suficientemente detallado para satisfacer ciertos requerimientos en la elaboración de un producto de software, a partir de la aplicación de distintas técnicas y principios. (39)

2.2.1 Valoración del análisis.

A partir de los procesos bancarios que se gestionan en el módulo Banco se realiza una valoración de las nuevas funcionalidades detectadas y los artefactos generados por el análisis. Se valida que los requerimientos funcionales den solución a las necesidades del usuario final. La correcta definición de los requisitos permite la retroalimentación del usuario con el diseño, lo que permite un eficiente entendimiento del sistema diseñado.

A continuación se muestra una tabla con los resultados de la valoración de los artefactos entregados por los analistas.

Tabla 1: Corrección de errores de artefactos entregados por los analistas.

Artefactos generados en diseño	Cantidad de artefactos	Con problemas	Corregidos
Descripción de requisitos.	23	1	1
Prototipos de interfaz de usuario.	7	1	1

Se realizó una revisión a las especificaciones de los requisitos establecidos para determinar si no existen ambigüedades o inconsistencias además de determinar si los errores encontrados durante la etapa de análisis fueron corregidos. No se detectaron errores, deficiencias u omisiones importantes por lo que fueron aprobadas las especificaciones de los requisitos. Se revisaron además los artefactos generados en el proceso de análisis de software, actividades de documentación, revisión y control de cambios. Se analizaron los requerimientos funcionales detectados a partir de la corrección de no conformidades detectadas por el equipo de calidad, encontrándose correctamente fundamentadas y documentadas.

Luego de comprobado que las necesidades de los clientes están cubiertas por los nuevos requerimientos funcionales y la corrección de las no conformidades, se procede a la realización del diseño y la implementación del módulo Banco.

Ver prototipos de interfaz de usuario obtenidos en el Anexo 8 de la versión digital del documento.

2.2.2 Principales funcionalidades.

Los requisitos funcionales obtenidos se agruparon atendiendo la funcionalidad y al componente que pertenecen:

- **Componente Instrumento.**
    1. Realizar el reconocimiento de cobros anticipados.
    2. Realizar la devolución de cobro anticipado.
    3. Realizar el recobro de pagos anticipados.
    4. Liquidar derechos fiscales y de cobro.
    5. Mostrar título de cuenta en instrumentos.
    6. Mostrar número de comprobante y estado de cuenta.
    7. Cancelar instrumentos según causas de cancelación.
    8. Mostrar antigüedad en obligaciones de pago.
    9. Cancelar de carga inicial.

10. Visualizar comprobante de operaciones.
11. Validar fechas y notificación de vencimiento de cheques.
12. Obtener la vista previa de instrumentos seleccionados.
13. Realizar filtrado de instrumentos por estado y por número.

- **Componente Configuración.**
    14. Configurar instrumentos de entrada.
    15. Visualizar datos de las cuentas bancarias.
    16. Gestionar causas de cancelación.
    17. Configurar talonarios para tipos de documentos, asociados a las cuentas bancarias.
    18. Configurar instrumentos asociados a talonarios.
- **Componente Estado de Cuenta.**
    19. Mostrar operaciones por cuentas bancarias.
    20. Realizar búsqueda en operaciones bancarias efectuadas.
    21. Gestionar conciliación según saldo ajustado.
    22. Gestionar conciliación saldo según libro
    23. Gestionar conciliación saldo banco.

2.2.3 Principales procesos de negocio descritos.

Los requerimientos funcionales descritos se establecieron a partir de los nuevos procesos detectados y se agruparon de acuerdo con su objetivo o función. A continuación se realiza la descripción de los procesos.

- **Gestionar motivos de cancelación:** El sistema permite gestionar los motivos de cancelación de las operaciones que se hayan realizado, reflejando además la fecha en que se realiza esta cancelación. Los motivos de cancelación son conceptos que podrán ser configurados por el cliente. En el caso de que se cancelen operaciones de la carga inicial u operaciones en los que se hayan realizado cambios en las cuentas, se permitirá especificar nuevas cuentas para la nueva contabilización.
- **Gestionar la liquidación de derechos fiscales y de cobro:** Los derechos de cobro tanto fiscales como aquellos contraídos con clientes son gestionados previamente en el subsistema Cobros y Pagos. En el subsistema Banco se procede a la liquidación de estos conceptos mediante la emisión de instrumentos bancarios que generarán una contabilización que afecta la cuenta bancaria y la cuenta por cobrar correspondiente.

- **Gestionar la cancelación de instrumentos de la carga inicial:** Los instrumentos bancarios registrados en la carga inicial son aquellos que en el período de apertura se encontraban "en tránsito", o sea, que fueron emitidos y contabilizados en los libros de la entidad pero el banco no los había reflejado en su contabilización. Estos instrumentos no tienen especificada la operación bancaria a que dio lugar por lo que al cancelarlos se debe especificar la cuenta contable que servirá de contrapartida. El proceso de cancelación de instrumentos de carga inicial consiste en especificar esta nueva cuenta contable y generar la contabilización correspondiente.
- **Gestionar el reconocimiento de cobros y pagos anticipados:** Se deben establecer los instrumentos de entrada (recepción) o de salida (emisión) para así tener un control y diferenciación de cada movimiento bancario de la entidad. Consiste en reconocer un nuevo pago anticipado que genera un derecho de cobro, o un cobro anticipado que genera una obligación de pago, que posteriormente serán liquidados.
- **Gestionar la conciliación bancaria:** Consiste en calcular la conciliación bancaria con el objetivo de conciliar el saldo ajustado según operaciones en el libro de contabilidad y el saldo ajustado según banco (efectivo real en banco más documentos en tránsito). La conciliación bancaria se debe calcular al finalizar las operaciones y se determina para cada cuenta bancaria. Se debe realizar por los tres métodos existentes (saldo ajustado, saldo según libro y saldo según banco).
- **Gestionar talonario de instrumento bancario:** Consiste en definir los talonarios o chequeras por cada tipo de instrumento de pago a considerar en el sistema y por cada una de las cuentas bancarias configuradas, con el fin de mantener el control de la consecutividad en el uso de los documentos primarios al emitir un pago.
- **Gestionar instrumento bancario:** En este proceso se agrupan las actividades que se corresponden con la gestión de instrumentos bancarios, con los que se realizarán movimientos bancarios que afectarán la cuenta bancaria y generarán liquidaciones de derechos de cobro y obligaciones de pago registradas previamente en Cobros y Pagos, reconocimiento de cobros y pagos anticipados, anticipos para caja, reembolsos de fondos y operaciones automáticas.
- **Gestionar estado de cuenta:** Consiste en marcar todas las operaciones cargadas por el banco a una cuenta bancaria en una fecha dada, proponiéndose la del día en curso. Para este proceso se utiliza el estado de cuenta, emitido por el banco al finalizar el día y el

registro de operaciones financieras que han movido a la cuenta bancaria, afectando el registro contable y que aún se encuentran en tránsito, o sea, no han sido procesados por el banco. El estado de cuenta es utilizado posteriormente para el cálculo de la conciliación bancaria. En el estado de cuenta se reflejan además las operaciones realizadas erróneamente por el banco que se considerarán diferencias.

2.2.4 Patrones de diseño empleados.

A continuación se relacionan los patrones de diseño empleados y su aplicación en la realización de los diagramas de clases del diseño para su posterior implementación.

- **Experto:** Se estableció en todos los componentes con la asignación de responsabilidades específicas por cada una de las clases del sistema. Los componentes cuentan con las clases controladoras, modelos y entidades, que poseen funcionalidades específicas atendiendo a la actividad que realizan y los procesos que gestionan. Así mismo se modeló una clase entidad por cada tabla de la base de datos, estableciendo un trabajo único por el experto en la información que maneja.
- **Creador:** En el módulo se emplea este patrón para garantizar el bajo acoplamiento, encapsulación y posibilitar la reutilización de código. Por cada componente las clases controladoras son las encargadas de crear las modelos y éstas a su vez las entidades.
- **Controlador:** Cada componente del módulo cuenta con las clases Controladoras que son las encargadas de gestionar el acceso a lógica de negocio y controlar todo el proceso. En el caso de que sean muy extensas, éstas se dividen en varias controladoras para separar la carga de procesamiento, disminuir la complejidad y responsabilidad de las clases.
- **Bajo acoplamiento:** Cada uno de los componentes está diseñado bajo este principio, lo que permite el desarrollo por separado a partir de las especializaciones individuales de cada uno. El intercambio de información entre componentes se realiza a través de servicios implementados que se encargan de distribuir la información para la realización de procesos dependientes, garantizando así el bajo acoplamiento. A su vez, el modelo relacional está definido de forma que no exista dependencia elevada entre las tablas para que se posibilite la reestructuración de los datos sin que afecte considerablemente el proceso general del módulo.
- **Alta cohesión:** El diseño y la dependencia entre clases están elaborados a partir de las funcionalidades que realizan, presentando alta relación de afinidad en las operaciones que

controlan. En el caso de las actividades de alta complejidad comparten relación con otros objetos, disminuyendo la carga de transacciones entre ellas.

- **Fachada:** El empleo de servicios posibilita el bajo acoplamiento y da aplicación a este patrón. El empleo de la clase interfaz de servicio por cada uno de los componentes permite la distribución de los servicios a través de los IOC que implementan todas las funcionalidades que el componente puede brindar.

2.2.5 Estructura del módulo banco.

El módulo Banco está estructurado en componentes que se encargan de gestionar los procesos bancarios, existiendo una relación interna de bajo acoplamiento y alta cohesión.

De forma general, la distribución de componentes con los que cuenta el módulo es la siguiente:

- **Configuración:** Gestiona los tipos de cuentas bancarias, las cuentas bancarias, los tipos de documentos asociados a talonarios de instrumentos y los talonarios de instrumentos asociados a las cuentas bancarias**.**
- **Instrumento:** Gestiona los instrumentos bancarios de entrada y salida, liquida obligaciones, realiza reembolsos, cobros y pagos anticipados.
- **Estado de cuenta:** Procesa los estados de cuenta, gestiona las conciliaciones bancarias, las diferencias-correcciones con banco y las operaciones asociadas a los estados de cuenta.
- **Cierre**: Contiene la funcionalidad realizar cierre de período contable y de ejercicio contable de módulo.
- **Carga Inicial**: Permite introducir al sistema los estados de cuenta, las diferencias de los estados de cuenta y los instrumentos y realizar conciliación bancaria.
- **Recuperaciones**: Gestiona los reportes del módulo.

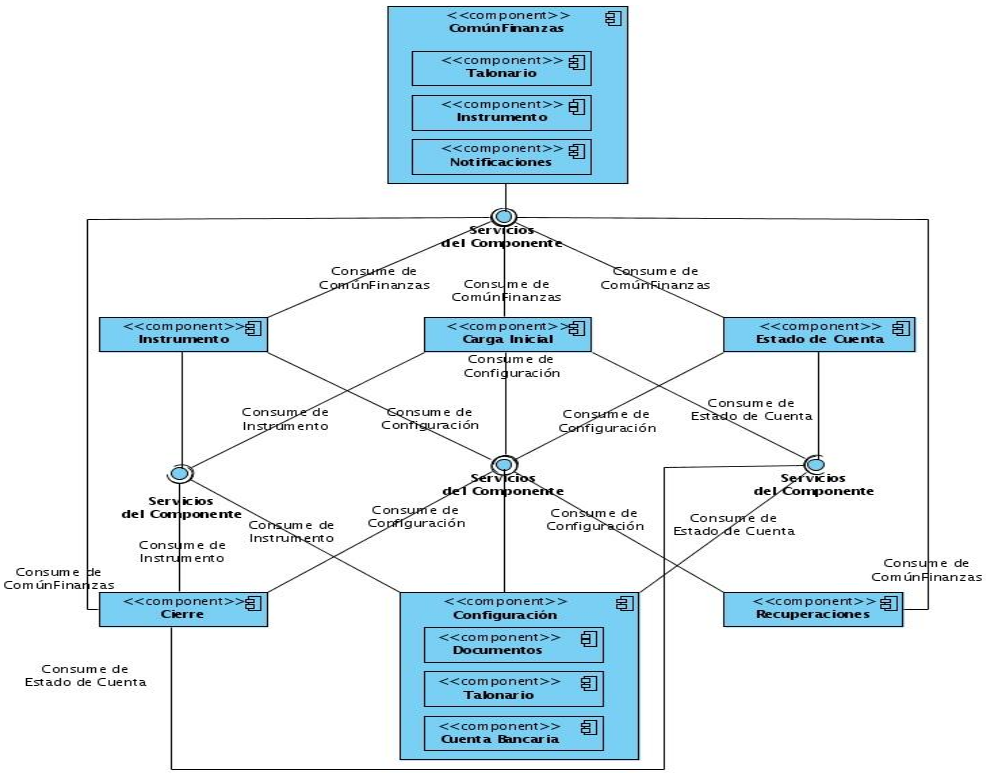

Figura 3: Diagrama de componentes.

La gestión de los instrumentos correspondientes a una operación realizada, asociados a cada talonario perteneciente a una cuenta, es responsabilidad del componente *Instrumento*. La clase interfaz *InstrumentoBancoService*, perteneciente a este componente brinda 20 servicios, consumidos por los componentes *CuentaBancaria*, *Cierre* y *CargaInicial*. Consume servicios del componente de integración *ComúnFinanzas* y del subcomponente *Talonarios,* ubicado en *Configuración.* Se integra con los subsistemas Caja, Cobros y Pagos, Configuración, Estructura y Composición, Contabilidad y Capital Humano.

El componente *Configuración* gestiona las cuentas bancarias, configurando el tipo de cuenta que atiende la entidad a través del subcomponente *CuentaBancaria*. Éste contiene las interfaces *CuentaService* y *TipoCuentaService que* brindan *18 servicios*. Excepto el subcomponente *Talonario,* los demás componentes del subsistema consumen de estos servicios. Consume servicios de los componentes *Instrumento* y *EstadoCuenta*, y del componente de integración C*omúnFinanzas*. Se integra con los subsistemas Estructura y Composición, Caja, Configuración y Contabilidad. La gestión de los talonarios asignados a cada cuenta bancaria es realizada por el subcomponente *Talonario*. Éste no brinda servicios, sólo consume del componente de integración C*omúnFinanzas*.

La gestión de los estados de las cuentas de la entidad y el proceso de conciliación bancaria son realizados por el componente *EstadoCuenta*. Las interfaces *EstadoCuentaService* y *ConciliacionService* pertenecientes a este componente brindan 29 servicios que son consumidos por *CuentaBancaria*, *Instrumento*, *Cierre*, *Recuperaciones* y *CargaInicial*. Consume servicios del subcomponente *Cuenta Bancaria* del propio módulo y del componente de integración C*omúnFinanzas*. Se integra con los subsistemas Configuración, Estructura y Composición y Contabilidad.

El componente *CargaInicial* ingresa al sistema la información acumulada antes del momento de iniciar el procesamiento en el sistema. No brinda servicios. Consume de los componentes *CuentaBancaria*, *Instrumento* y *EstadoCuenta* del propio módulo y del componente de integración C*omúnFinanzas*. Se integra con los subsistemas Caja, Configuración y Contabilidad. (40).

El componente *Cierre* realiza el cierre diario de período contable y de ejercicio fiscal. No ofrece servicios, pero consume de los componentes *CuentaBancaria*, *Instrumento*, *EstadoCuenta* y del componente de integración C*omúnFinanzas*. Se integra con el subsistema Caja. (40).

El componente *Recuperaciones* gestiona los reportes del módulo. No ofrece servicios, pero consume de los componentes *CuentaBancaria*, *EstadoCuenta* y del componente de integración C*omúnFinanzas. S*e integra con el subsistema Caja. (40).

2.2.6 Componente *ComúnFinanzas.*

La integración de operaciones que afectan a todos los módulos del subsistema Finanzas, así como el empleo de talonarios y la generación de instrumentos para realizar operaciones contables y respaldar la ejecución de éstas, es realizada por el componente *ComúnFinanzas.* Éste integra funcionalidades comunes entre los módulos Banco, Caja y Cobros y Pagos. Este componente fue diseñado para el registro y la centralización de las operaciones. Se recibe individualmente las peticiones de los distintos módulos asociados a él de manera simultánea, lo que disminuye el consumo de recursos y la disminución de tiempo para dar respuesta a las peticiones realizadas para el desarrollo de la solución.

El componente *ComúnFinanzas* cuenta con 3 subcomponentes:

- **Instrumentos:** Gestiona todos los documentos e instrumentos generados en el subsistema Finanzas.
- **Talonarios:** Gestiona los talonarios de los documentos e instrumentos empleados en el subsistema Finanzas.
- **Notificaciones:** Genera las notificaciones relacionadas con la Gestión de Créditos.

El subcomponente *Instrumento* brinda un conjunto de funcionalidades empleadas por los subcomponentes *Talonario e Instrumentos* a través de las interfaces *DocumentoFinanzasService* e *InstrumentofnzService*. A su vez, los componentes de los módulos Caja, Banco y Cobros y Pagos también consumirán de ellas para realizar su negocio. El subcomponente *Talonario* ofrece servicios a los otros componentes del subsistema Finanzas a través de la interfaz *NomTalonarioService*. La gestión de notificaciones es realizada por el subcomponente *Notificaciones* a través de la interfaz NotificacionesfnzService, que brinda servicios al subcomponente *Créditos.*

2.2.7 Diseño de clases por componentes.

El diagrama de clases del diseño describe la estructura de un sistema, donde se muestran las clases, las interfaces y la relación que existe entre ellas. Esta descripción contiene los siguientes elementos:

- Clases, atributos, métodos y relación entre ellas.
- Interfaces con sus componentes.
- Navegabilidad.
- Dependencias. (42)

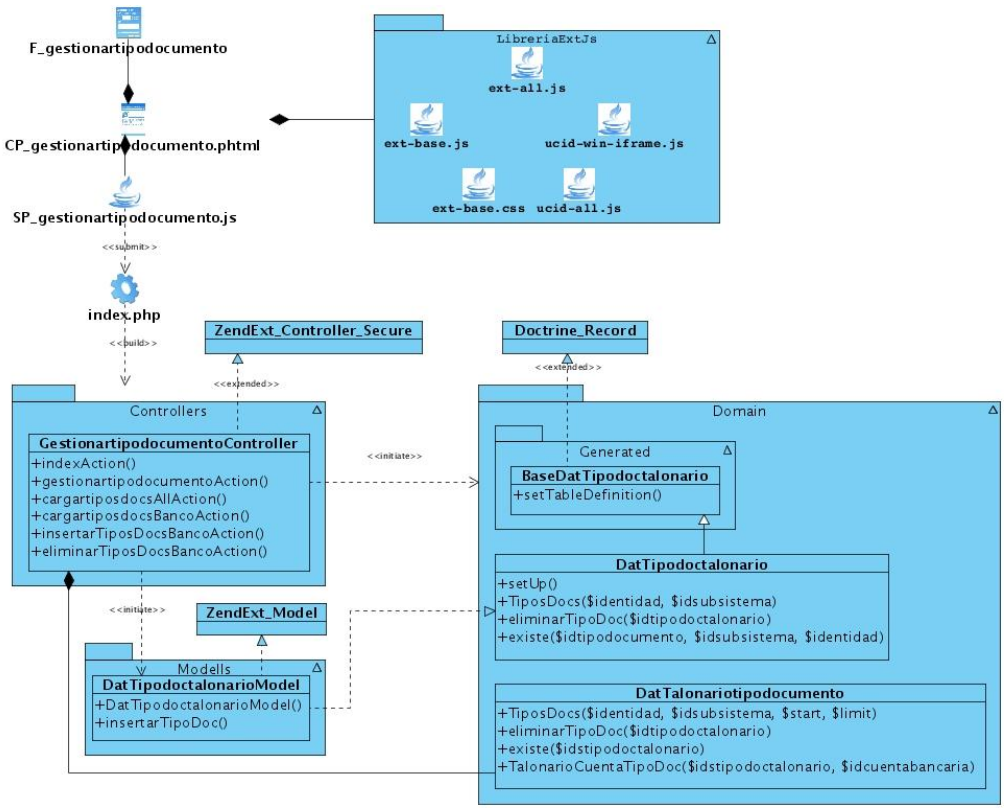

Figura 4: Diagrama de clases del diseño del subcomponente documento, componente configuración.

Como el sistema Cedrux constituye una aplicación web, el diseño de clases fue realizado con estereotipos web para su representación. Las clases .phtml representadas en el diagrama, contienen los elementos generados por las librerías JavaScript y ExtJs que son responsables de mostrar la información necesaria para interactuar con el usuario y visualizan la información necesaria para seleccionar los tipos de documentos por asociar a los talonarios, así como los tipos de documentos previamente asociados a talonarios. Los archivos .js son los encargados de realizar el envío de los datos a la página servidora index.php. Esta página envía y construye a las clases controladoras que controlan todo el flujo de información entre la vista y el modelo, además de gestionar la implementación de las funcionalidades. La seguridad es gestionada por la clase *ZendExt_Controller_Secure,* de la cual heredan.

Las clases del modelo de negocio son representadas en el paquete *Model,* que contiene la descripción del negocio y heredan de *ZentExt_Model* para la seguridad en los datos persistentes, así como el dominio con las clases persistentes que heredan de *Doctrine_Record* para mapear del modelo objeto al modelo relacional.

Ver otros diagramas en el Anexo 2 de la versión digital del documento.

2.3 Modelo de datos.

En el modelo relacional diseñado para el módulo se agruparon las tablas por componentes. La distribución de las tablas se realizó para que se puedan aprovechar las ventajas del diseño por componentes y la reutilización de los mismos.

Ver diagrama de modelo de datos en el anexo 3 de la versión digital del documento.

2.4 Modelo de implementación.

2.4.1 Integración entre componentes.

El patrón MVC gestiona el flujo de información desde la vista (view) a través del negocio (controllers), hacia el acceso a datos (models) y viceversa, consta de cuatro nodos de integración: vista-controlador, controlador-modelo, modelo-framework Doctrine y Doctrine-base de datos. La comunicación entre capas del mismo componente se realiza con la llamada directa a los métodos.

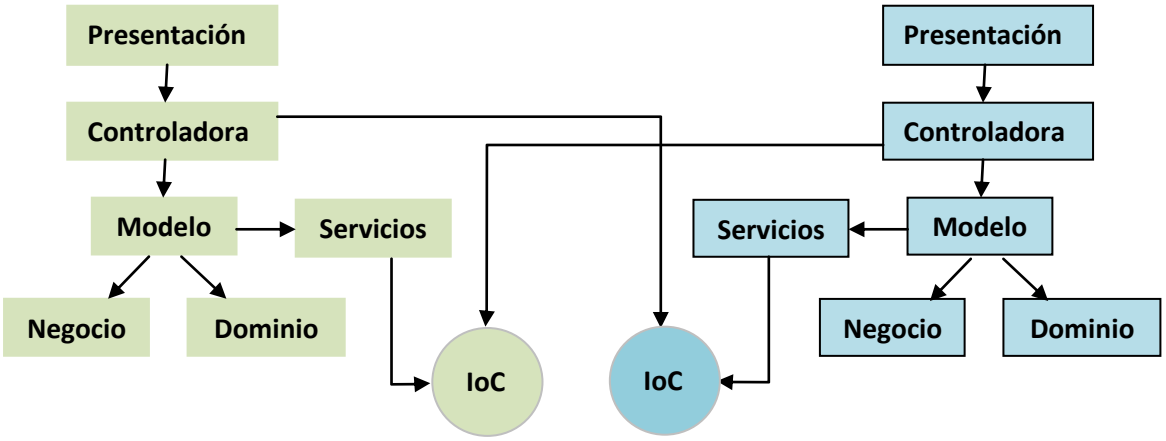

Figura 5: Integración entre componentes.

La integración para la comunicación entre módulos y componentes se realiza empleando el principio arquitectónico de inversión y control (IoC), que está implementado en el framework Zend_Ext y permite realizar la inyección de dependencias y operar sobre distintos esquemas en la base de datos, realizando las transacciones pertinentes. En este componente se define el fichero ioc.xml, que contiene la ubicación de los componentes y los servicios que brindan las clases services correspondientes. La integración se realiza tanto interna entre módulos como externa entre subsistemas.

- IoC Interno: para la integración entre componentes de un subsistema.
- IoC Externo: para la integración entre subsistemas.

2.4.2 Consumo de servicios externos.

La gestión de actividades bancarias, generan un conjunto de acciones que son gestionadas a partir de la integración con otros subsistemas. Esta integración es necesaria pues son otros subsistemas los que tienen la responsabilidad de realizar operaciones como la administración de las cuentas contables, registro de las actividades contables, control de las obligaciones de pago y derechos de cobro, chequeo del efectivo contable de la entidad para la realización de los estados de cuenta y los servicios de seguridad, entre otros. El módulo banco realiza estas operaciones y gestiona actividades de su negocio a partir del consumo de servicios externos brindados por los demás módulos.

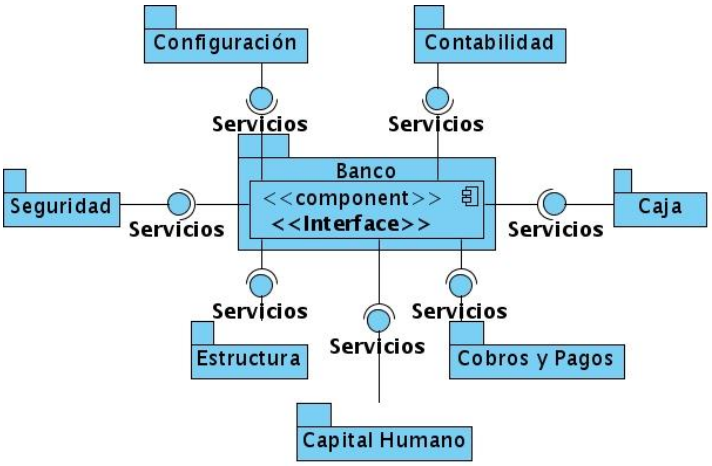

Figura 6: Consumo de servicios de subsistemas externos.

A continuación se realiza una breve descripción de las actividades que realizan los subsistemas con los que se relaciona el módulo Banco.

- **Seguridad:** Subsistema encargado de controlar la seguridad del sistema Cedrux. Gestiona las funcionalidades que brindan cada uno de los subsistemas y de estas sus acciones. De igual forma se gestionan los roles, servicios consumidos y ofertados a otros subsistemas, los usuarios y los permisos que tienen cada uno de estos dentro de las aplicaciones.
- **Contabilidad:** Subsistema encargado de configurar el formato de los estados financieros y definir las cuentas para registrar los ingresos o gastos por redondeos en las diferencias de las tasas de cambio. Permite gestionar las cuentas contables y los niveles de análisis o aperturas necesarios para la entidad, así como los tipos de cuentas. Realiza la gestión de los comprobantes donde se registran diariamente pases de las cuentas que son afectadas por operaciones contables, así como también permite registrar y contabilizar los

comprobantes de operaciones que son generados en los demás subsistemas. Gestiona el cierre de las operaciones contables en un período intermedio o de todo el ejercicio económico. Brinda los resúmenes del estado económico-financiero de la entidad mediante los Estados financieros, el Mayor, el Submayor y el Balance de comprobación de saldos.

- **Caja:** Subsistema encargado de gestionar la existencia de medios monetarios y valores depositados en las cajas de la entidad. Comprenden entre otros los importes que se ingresan en la caja para ser depositados en las cuentas bancarias correspondientes. Incluyen las existencias de cheques recibidos en divisas por entidades que no generan estas monedas, para pagos a suministradores así como los importes y cheques recibidos en moneda nacional y en divisas para ser depositados en las cuentas bancarias. Realiza operaciones de arqueo, y cierre.
- **Cobros y Pagos:** Subsistema encargado de gestionar las cuentas por cobrar y por pagar de la entidad. Las cuentas por pagar son las obligaciones presentes provenientes de las operaciones de transacciones pasadas. Las cuentas por cobrar representan derechos que tiene una entidad por las mercancías vendidas a crédito, servicios prestados, entre otros. Realiza operaciones de liquidación de los cobros y pagos anticipados que han sido reconocidos también por la entidad y se lleva un control de cuánto va quedando pendiente. Se hace además un registro de las conciliaciones realizadas con clientes y proveedores de los derechos y obligaciones pendientes, y como resultados de las mismas se puede pactar un nuevo plazo para cobrar o pagar y producir cancelaciones por expedientes de estas cuentas.
- **Estructura:** Subsistema encargado de gestionar la estructura organizativa de las entidades y agruparlas. En este se crean todas las entidades y su nivel de subordinación física. Luego el resto de los subsistemas se encargan de desarrollar los procesos especializados para algunas de estas entidades que fueron previamente creadas. Permite al usuario gestionar las subordinaciones lógicas a partir de la estructura organizativa antes creada.
- **Configuración:** Subsistema encargado de gestionar la configuración del sistema Cedrux. Modela los principales parámetros del sistema, los nomencladores generales del ERP, multimoneda y modela la existencia de objetos, operaciones y documentos. Brinda servicios como: obtener ejercicio y período contable, obtener período contable de apertura, obtener período contable de cierre.

- **Capital Humano:** Subsistema encargado de crear la estructura de la entidad pudiéndose obtener la plantilla de cargos y ocupaciones, además de generar automáticamente todos los puestos de trabajo de un área y un cargo determinado. Permite definir pagos adicionales de diferentes tipos porcentuales, fijos, proporcionales al tiempo trabajado y por tarifa. A su vez, realiza la formalización de la relación laboral realizando automáticamente el movimiento de nómina correspondiente. Genera el reporte de incidencias a los trabajadores y lleva el control de la provisión de las vacaciones de cada uno de los trabajadores.

2.4.3 Estándares de codificación.

Los estándares de codificación son pautas de programación que están enfocadas a la estructura de la lógica del código para facilitar su lectura, comprensión y mantenimiento.

Las normas de codificación definidas por la línea de arquitectura están enfocadas a lograr una mejor comprensión del código que responda a la complejidad dada por la cantidad de componentes y el alto nivel de integración que existe entre ellos. Esto permite un mejor entendimiento por parte del personal involucrado en el desarrollo del sistema, además de garantizar un código legible y reutilizable.

2.4.3.1 PascalCasing.

PascalCasing establece que los nombres de los identificadores, las variables, métodos y clases están compuestos por una o más palabras juntas, iniciando cada palabra con letra mayúscula y el resto en minúscula.

Nomenclatura de las clases según su tipo.

Todas las clases están nombradas siguiendo el estándar PascalCasing, nombrándolas de acuerdo con el propósito y la función que realizan.

- **Controllers:** Clases controladoras del dominio, su identificador está estructurado por el nombre de la misma seguido por la palabra “Controller”. Ejemplo: GestionarInstrumentoBancarioController.
- **Bussines:** Clases que gestionan la lógica del negocio, su identificador está estructurado por la función que realizan seguido de la palabra “Model”. Ejemplo: ObtenerBancariosModel.

- **Domain:** Clases que gestionan el acceso a la base de datos a través de consultas, su identificador está estructurado por el nombre de la tabla a la que acceden, pueden incluir el prefijo “Dat” o “Nom”. Ejemplo: DatMovimientoBancario y NomTipooperacionbnk.
- **Generated:** Clases base del dominio encargadas de realizar el mapeo de modelo de objetos al modelo relacional, su identificador está estructurado por el prefijo “Base”, seguido de “Dat” o “Nom” y luego el nombre de la tabla a la que hacen referencia. Ejemplo: BaseDatMovimientoBancario y BaseNomTipooperacionbnk.
- **Validators:** Clases que validan, su identificador está estructurado por el identificador seguido de la palabra “Validator”. Ejemplo: TipoMovimientoValidator.
- **Services:** Constituyen las clases que ofrecen servicios en los componentes, su identificador está estructurado con un nombre sugerente de acuerdo con las operaciones que realizan y prestaciones que brindan, seguido de la palabra “Service”. Ejemplo: EstadoCuentaService.

2.4.3.2 CamelCasing.

CamelCasing es similar a PascalCasing con diferencia en la letra inicial del identificador que no comienza con mayúscula. Esta notación se utilizó para el nombre de funciones y atributos.

Nomenclatura de las funciones.

Los identificativos de las funciones o métodos se escriben con la primera palabra en minúscula de acuerdo con la función que realizan. Ejemplo: obtenerOperacionesGuardadas.

Los denominadores de las acciones de las clases controladoras tienen la peculiaridad de ir seguidos por la palabra “Action”. Ejemplo: insertarEstadoAction.

Nomenclatura de los atributos.

El identificativo de los atributos se escribe de acuerdo con su objetivo con la primera letra en minúscula. Ejemplo: idInstrumento y newPersona.

2.4.3.3 Notación húngara.

Notación también conocida como REDDICK por el nombre de su autor, está basada en definir prefijos para cada tipo de datos según el ámbito de las variables, con la finalidad de lograr mayor comprensión del nombre de la variable, método o función.

Esta notación fue utilizada de acuerdo con los siguientes prefijos:

Tabla 2: Prefijos para la creación de variables.

Tipo de Datos	Prefijo
Arreglos	arr
Objetos	obj
Enteros	int
Cadenas	cad
Float	flt
Boolean	bool

Nomenclatura de las variables.

El identificativo de los atributos se escribe atendiendo a su objetivo y de acuerdo con el estándar CamelCasing, con la primera letra en minúscula. Ejemplo: arrCuenta, que define un arreglo de cuentas.

Nomenclatura de las constantes.

El identificativo de las constantes se realiza utilizando todas las letras en mayúscula. Ejemplo: SUBSISTEMA.

2.4.4 Descripción de clases por componente.

De acuerdo con la distribución de las clases y las funcionalidades que realizan, a continuación se realiza la descripción del subcomponente *Documento*, perteneciente al componente *Configuración*.

Clase Controladora.

Tabla 3: Descripción de la clase GestionarTipoDocumentoAction.

Nombre: GestionarTipoDocumentoController	
Tipo de clase: Controladora	
Para cada responsabilidad	
Funcionalidad	**Descripción**
cargarTiposDocsAllAction	Cargar tipos de documentos configurados para banco.
cargarTiposDocsBancoAction	Cargar tipos de documentos configurados para banco que pueden tener talonarios.

Clases Entidades.

Tabla 4: Descripción de la clase NomTalonariofnz.

Nombre: NomTalonariofnz	
Tipo de clase: Entidad	
Para cada responsabilidad	
Funcionalidad	**Descripción**
talonariosCaja ($identidad)	Obtener los talonarios de caja.
DameTalonarios ($idtipodoc)	Obtener todos los talonarios registrados.
obtenerArbolTalonario ($idestructura, $idpadre, $subsist)	Obtener el árbol de talonarios.
DameDisponiblesMiPadre ($id_tal_padre)	Obtener talonarios disponibles.
obtenerNumeroTalonario ($cajban, $elemasoc, $tipodoc)	Obtener número de talonarios.
obtenerTalonariosAsig ($cajban, $elem, $mifondo)	Obtener los talonarios asignados.
talonariosAsociar ($idestructura, $idpadre, $subsist, $cajban, $elemasoc, $arr_idtipodoc)	Asociar talonarios.
cuentasBanAsoc ($idestructura, $subsist, $idtipdoc)	Obtener las cuentas bancarias asociadas a los talonarios.
DameNumeroFinPorTipoDoc ($idtipodoc)	Obtener el número final de un documento dado su tipo.
ObtenerCuentaBancaria ($idtalonario)	Obtener la cuenta bancaria a la que pertenece un talonario
DameTalonariosPorTipoDoc ($idfondo, $tipoDoc)	Obtener todos los talonarios asociados a un tipo de documento.
obtenerArbolTalonarioBanco ($idestructura, $idpadre, $subsist)	Obtener todos los talonarios de banco.

Tabla 5: Descripción de la clase NomTalonariobancofnz.

Nombre: NomTalonariobancofnz	
Tipo de clase: Entidad	
Para cada responsabilidad	
Funcionalidad	**Descripción**

Obtener ($idtalonario)	Obtener un talonario.
TalonarioPorCuenta ($idcuenta)	Obtener los talonarios asociados a una cuenta.
TalonariosDispobibles ($idcuenta)	Obtener los talonarios disponibles asociados a una cuenta.
ObtenerTalonarioBancarioPorId ($idtalonario)	Obtener un talonario bancario asociado a una cuenta.

Tabla 6: Descripción de la clase NomTalonariocajafnz.

Nombre: NomTalonariocajafnz	
Tipo de clase: Entidad	
Para cada responsabilidad	
Funcionalidad	**Descripción**
Obtener ($idtalonario)	Obtener un talonario dado su id.

Tabla 7: Descripción de la clase DatTipodoctalonario.

Nombre: DatTipodoctalonario	
Tipo de clase: Entidad	
Para cada responsabilidad	
Nombre	**Descripción**
TiposDocs ($identidad, $idsubsistema)	Obtener tipo de documentos pertenecientes a un subsistema.

Ver el resto de las descripciones en el Anexo 4 de la versión digital del documento.

2.5 Conclusiones del capítulo.

Con la realización del presente capítulo se llegó a la conclusión de que el análisis propuesto por los analistas presenta una correcta descripción de las nuevas funcionalidades y cumplen con las necesidades del usuario final. A partir de los artefactos generados se desarrolló la solución de la nueva versión del módulo Banco del Sistema Integral de Gestión Cedrux. Ésta tiene como objetivo mejorar la gestión de los procesos bancarios en las entidades cubanas.

Se describieron los procesos bancarios sobre los que se desarrolló la solución. La descripción del diseño arquitectónico permitió la correcta implementación e integración con los demás subsistemas.

La descripción de clases importantes facilitó una idea general de la puesta en práctica de los patrones arquitectónicos, además de mejorar el entendimiento del código que facilite el mantenimiento y la reutilización. Se mantuvo la flexibilidad arquitectónica y la posibilidad de inclusión de nuevos procesos financieros que garanticen la independencia tecnológica del país.

Capítulo III: Validación de la Solución.

3.1 Introducción

En la industria del software, en la medida que aumente la complejidad y tamaño de un producto, mayor es el riesgo de que contenga errores y presente inconsistencias en el cumplimiento de los requisitos funcionales, y con ello que las necesidades del usuario final se vean afectadas. Por tanto, es imposible asegurar que un software sea totalmente seguro y no contenga errores. Existen diferentes formas y métodos para comprobar si el producto es lo suficientemente óptimo como para satisfacer las necesidades del cliente.

En el presente capítulo se realiza la validación de la solución propuesta, para lo que se aplican diferentes métricas de diseño con el objetivo de valorar en qué medida se garantiza la calidad y complejidad de la solución. Se aplican además pruebas exploratorias para comprobar la funcionalidad y validar la estructura de los componentes desarrollados.

3.2 Métricas para la evaluación del modelo de diseño.

Para realizar la evaluación del modelo de diseño del módulo Banco se aplicaron diferentes métricas, de las cuales a continuación se describen y muestran los resultados.

3.2.1 Métrica de Tamaño Operacional de Clase (TOC).

La métrica TOC está dada por la cantidad de funcionalidades contenidas en las clases, a partir de las cuales se determina la afectación que ejerce en el diseño.

A continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica.

Tabla 8: 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.

Tabla 9: 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.

Ver los instrumentos y la tabla de resultados para la métrica TOC en el Anexo 5 de la versión digital del documento.

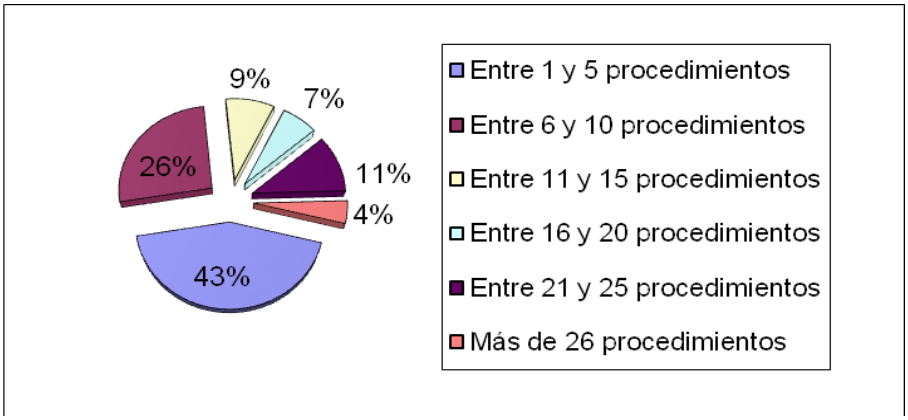

Figura 7: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos según la métrica TOC.

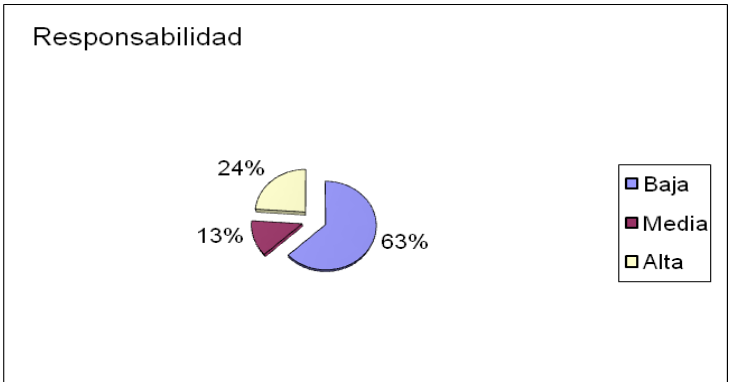

Figura 8: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

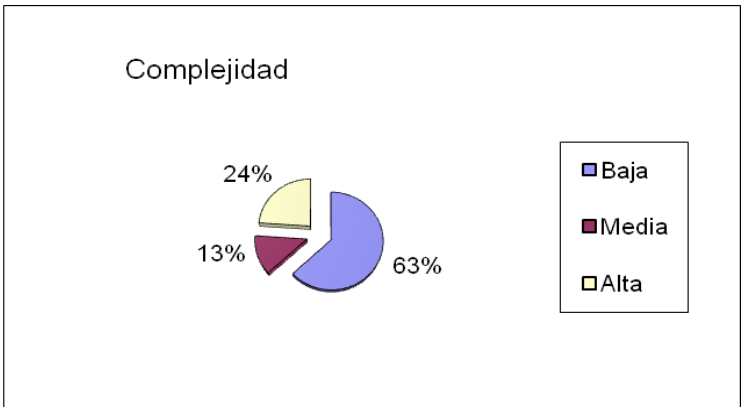

Figura 9: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

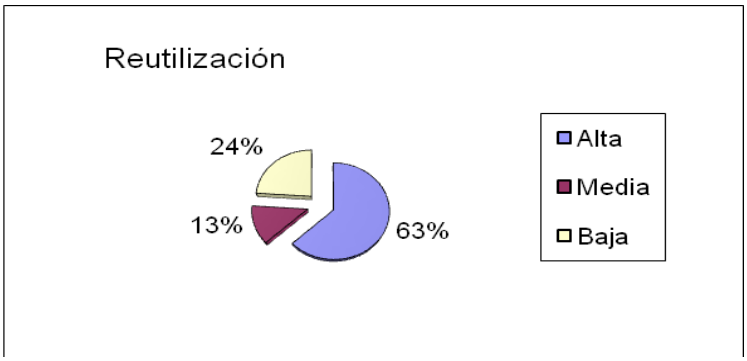

Figura 10: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Durante la evaluación de la métrica TOC los resultados obtenidos demostraron que la mayoría de las clases poseen menos cantidad de operaciones que la media registrada (43,5%), por lo que se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad de las clases se encuentran por encima del nivel medio satisfactorio a un 63%; de manera que se puede confirmar la elevada reutilización (elemento clave en el proceso de desarrollo de software) y la reducción en menor grado de la responsabilidad y la complejidad de implementación.

3.2.2 Métrica de Relaciones entre Clases (RC).

La métrica RC está dada por la cantidad de relaciones de uso existentes entre las clases contenidas en el diseño, a partir de las cuales se determina la afectación que ejerce.
A continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica.

Tabla 10: Afectaciones en el diseño según la métrica RC.

Relaciones entre Clases (RC)	
Atributos	**Afectación**
Acoplamiento	El aumento del RC provoca un aumento del acoplamiento de la clase.
Complejidad del mantenimiento	El aumento del RC provoca un aumento de la complejidad del mantenimiento de la clase.
Reutilización	El aumento del RC provoca una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	El aumento del RC provoca un aumento de la cantidad de pruebas de unidad necesarias para probar una clase.

Para la evaluación de cada atributo se establece un rango de valores que determinan la complejidad en la aplicación de la métrica.

Tabla 11: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica RC.

	Categoría	Criterio

Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

Complejidad de Mantenimiento.	Baja	<= promedio.
	Media	Entre promedio. y 2* promedio.
	Alta	> 2* promedio.

Reutilización	Baja	>2* promedio.
	Media	Entre promedio. y 2* promedio.
	Alta	<= promedio.

Cantidad de Pruebas	Baja	<= promedio.
	Media	Entre promedio. y 2* promedio.
	Alta	> 2* promedio.

Ver los instrumentos y la tabla de resultados para la métrica RC en el Anexo 6 de la versión digital del documento.

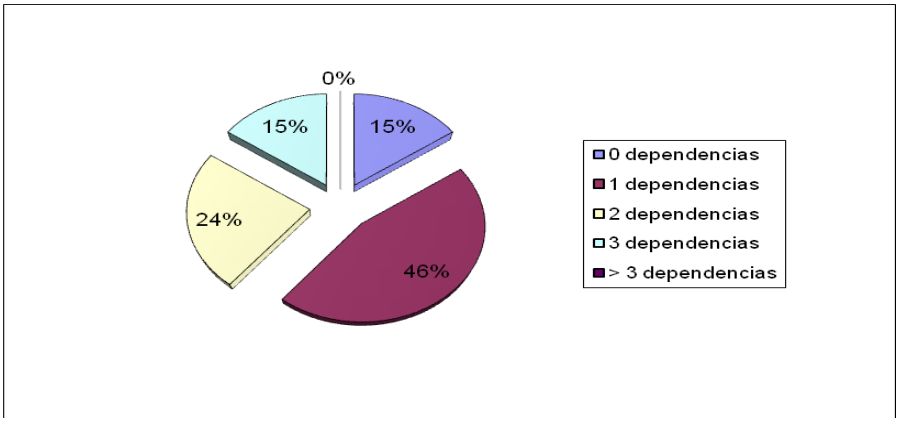

Figura 11: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos según la métrica RC.

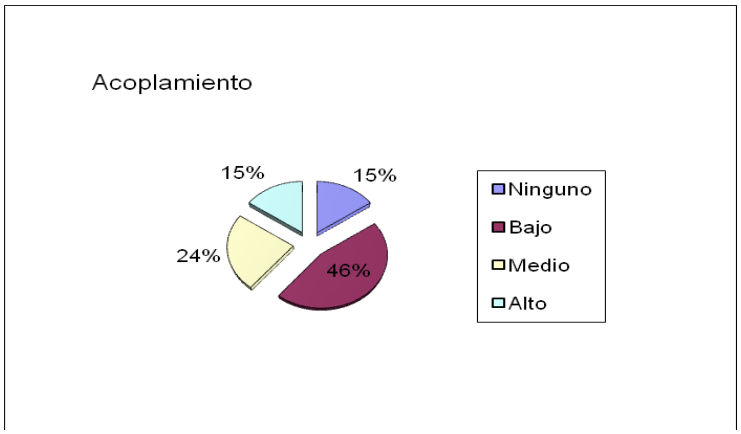

Figura 12: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

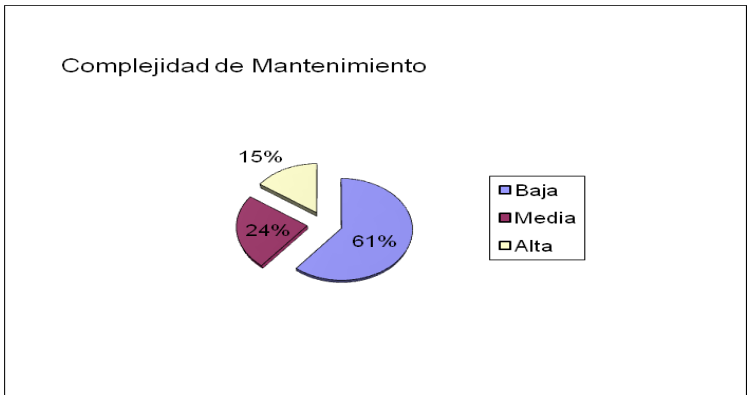

Figura 13: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

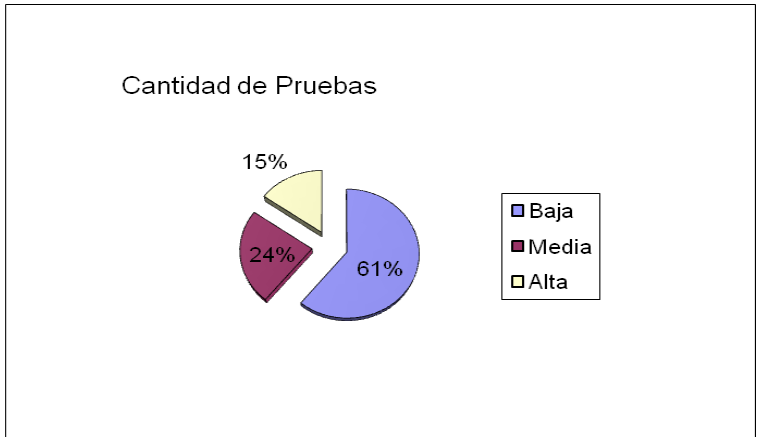

Figura 14: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

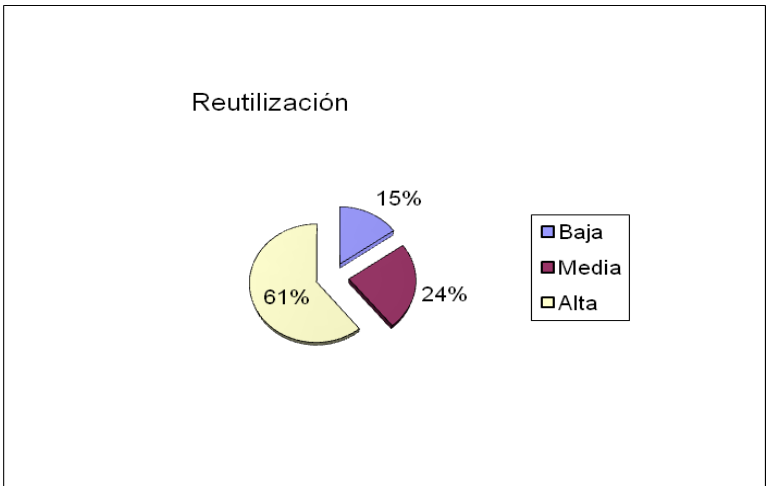

Figura 15: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Durante la evaluación de la métrica RC los resultados obtenidos demostraron que la mayoría de las clases (46%), poseen menos de 3 dependencias entre clases, por lo que se encuentra dentro de los niveles aceptables de calidad. Los atributos de calidad se encuentran en un nivel satisfactorio; en el 46% de las clases el grado de dependencia o acoplamiento es mínimo. La complejidad de mantenimiento y la cantidad de pruebas son bajas a un 61%, lo que representa valores favorables en el diseño realizado. Así mismo, existe un alto grado de reutilización al 61%, comportamiento también favorable para este atributo de calidad.

3.2.3 Evaluación del resultado de las métricas.

Para la evaluación final del diseño se promedian los valores obtenidos y se evalúan en los umbrales de los parámetros establecidos para la evaluación de la calidad del diseño.

A continuación se muestra una gráfica con el resultado final de promediar los atributos medibles según las métricas.

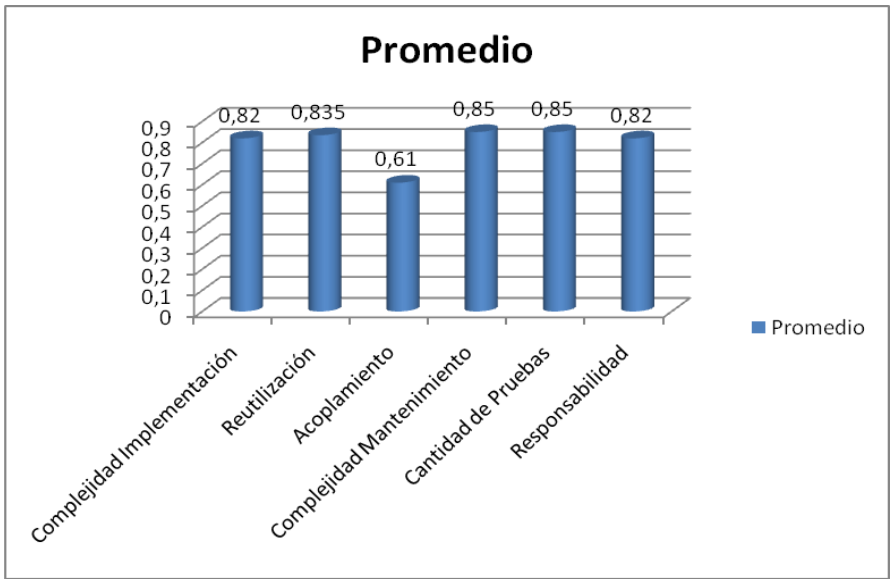

Figura 16: Promedio de valores óptimos obtenidos durante la aplicación de las métricas.

Como se muestra en la figura anterior en el caso de la complejidad de implementación, reutilización, complejidad de mantenimiento, cantidad de pruebas y responsabilidad, se encuentran dentro del rango de buena, lo que demuestra que el diseño posee una baja complejidad de implementación, mantenimiento y responsabilidad. La cantidad de pruebas necesarias aplicables al sistema no es elevada y la reutilización que provee es elevada. En el caso del acoplamiento se comportó de manera regular lo que demuestra que el sistema posee un acoplamiento medio.

De manera general el promedio de calidad del diseño obtenido por la evaluación de las métricas es de 0.82 lo que demuestra que el diseño obtenido es bueno y cumple con los requerimientos de calidad.

3.3 Niveles de prueba aplicados.

Para precisar la calidad y correcto funcionamiento del módulo, durante el desarrollo del mismo se definieron diferentes tipos de pruebas aplicables a diferentes niveles, las cuales comprenden desde el componente más pequeño hasta la integración de todos los componentes y el subsistema con el sistema en general. Las pruebas aplicadas son las siguientes:

- Pruebas Internas: Diseñadas e implementadas por el grupo de calidad interna para verificar el correcto funcionamiento del módulo.
- Pruebas de Integración: Verifican que los módulos probados operen correctamente cuando se combinen. Descubren errores en las especificaciones de las interfaces de los paquetes.

3.4 Estrategia de pruebas.

Para realizar el proceso de la aplicación de pruebas se define una estrategia de pruebas regida según el modelo de desarrollo establecido, con el fin de garantizar la calidad del software.

Se concretan los casos de pruebas, a partir de la construcción de todos los posibles caminos de ejecución, o escenarios de cada componente desarrollado, basados en los requisitos implementados, comparando cada funcionalidad descrita con lo implementado para verificar hasta qué punto concuerdan y cumplen con las necesidades del cliente. Se obtiene como resultado un listado final con los casos de prueba identificados a partir de los posibles escenarios, los resultados esperados para cada caso y las condiciones o valores requeridos para la ejecución de los distintos escenarios.

Se realizan pruebas internas del sistema antes de incorporarlo al equipo central de calidad para que realice las pruebas de liberación, tratando de garantizar la detección de la mayor cantidad de no conformidades posibles. En el nivel de integración se realiza la verificación y validación de las funcionalidades del módulo, como un conjunto de componentes integrados, según la jerarquía y comunicación entre ellos.

Independiente a la estrategia de pruebas definida, se realizan pruebas de calidad dirigidas por el equipo central de calidad de Cedrux, encargado de probar el módulo para su liberación posterior a las entidades piloto, el cual prueba cada uno de los juegos de datos así como los casos de prueba desarrollados por las líneas. Se realizan además tantas iteraciones como sean necesarias para garantizar la calidad del sistema a desplegar y comprobar que el software funciona según las expectativas del cliente, y que se encuentra ejecutando las funciones y tareas para las cuales fue construido.

3.5 Diseño de casos de prueba.

Los casos de prueba están basados en diferentes entradas que puede recibir el sistema con sus correspondientes valores de salida. Se centran en realizar pruebas de software para comprobar su funcionalidad.

Los casos de prueba demuestran que:

- Las funciones del software son operativas.
- Las entradas se aceptan de la forma adecuada produciendo el resultado esperado.
- La integridad de la información externa (por ejemplo archivos de datos) se mantiene.

Para verificar que la aplicación cumpla con los requisitos establecidos por el cliente se diseñan los casos de prueba y se realizan las pruebas internas. A continuación se especifica el caso de prueba

"Realizar cobro anticipado" el cual define que dado una operación de cobro se realice el cobro anticipado de la misma y se registre el instrumento generado por esta.

Condiciones de ejecución.

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema de Finanzas.
- Se debe seleccionar la opción **Banco/Instrumento/Instrumento.**
- Se está adicionando un instrumento bancario de tipo **Entrada**.
- Se ha seleccionado el movimiento **Anticipos.**

Requisito a probar.

Tabla 12: Descripción del caso de prueba para el requisito Realizar cobro anticipado.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Realizar cobro anticipado.	Se realiza un cobro anticipado.	EP 1.1: Realizar un cobro anticipado introduciendo datos válidos.	– Se presiona el botón **Adicionar**. – Se introducen los datos del cobro anticipado que se desea realizar. – Se presiona el botón **Aceptar.**
		EP 1.2: Realizar un cobro anticipado introduciendo datos inválidos.	– Se presiona el botón **Adicionar**. – Se introducen datos inválidos del cobro anticipado que se desea realizar.

		EP 1.3: Realizar un cobro anticipado dejando campos obligatorios en blanco.	– Se presiona el botón **Adicionar**. – Se introducen los datos del cobro anticipado que se desea realizar dejando algún campo obligatorio en blanco. – Se presiona el botón **Aceptar**.
		EP 1.4: Cancelar la operación.	– Se presiona el botón **Adicionar**. – Se introducen o no los datos del cobro anticipado que se desea realizar. – Se presiona el botón **Cancelar**.

Descripción de variable.

Tabla 13: Descripción de las variables del caso de prueba para el requisito Realizar cobro anticipado.

No.	Nombre del campo	Clasificación	Válido	Inválido
1	Tipo instrumento	Campo de texto.	NA	NA
2	Operación	Lista desplegable	NA	NA
3	No instrumento	Numérico	NA	NA
4	Importe original	Numérico	524111100.00	8888888888888888888888888888000000000000

5	Importe contable	Numérico	NA	NA
6	Moneda	Numérico	NA	NA
7	Comentario	Alfanumérico	Se realiza un cobro anticipado al cliente 521411 por servicios.	(Cadena con más de 255 caracteres.)

Juego de datos a probar.

Tabla 14: Datos de prueba del caso de prueba para el requisito Realizar cobro anticipado.

Id del escenario	Escenario	Operación	Importe original	Comentario	Respuesta del sistema
EP 1.1	Realizar un cobro anticipado introduciendo datos válidos.	V (Anticipo de cobro)	V (1000.00)	V (Se realiza un cobro anticipado por conceptos de activos fijos.)	El sistema realiza un cobro anticipado asociado al instrumento bancario.
EP 1.2	Realizar un cobro anticipado introduciendo datos inválidos.	I (Ant. “33@#@#)	I(10000000000000000000000000000000000000000000000000.0025)	V (Se realiza un cobro anticipado por conceptos de activos fijos.)	El sistema no permite registrar datos incorrectos.
		V (Anticipo de cobro)	I(Mil)	V (Se realiza un cobro anticipado por conceptos de activos fijos.)	
EP 1.3	Realizar un cobro anticipado dejando campos obligatorios en	I (Vacío)	V(1000.00)	V (Se realiza un cobro anticipado por conceptos de activos fijos.)	El sistema emite una alerta: “Este campo es obligatorio.”.

	blanco.	V (Anticipo de cobro)	I (Vacío)	V (Se realiza un cobro anticipado por conceptos de activos fijos.)	
		V (Anticipo de cobro)	V(1000.00)	I (Vacío)	

3.6 Resultados de las pruebas internas.

Para comprobar la calidad y funcionalidad del software desarrollado y con el empleo de los diseños de casos de pruebas para la aplicación, se realizaron dos iteraciones de revisiones por el grupo de calidad. Durante la realización de las pruebas se detectaron un conjunto de no conformidades que posteriormente se procedió a su resolución.

Las no conformidades detectadas en las dos etapas de pruebas se muestran en el Anexo 7 de la versión digital del documento.

3.6.1 Primera iteración de pruebas internas de calidad.

Durante la primera iteración se detectaron 21 no conformidades, de ellas 12 fueron significativas, lo que representa un 57,14% del total.

A continuación se muestra un gráfico con el porciento que representan del total la cantidad de no conformidades por tipo.

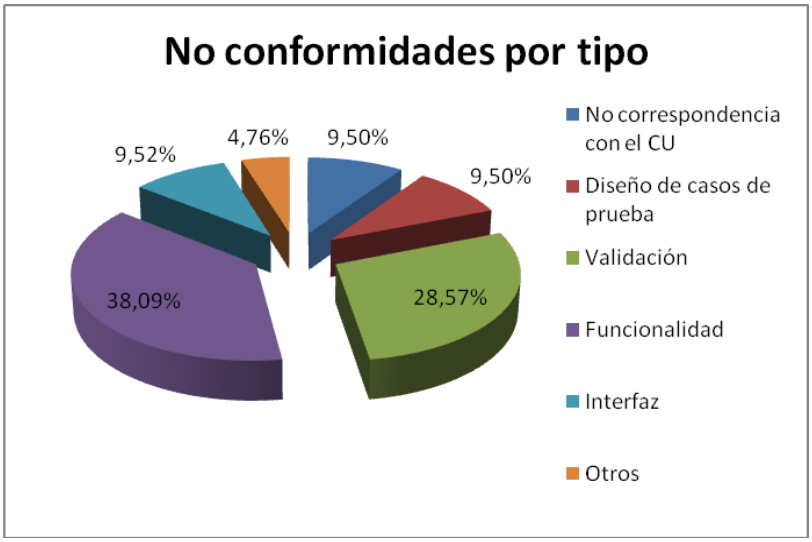

Figura 17: Porciento que representan la cantidad de no conformidades por tipo respecto al total de la primera iteración.

Las no conformidades referentes a Funcionalidades y Validación representan en mayor medida el nivel de afectación en la aplicación para un 38,09 y 28,57 porcientos respectivamente. No siendo así las de tipo No Correspondencia con CU (Casos de Uso), Diseños de Casos de Prueba e Interfaz, que representan un nivel de afectación del 9,52 % y Otras que en menor medida presentan un 4,76 %. Todas las no conformidades fueron resueltas en su totalidad para un 100 % de solución.

3.6.2 Segunda iteración de pruebas internas de calidad.

Durante la segunda iteración de las pruebas internas se detectaron 9 no conformidades. De ellas 5 fueron significativas, lo que representa un 55,55 % del total.

A continuación se muestra un gráfico con el porciento que representan del total la cantidad de no conformidades por tipo.

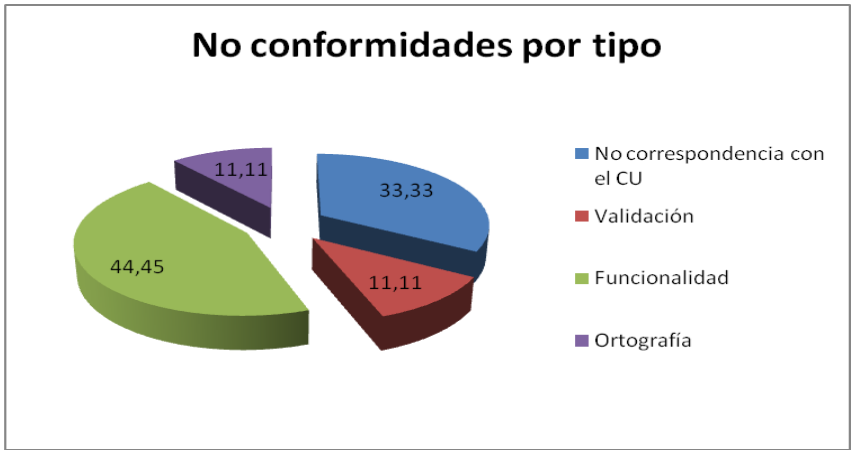

Figura 18: Porciento que representan la cantidad de no conformidades por tipo respecto al total de la segunda iteración.

Las no conformidades referentes a Funcionalidades y No Correspondencia con CU (Casos de Uso) representan en mayor medida el nivel de afectación en la aplicación para un 44,45 y 33,33 porcientos respectivamente. No siendo así las de tipo Validación y Ortografía que representan un nivel de afectación del 11,11 % del total.

Como se pudo apreciar durante la segunda iteración de pruebas, disminuyeron considerablemente el número de no conformidades detectadas, lo que demuestra un incremento de la calidad del software y un mejoramiento en cuanto a la correspondencia de los procesos descritos con respecto a lo implementado, logrando un mayor acercamiento a la satisfacción de las necesidades de los clientes y el usuario final.

Para la validación de los resultados obtenidos y como certificación de que todas las funcionalidades fueron corregidas, que la implementación cumple con el alcance planificado, la calidad requerida y no presenta no conformidades, se emitió la carta de liberación por el grupo de calidad del CEIGE. Ver la carta de liberación en el Anexo 9 de la versión digital del documento.

3.7 Conclusiones del capítulo.

Con el desarrollo del presente capítulo se llegó a la conclusión de que el diseño elaborado cumple con los diferentes aspectos medidos a partir de la caracterización cuantitativa, obtenida de la evaluación realizada mediante la aplicación de las métricas de diseño. Se aplicaron además pruebas que permitieron evaluar todos los elementos del software. Igualmente, se concluye que las funcionalidades implementadas mostraron el correcto funcionamiento y respuesta a los requisitos funcionales y garantizan las necesidades de los clientes; esto se verificó a partir de las pruebas funcionales realizadas por el grupo de calidad del CEIGE.

Con los resultados logrados de aplicar las métricas de diseño y las pruebas internas se comprobó la obtención de un software de calidad, funcionalmente probado y se evaluaron satisfactoriamente los atributos relacionados con el desarrollo de software.

Conclusiones Generales

Después de confirmar la necesidad de realizar el diseño y la implementación de la nueva versión del módulo Banco se llegó a la conclusión de que se le dio cumplimiento a los objetivos específicos de la investigación y que las tareas ejecutadas permitirán mejorar el funcionamiento de la actividad financiera en las entidades cubanas.

Se demostró, a partir de la aplicación de diferentes métricas de diseño y pruebas de calidad, que el diseño y la implementación realizados cumplen con los parámetros establecidos y dan solución a los requisitos funcionales del sistema. Se logró por tanto la obtención de la nueva versión del módulo Banco del Sistema Integral de Gestión Cedrux.

Cumpliendo con el objetivo general, se obtuvo el módulo Banco como un producto configurable y flexible arquitectónicamente, totalmente funcional que cumple con las necesidades del cliente, permite el control de los procesos bancarios descritos y se adapta a las legislaciones vigentes referentes a la actividad financiera en las empresas cubanas. El sistema se integra de manera eficiente a otros subsistemas, por lo cual permite realizar de forma ágil las operaciones contables.

La investigación realizada contribuye al correcto funcionamiento de la gestión de las actividades bancarias en las empresas cubanas y al incremento del uso de las tecnologías de la informática y las comunicaciones.

Recomendaciones

A partir del estudio realizado se recomienda:

- Continuar con la realización de pruebas funcionales que permitan garantizar la calidad del producto.
- Extender el proceso de implantación a las entidades del país.
- Analizar la posibilidad de agregar nuevos procesos de gestión bancaria a partir de la identificación de nuevas funcionalidades para futuras versiones del producto.
- De manera general, refinar la solución obtenida a partir de sugerencias y otras recomendaciones.

Referencias Bibliográficas

1. **Corzo, Giancarlo.** Desarrollo en WEB. [En línea] 2008. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

2. **Framework, Zend.** Introducción a Zend Framework. [En línea] 2010. http://manual.zfdes.com/.

3. **GBM.** SAP Enterprise Resource Planning (ERP). *Aplicaciones ERP: la nueva alternativa para la gestión de recursos.* [En línea] 2010. http://www.gbm.net/soluciones/enterprise_resource_planning.php.

4. **Gestión, Centro de Soluciones de.** *Definición del ciclo de vida de los proyectos de desarrollo de software v1.0.* 2009.

5. **Global, SAP.** SAP Solutions. [En línea] 2010. http://www.sap.com.

6. **Integral, ASSEST. Sistema de Gestión.** ¿Qué es lo nuevo en AssetsNS versión 2.0? [En línea] 2006. http://www.assets.co.cu/texto.asp?id=5.

7. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. http://www.proactiva-calidad.com/java/patrones/.

8. **Mariaelena.** *Entrevista con funcionales del Ministerio de Finanzas y Precios. Procesos bancarios de una entidad.* 2010.

9. **Mata, Manel Pérez.** TecnoRetales. *Qué es Doctrine ORM?* [En línea] 2009. http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

10. **Openbravo.** Openbravo. Manual de Usuario 1.1. [En línea] 2006. http://www.openbravo.com/docs/openbravo-manual-de-Usuario-v1.1.pdf.

11. **Hilliard, Rich.** IEEE-Std-1471-2000. [En línea] http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-2000.pdf.

12. **IBM.** Rational Host Integration Solution. [En línea] http://www-01.ibm.com/software/awdtools/hostintegration/features/?S_CMP=mav.

13. **EcuRed.** Ide de Programación. [En línea] 2011. http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

14. **PostgreSQL.** PostgreSQL 8.3.15 Documentation. [En línea] http://www.postgresql.org/docs/8.3/static/release-8-3.html.

15. **Pérez, Javier Eguilus.** Introducción a AJAX. [En línea] 2008. http://www.librosweb.es/ajax/.

16. **Consortium, World Wide Web.** Guía Breve de Tecnologías XML. [En línea] 9 de 1 de 2008. http://www.w3c.es/divulgacion/guiasbreves/tecnologiasxml.

17. **SISCONT.** SISCONT. Software Contable-Financiero. [En línea] 2009. http://www.siscont.com/DESCRIPTIVO%20SISCONT.pdf.

18. **Consultora, DISAIC.Casa.** El VERSAT-Sarasola: Sistema cubano de Gestión Contable-Financiero. [En línea] 2010. http://www.disaic.cu/modules.php?name=Content&pa=showpage&pid=818.

19. **Ciberaula.** Una Introducción a APACHE. [En línea] 2006. http://linux.ciberaula.com/articulo/linux_apache_intro/.

20. **Ramos Arias, Taimé y Torres Salas, Pedro Antonio.** *Diseño e implementación del módulo Banco del Sistema Integral de Gestión CEDRUX.* La Habana : s.n., 2010.

21. **Ciberaula.** Una Introducción a APACHE. [En línea] 2006. http://linux.ciberaula.com/articulo/linux_apache_intro/ .

22. **Language, OMG. Unified Modeling.** [En línea] 2011. http://www.uml.org/.

23. **Ramirez, Vicente Rico.** Lenguajes. *Instituto Tecnológico de Selaya.* [En línea] http://www.iqcelaya.itc.mx/~vicente/Programacion/Lenguajes.pdf.

24. **Andux, Yadira Piñera.** *Formalización y estandarización de la documentación técnica de la arquitectura tecnológica del Marco de Trabajo Sauxe versión 2.0.* La Habana : s.n., 2010.

25. **Paradigm, Visual.** Visual Paradigm for UML - UML tool for software application development. [En línea] 2011. http://www.visual-paradigm.com/product/vpuml/.

26. **TortoiseSVN.** TortoiseSVN. [En línea] 2011. http://tortoisesvn.net/.

27. **Stefan, Kung, Lubbe, Onken y Simon, Large.** *TortoiseSVN. Un cliente de subversión para Windows.* 2010.

28. **Serradilla, Juan Luis.** Controlo de Versiones con Subversion y TortoiseSVN. *Sección de Metodología, Normalización y Calidad del Software. ATICA. Universidad de Murcia.* [En línea] 2007. http://www.um.es/atica/documentos/PREsubversion.pdf.

29. **Area de las tecnologías de la Información, las Comunicaciones y la Informática aplicada. Universidad de Murcia.** Manual Básico de Creación de Páginas Web. [En línea] http://www.um.es/atica/documentos/html.pdf.

30. **Gutiérrez, Javier J.** Lenguajes y Sistemas Informáticos. *Universidad de Sevilla.* [En línea] 2008. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

31. **Baryolo, Oiner Gómez, y otros.** *Plantilla Registro de la Propiedad intelectual(Sauxe).* 2008.

32. **México, Instituto Politécnico de.** Herramientas automatizadas. [En línea] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

33. **Acebal, Cesar Fernández.** Escuela de Ingeniería Informática. *Universidad de Oviedo.* [En línea] 2010. http://aainfo.ccu.uniovi.es/ficheros/apuntes/comercio_electronico/Tema%206%20-%20Servidores%20de%20aplicaciones.pdf.

34. **México, Universidad Autónoma Indígena de.** Institución Intercultural del Estado Sinaloa. [En línea] 2010. http://www.uaim.edu.mx/web-carreras/carreras/sistemas%20computacionales/Decimo%20Trimestre/BASE%20DE%20DATOS.pdf.

35. **Hoy, Informática.** Que es un navegador web. *Informática Hoy.* [En línea] 2010. http://www.informatica-hoy.com.ar/aprender-informatica/Que-es-un-navegador-web.php.

36. **Mancha, Universidad de Castila-La.** El diseño de software. *Diseño y Programación Orientado a Objetos. Ingeniería Técnica de Informática de Sistemas y Gestión.* [En línea] 2006. http://www.info-ab.uclm.es/asignaturas/42579/pdf/01-Capitulo1.pdf.

37. **Ministerio de Relaciones Exteriores de la República de Cuba.** Informe de Cuba sobre la resolución 62/3 de la Asamblea General de las Naciones Unidas. *Cuba Minrex.* [En línea] 2006. http://embacuba.cubaminrex.cu/Default.aspx?tabid=8745.

38. **S.L., Bab Software Applications.** BabSoftware. *Diseño y desarrollo de aplicaciones web.* [En línea] 2009. http://www.bab-soft.com/es/diseno_desarrollo_aplicaciones_web.php.

39. **Prieto, Felix.** Departamento de Informática. Universidad de Valladolid. [En línea] 2009. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

40. **López, Alejandro Rivera.** UDLAP. Universidad de las Américas de Puebla. [En línea] 2008. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo2.pdf.

41. **Peredo Valderrama, Ruben, y otros.** Arquitectura para sistemas de información basada en web usando programación orientada a componentes. [En línea] 2008. http://www.google.com.cu/url?sa=t&source=web&cd=4&ved=0CCwQFjAD&url=http%3A%2F%2Fwww2.pucpr.br%2Freol%2Findex.php%2FDIALOGO%3Fdd1%3D2033%26dd99%3Dpdf&rct=j&q=arquitectura%20orientada%20a%20componentes&ei=Ncj2Td-XGIPy0gGPvZWGCw&usg=AFQjCNESTUc8qQelUE_qnvtz.

42. **Corporation, Oracle.** Netbeans IDE. [En línea] 2010-2011. http://netbeans.org/.

43. **Información, Escuela de Ingeniería y Sistemas de.** Universidad del Valle. Colombia. [En línea] 2010. http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Pruebas.pdf.

44. **Visconti, Marcello y Astudillo, Hernán.** Departamento de Informática. *Universidad Técnica de Federico de Santa Maria.* [En línea] 2010. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf.

Bibliografía

1. **Corzo, Giancarlo.** Desarrollo en WEB. [En línea] 2008. http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

2. **Framework, Zend.** Introducción a Zend Framework. [En línea] 2010. http://manual.zfdes.com/.

3. **GBM.** SAP Enterprise Resource Planning (ERP). *Aplicaciones ERP: la nueva alternativa para la gestión de recursos.* [En línea] 2010. http://www.gbm.net/soluciones/enterprise_resource_planning.php.

4. **Gestión, Centro de Soluciones de.** *Definición del ciclo de vida de los proyectos de desarrollo de software v1.0.* 2009.

5. **Global, SAP.** SAP Solutions. [En línea] 2010. http://www.sap.com.

6. **Integral, ASSEST. Sistema de Gestión.** ¿Qué es lo nuevo en AssetsNS versión 2.0? [En línea] 2006. http://www.assets.co.cu/texto.asp?id=5.

7. **Lago, Ramiro.** Patrones de diseño software. [En línea] Abril de 2007. http://www.proactiva-calidad.com/java/patrones/.

8. **Mariaelena.** *Entrevista con funcionales del Ministerio de Finanzas y Precios. Procesos bancarios de una entidad.* 2010.

9. **Mata, Manel Pérez.** TecnoRetales. *Qué es Doctrine ORM?* [En línea] 2009. http://www.tecnoretales.com/programacion/que-es-doctrine-orm/.

10. **Openbravo.** Openbravo. Manual de Usuario 1.1. [En línea] 2006. http://www.openbravo.com/docs/openbravo-manual-de-Usuario-v1.1.pdf.

11. **Hilliard, Rich.** IEEE-Std-1471-2000. [En línea] http://www.enterprise-architecture.info/Images/Documents/IEEE%201471-2000.pdf.

12. **IBM.** Rational Host Integration Solution. [En línea] http://www-01.ibm.com/software/awdtools/hostintegration/features/?S_CMP=mav.

13. **EcuRed.** Ide de Programación. [En línea] 2011. http://www.ecured.cu/index.php/IDE_de_Programaci%C3%B3n.

14. **PostgreSQL.** PostgreSQL 8.3.15 Documentation. [En línea] http://www.postgresql.org/docs/8.3/static/release-8-3.html.

15. **Pérez, Javier Eguilus.** Introducción a AJAX. [En línea] 2008. http://www.librosweb.es/ajax/.

16. **Consortium, World Wide Web.** Guía Breve de Tecnologías XML. [En línea] 9 de 1 de 2008. http://www.w3c.es/divulgacion/guiasbreves/tecnologiasxml.

17. **SISCONT.** SISCONT. Software Contable-Financiero. [En línea] 2009. http://www.siscont.com/DESCRIPTIVO%20SISCONT.pdf.

18. **Consultora, DISAIC.Casa.** El VERSAT-Sarasola: Sistema cubano de Gestión Contable-Financiero. [En línea] 2010. http://www.disaic.cu/modules.php?name=Content&pa=showpage&pid=818.

19. **Ciberaula.** Una Introducción a APACHE. [En línea] 2006. http://linux.ciberaula.com/articulo/linux_apache_intro/.

20. **Ramos Arias, Taimé y Torres Salas, Pedro Antonio.** *Diseño e implementación del módulo Banco del Sistema Integral de Gestión CEDRUX.* La Habana : s.n., 2010.

21. **Ciberaula.** Una Introducción a APACHE. [En línea] 2006. http://linux.ciberaula.com/articulo/linux_apache_intro/ .

22. **Language, OMG. Unified Modeling.** [En línea] 2011. http://www.uml.org/.

23. **Ramirez, Vicente Rico.** Lenguajes. *Instituto Tecnológico de Selaya.* [En línea] http://www.iqcelaya.itc.mx/~vicente/Programacion/Lenguajes.pdf.

24. **Andux, Yadira Piñera.** *Formalización y estandarización de la documentación técnica de la arquitectura tecnológica del Marco de Trabajo Sauxe versión 2.0.* La Habana : s.n., 2010.

25. **Paradigm, Visual.** Visual Paradigm for UML - UML tool for software application development. [En línea] 2011. http://www.visual-paradigm.com/product/vpuml/.

26. **TortoiseSVN.** TortoiseSVN. [En línea] 2011. http://tortoisesvn.net/.

27. **Stefan, Kung, Lubbe, Onken y Simon, Large.** *TortoiseSVN. Un cliente de subversión para Windows.* 2010.

28. **Serradilla, Juan Luis.** Controlo de Versiones con Subversion y TortoiseSVN. *Sección de Metodología, Normalización y Calidad del Software. ATICA. Universidad de Murcia.* [En línea] 2007. http://www.um.es/atica/documentos/PREsubversion.pdf.

29. **Area de las tecnologías de la Información, las Comunicaciones y la Informática aplicada. Universidad de Murcia.** Manual Básico de Creación de Páginas Web. [En línea] http://www.um.es/atica/documentos/html.pdf.

30. **Gutiérrez, Javier J.** Lenguajes y Sistemas Informáticos. *Universidad de Sevilla.* [En línea] 2008. http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf.

31. **Baryolo, Oiner Gómez, y otros.** *Plantilla Registro de la Propiedad intelectual(Sauxe).* 2008.

32. **México, Instituto Politécnico de.** Herramientas automatizadas. [En línea] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

33. **Acebal, Cesar Fernández.** Escuela de Ingeniería Informática. *Universidad de Oviedo.* [En línea] 2010. http://aainfo.ccu.uniovi.es/ficheros/apuntes/comercio_electronico/Tema%206%20-%20Servidores%20de%20aplicaciones.pdf.

34. **México, Universidad Autónoma Indígena de.** Institución Intercultural del Estado Sinaloa. [En línea] 2010. http://www.uaim.edu.mx/web-carreras/carreras/sistemas%20computacionales/Decimo%20Trimestre/BASE%20DE%20DATOS.pdf.

35. **Hoy, Informática.** Que es un navegador web. *Informática Hoy.* [En línea] 2010. http://www.informatica-hoy.com.ar/aprender-informatica/Que-es-un-navegador-web.php.

36. **Mancha, Universidad de Castila-La.** El diseño de software. *Diseño y Programación Orientado a Objetos. Ingeniería Técnica de Informática de Sistemas y Gestión.* [En línea] 2006. http://www.info-ab.uclm.es/asignaturas/42579/pdf/01-Capitulo1.pdf.

37. **Ministerio de Relaciones Exteriores de la República de Cuba.** Informe de Cuba sobre la resolución 62/3 de la Asamblea General de las Naciones Unidas. *Cuba Minrex.* [En línea] 2006. http://embacuba.cubaminrex.cu/Default.aspx?tabid=8745.

38. **S.L., Bab Software Applications.** BabSoftware. *Diseño y desarrollo de aplicaciones web.* [En línea] 2009. http://www.bab-soft.com/es/diseno_desarrollo_aplicaciones_web.php.

39. **Prieto, Felix.** Departamento de Informática. Universidad de Valladolid. [En línea] 2009. http://www.infor.uva.es/~felix/datos/priii/tr_patrones-2x4.pdf.

40. **López, Alejandro Rivera.** UDLAP. Universidad de las Américas de Puebla. [En línea] 2008. http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/rivera_l_a/capitulo2.pdf.

41. **Peredo Valderrama, Ruben, y otros.** Arquitectura para sistemas de información basada en web usando programación orientada a componentes. [En línea] 2008. http://www.google.com.cu/url?sa=t&source=web&cd=4&ved=0CCwQFjAD&url=http%3A%2F%2Fwww2.pucpr.br%2Freol%2Findex.php%2FDIALOGO%3Fdd1%3D2033%26dd99%3Dpdf&rct=j&q=arquitectura%20orientada%20a%20componentes&ei=Ncj2Td-XGIPy0gGPvZWGCw&usg=AFQjCNESTUc8qQelUE_qnvtz.

42. **Corporation, Oracle.** Netbeans IDE. [En línea] 2010-2011. http://netbeans.org/.

43. **Información, Escuela de Ingeniería y Sistemas de.** Universidad del Valle. Colombia. [En línea] 2010. http://eisc.univalle.edu.co/materias/Material_Desarrollo_Software/Pruebas.pdf.

44. **Visconti, Marcello y Astudillo, Hernán.** Departamento de Informática. *Universidad Técnica de Federico de Santa Maria.* [En línea] 2010. http://www.inf.utfsm.cl/~visconti/ili236/Documentos/10-DisenoOO.pdf.

45. **Banco Central de Cuba**. 2008. Resolución No. 245/08. Septiembre 26, 2008.

46. **Banco Central de Cuba**. 2007. INSTRUCCIÓN No. 1/07. Octubre 31, 2007.

47. **Banco Nacional de Cuba.** 1994. RESOLUCIÓN No. 324/94. Noviembre 21, 1994.

46. **Centro de soluciones de gestión.** 2009. Ciclo de vida del Proyecto. 2009.

47. **Equipo de producción.** 2009. Modelo de Desarrollo orientado a componentes del proyecto ERP - CUBA. 2009.

48. **Leyet Fernández, Osmar.** 2010. Documento De Descripción De La Arquitectura De Software. La Habana, Cuba. 2010

49. **Ministerio de Finanzas y Precios. 2007.** Resolución No.12/2007. 2007.

50. **Ministerio de Finanzas y Precios.. 2007.** Resolución No. 14/2007. 2007.

51. **Sola, Elianys Hurtado.** 2007. ERP-ARQ Manual del marco de trabajo. 2007.

Glosario de Términos

Actividad: define un conjunto de tareas, funciones y definiciones, agrupadas bajo un mismo contexto, teniendo en cuenta la relación entre estas funciones, el lugar y el personal que las realiza, basados además en los principios del control interno.

Anticipos: dinero que se paga o cobra con anterioridad para realizar una operación de cobro o pago.

API (Application Programming Interface): interfaz de programación de aplicaciones. Es el conjunto de funciones y métodos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Base de Datos: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

Cheque: mandato de pago en el que se consigna el beneficiario y no se permiten endosos.

Conciliación Bancaria: documento donde se realiza un análisis por la entidad para determinar las causas de las diferencias existentes entre el Estado de Cuenta del Banco y el saldo en libros para llegar a determinar el saldo correcto.

CSS (Cascading Style Sheets): es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). Las hojas de estilo en cascada permiten separar la estructura de un documento de su presentación.

Cuenta Bancaria: contrato establecido por el Banco a personas jurídicas o naturales comprometiéndose a prestar los servicios que se hayan acordado del dinero que se deposite en su cuenta.

Dependencia Tecnológica: depender de una tecnología indispensable para lograr un propósito, su ausencia generaría severos problemas y colapsaría todo un sistema basado en la misma.

Documento: modelo que contiene información primaria, y refleja hechos económicos y financieros. Los tipos definidos son: documento de pago, documento de obligaciones, documento de liquidación y documento de letra de cambio.

Documento de Pago: documentos que afectan a una cuenta bancaria, aumentando o disminuyendo su saldo. En esta clasificación se incluyen los instrumentos y efectos de pago, recibo de ingresos en efectivo y otras órdenes de cobros y pagos.

DOM (Document Object Model): es un conjunto de utilidades específicamente diseñadas para manipular documentos XML o XHTML y HTML. Técnicamente, es una API de funciones que se pueden utilizar para manipular las páginas XHTML de forma rápida y eficiente.

DQL (Doctrine Query Language): lenguaje de consultas empleado por el framework doctrine para el acceso a la base de datos.

Empresa: organización o institución dedicada a actividades o persecución de fines económicos o comerciales. Sistema que interacciona con su entorno materializando una idea de forma planificada, dando satisfacción a demandas y deseos de clientes, a través de una actividad económica.

Entidad: organización administrativa, comercial, económica, productiva y de servicios de carácter estatal, cooperativa, privada o mixta, residentes en el territorio nacional; así como las organizaciones sociales y de masas del país.

Estado de Cuenta: documento o relación detallada del movimiento de una cuenta bancaria en una fecha determinada y el saldo al final del mismo.

Fichero: es una manera particular de codificar información para almacenarla en un archivo informático.

Instrumentos Bancarios: documentos generados a partir de las operaciones realizadas sobre las cuentas bancarias.

Interfaz de Usuario (UI): medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente, suelen ser fáciles de entender y fáciles de accionar.

IOC: patrón arquitectónico empleado para la relación entre clases.

JSON: acrónimo de *JavaScript Object Notation*, formato ligero para el intercambio de datos entre la capa de presentación y las demás capas de la arquitectura. Subconjunto de la notación literal de objetos de Java Script que no requiere el uso de XML.

Letra de cambio: título-valor que obliga a pagar una deuda a su vencimiento en un lugar determinado a favor de quien resulte su legítimo tenedor, se ajusta a las formalidades que establece la ley.

Liquidaciones: pago completo de una deuda o cuenta.

Multimoneda: utilización de varias monedas en las transacciones económicas y su registro; funcionalidad que supone la existencia de una moneda base y de varias monedas que tienen convertibilidad con relación a la moneda base.

Open Source: código fuente disponible públicamente.

Operación: operación a toda acción sobre el documento que genera asientos contables y que transfiere al documento de un estado a otro.

Operaciones Bancarias Automáticas: operaciones o transacciones realizadas por el banco ya sean por solicitud propia de la entidad o por operaciones que incluya el banco.

ORM (Object Relational Mapping): es una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional. El

mapeo objeto-relacional posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo).

Período contable: intervalo de tiempo en que serán registrados los hechos económicos acaecidos en una entidad. El período contable puede ser un día, una semana, un mes o un año.

Reembolsos: devolución de una cantidad de dinero a la persona o entidad que la ha pagado con anterioridad.

Talonario: cuadernillo con el fin de mantener el control numérico y consecutivo en el uso de los documentos primarios, al emitir un pago.

Transferencia bancaria: el banco la realiza siguiendo instrucciones de su cliente, debitando la cuenta del cliente por la cantidad objeto de la transferencia y acredita la cuenta del beneficiario.

XSLT: lenguaje para transformar documentos XML en otros documentos XML. Diseñado para uso como parte de XSL que es un lenguaje de hoja de estilo para XML.