Universidad de las Ciencias Informáticas

Facultad 3

DISEÑO E IMPLEMENTACIÓN DE LA VERSIÓN 1.5 DEL MÓDULO CAJA, PERTENECIENTE AL SUBSISTEMA FINANZAS DEL SISTEMA DE GESTIÓN INTEGRAL DE ENTIDADES CEDRUX.

Trabajo de Diploma para optar por el título de ingeniero informático

Autor: Yoan Pérez de Ordaz Valdés

Tutores: Ing. Rigoberto Martínez Álvarez

Ing. Enrique Chaviano Gómez

Ciudad de la Habana, __ de _____ del 2011.

"Año 53 de la Revolución"

"A veces sentimos que lo que hacemos es tan sólo una gota en el mar pero el mar sería menos si le faltara esa gota"

Teresa de Calcuta

DECLARACIÓN DE AUTORÍA

Yo Yoan Pérez de Ordaz Valdés declaro que soy el único autor de este trabajo y autorizo a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Yoan Pérez de Ordaz Valdés

_____

Autor

Ing. Rigoberto Martínez Álvarez

_____

Tutor

Ing. Enrique Chaviano Gómez

_____

Tutor

A mis abuelos, sin ellos me llamaría igual pero no sería ni la mitad de lo que soy.

A mi madre, encargada de darme impulso cuando la pendiente era demasiado inclinada para mis fuerzas

A mi tío Denis, por ser tío y padre todos estos años de carrera.

A mi padre, que desde lejos hizo suyo mis logros con sus atenciones.

A mis hermanos en la fe, tanto los de aquí como los de mi Iglesia, por guardar cada paso mío como si fueran los suyos.

Andrés, Arianny y Osvaldo: gracias por traer a la UCI un pedazo de Guayos, incluyendo apoyo incondicional y largas noches.

Mis magníficos tutores -ellos fueron más de lo que merecía-, a Kenny y al Chino y a los muchachos de la línea, en especial a Driggs, por ser mi apoyo en tiempo de dudas.

Si hay algo digno de reconocer en esta tesis, los méritos y la alabanza sea para Dios, del cual dependieron todos mis logros en la Universidad. A Él sea la Gloria. Amén

Dedicada a mi hermano Emanuel: el último que llegó siempre es el más querido. Pensando en ser tu ejemplo son todos mis esfuerzos. Tú eres el mejor.

RESUMEN

En los últimos años, con el desarrollo de la contabilidad financiera dentro de la informática con el fin de agilizar la gestión de los procesos administrativos de las empresas a nivel mundial, han surgido infinidad de programas con este fin. No obstante a pesar de la gran variedad de estos a nivel nacional e internacional, no se ha logrado obtener un sistema estándar para beneficiar la gestión contable en las empresas cubanas.

El área de la gestión financiera abarca los subsistemas de Cobros y Pagos, Banco y Caja. Este último necesita de una gran cantidad de operaciones para realizar movimientos, por lo que el proceso se vuelve engorroso a la hora del trabajo manual. Incluso no todas las entidades nacionales realizan los procesos de Caja por el mismo método, ni de la misma forma, resultando aún más complicado encontrar una solución genérica que gestione todos sus procesos.

La siguiente investigación tiene como objetivo proponer una solución a lo anteriormente planteado, para lo cual se llevará a cabo, en el primer capítulo, una investigación acerca de un grupo de sistemas de gestión integral, tanto nacionales como extranjeros, que llevan a cabo la automatización, entre otros, de los procesos de caja. Se identificarán las ventajas y desventajas de cada uno en función de lograr una mejor adaptación de sus logros al sistema económico cubano.

Se realizará un estudio de cada una de las herramientas, lenguajes, técnicas y metodologías utilizadas en la búsqueda de una posible solución al problema planteado anteriormente.

En el segundo capítulo, se abordarán los detalles de la solución correspondientes a la nueva versión, partiendo de la validación del diseño propuesto mediante la aplicación de un conjunto de métricas. Se describirán los nuevos procesos a implementar en el sistema así como las principales funcionalidades del módulo, la arquitectura propuesta para su implementación y los patrones utilizados con el fin de lograr una solución genérica que pueda ser utilizada en todas las entidades nacionales.

El tercer capítulo y final abordara los aspectos relacionados a la validación de la solución, en busca de asegurar la calidad del sistema a partir de las pruebas realizadas.

Índice

INTRODUCCIÓN

En nuestro país con el objetivo de planificar de manera eficiente los procesos contables de las empresas, así como los recursos de las mismas, se le da la tarea a la Universidad de las Ciencias Informáticas (UCI) de desarrollar un sistema de gestión integral basado en la soberanía tecnológica. En este momento se crea el proyecto ERP que es el encargado de realizar la versión inicial de Cedrux, también conocido como el ERP Cubano. El mismo cuenta con una gran cantidad de subsistemas que gestionan la contabilidad, los costos, los inventarios, el capital humano y los procesos de Banco, Cobros y Pagos y Caja.

No obstante, a pesar de que el sistema tuvo gran aceptación por los usuarios, no todas las necesidades de las empresas fueron satisfechas. En el caso específico del subsistema Caja, a pesar de haber logrado implementar de manera eficiente el método de Reembolso a partir de los fondos de Entrada, Salida y Viajes, no permitía realizar movimientos por el método del Fondo Fijo, provocando que no se pudieran realizar operaciones de entradas y salidas sobre un mismo fondo. Además no se tuvo en cuenta la utilización de un fondo que gestionara los pagos planificados, el cual permitiría un movimiento de entrada a la caja de los salarios de los trabajadores, para luego de efectuado el pago, se liquidara el efectivo entregado contra la cuenta de gastos por salarios de la entidad y el resto se pasara al fondo fijo como parte de dicho proceso.

Al no encontrarse estas características en la solución inicial se afectó la reusabilidad y escalabilidad del módulo, así como su idoneidad para la automatización de los procesos contables de la caja en las entidades nacionales.

De tal situación surge el problema científico siguiente: ¿Cómo solucionar la falta de escalabilidad y reusabilidad del subsistema Caja permitiendo realizar operaciones por el método de Fondo Fijo de modo que se ajuste a las necesidades generales de cada entidad? El objeto de estudio en la investigación serán los procesos contables del Fondo Fijo de Caja en los sistemas de gestión, y el campo de acción la implementación de los procesos contables del Fondo Fijo de la Caja en el Sistema de Gestión Integral Cedrux.

El objetivo general es la implementación de los procesos de caja por el método de Fondo Fijo, logrando la escalabilidad y reusabilidad del subsistema Caja en Cedrux. Los objetivos específicos en función de lo anteriormente planteado son:

- ✓ Realizar la fundamentación teórica.
- ✓ Implementar los nuevos requerimientos funcionales del sistema.

- ✓ Validar el diseño propuesto.

En relación con el problema planteado, trazamos como idea a defender, que la implementación de la nueva versión del subsistema Caja, mejorará la escalabilidad y reusabilidad los procesos contables Caja, permitiendo que el sistema pueda ser utilizado por cada una de las entidades nacionales.

El posible resultado es el siguiente: la obtención de una nueva versión del subsistema Caja del Sistema de Gestión Integral de Entidades Cedrux.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

El presente capítulo estará centrado en la realización de un análisis para obtener información actualizada de algunos sistemas contables que han sido implementados nacional e internacionalmente y a partir de ahí conocer qué aspectos deben ser tomados en cuenta para que el sistema esté al nivel de lo que se exige por la dirección del Ministerio de Finanzas y Precios. Se valorarán las tendencias actuales en el contexto informático partiendo de que continuamente surgen aplicaciones novedosas con mejoras que se van imponiendo en la industria del software y se especificarán las tecnologías y herramientas de desarrollo a utilizar durante la implementación para facilitar el cumplimento de los requisitos tanto técnicos como funcionales.

1.2 CONCEPTOS Y TÉRMINOS FUNDAMENTALES.

1.2.1 ERP

Los sistemas de planificación de recursos de la empresa (ERP, Enterprise Resource Planning) son sistemas de gestión de información que integran y automatizan muchas de las prácticas de negocio asociadas con los aspectos operativos o productivos de una empresa. Ellos están caracterizados por su composición en diferentes partes las cuales están integradas en una única aplicación. Estas partes son de diferente uso, por ejemplo: producción, ventas, compras, logística, contabilidad (en su variedad de formas), gestión de proyectos, inventarios y control de almacenes, pedidos, nóminas, etc. Sólo es posible definir un ERP como la integración de todas estas partes. Él integra todo lo necesario para el funcionamiento de los procesos de negocio de la empresa. No es posible hablar de ERP en el momento que tan sólo se integra uno o una pequeña parte de los procesos de negocio. La propia definición indica la necesidad de disponibilidad de toda la información para todo el mundo todo el tiempo. (Saavedra, 2011)

El propósito fundamental de un Sistema de Integración es otorgar apoyo a los clientes del negocio, tiempos rápidos de respuesta a sus problemas así como un eficiente manejo de información que permita la toma oportuna de decisiones.

1.2.2 Subsistema de Caja

Los procesos de caja permiten la gestión y control de las cajas de la empresa. Representa la existencia de medios monetarios y valores depositados en las cajas de la entidad. Comprenden entre otros: efectivo para pagos menores para cambios, fondo fijo para atenciones específicas u otros destinos, así como los importes que se

ingresan en la caja para ser depositados en las cuentas bancarias correspondientes o para pagos de nóminas. Incluyen las existencias de sellos adquiridos para uso de la entidad y los 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 o en otras instituciones financieras. Se debitan por las transferencias de efectivo a estas cuentas, al crear los fondos o al aumentarlos, así como por los cobros en efectivo pendientes de depositar en la sucursal bancaria, por los importes de los sellos comprados que se encuentran en existencia y por los cheques recibidos y se acreditan por las rebajas, utilización o cancelación de los fondos y por los depósitos efectuados en las cuentas bancarias de la entidad. (Martinez Alvarez, y otros, 2009)

1.3 ALGUNOS SISTEMAS USADOS EN LA GESTIÓN DE LOS PROCESOS CONTABLES DE CAJA EN CUBA Y EL MUNDO

1.3.1 Sistemas extranjeros

- **OpenBravo**

OpenBravo es un software desarrollado por la Universidad de Navarra en España en el año 2005, el cual es destinado principalmente para las pequeñas y medianas empresas. Las grandes áreas que integra actualmente este sistema de gestión son:

- ✓ Gestión de almacenes: almacenes y ubicaciones, unidades de almacén, lotes, números de serie, bultos, etiquetas, entradas, salidas, movimientos entre almacenes, inventarios y valoración de existencias y transportes.
- ✓ Gestión de proyectos: proyectos, fases, presupuestos, gastos y las compras asociadas. Gestión de servicios: recursos, servicios y los gastos.
- ✓ Gestión comercial y gestión de las relaciones con clientes (CRM): pedidos de venta, tarifas, albaranes, facturación y CRM.
- ✓ Gestión económico-financiera: plan de cuentas, cuentas contables, impuestos, contabilidad general, cuentas a pagar, cuentas a cobrar, contabilidad bancaria, balance, cuenta de resultados, y activos fijos.

OpenBravo se caracteriza por ser una aplicación completamente web que ha sido desarrollada siguiendo el modelo MVC (en inglés Model-View-Controler) que facilita el desacoplamiento de las áreas de desarrollo. Implementada en el lenguaje Java, presenta soporte para bases de datos PostgreSQL y Oracle, se ejecuta sobre Apache y Tomcat. La estructura de datos de la aplicación está basada originalmente en una

versión antigua de Compiere[1]. Una de las principales ventajas con las que cuenta este ERP es que sigue un licenciamiento de software libre asegurando el acceso público al código fuente y la posibilidad de modificar dicho código libremente, adaptándolo a las necesidades de cualquier empresa. (OpenBravo, 2007)

OpenBravo se describe a sí mismo como el ERP de código abierto más destacado y sencillo que existe hasta el momento. Es una de las aplicaciones que gestiona de manera eficiente los datos económicos basándose en los registros contables de las empresas según los centros de costos, sin embargo, el proceso de Caja se encuentra disperso en el módulo Gestión Financiera, compartido entre los componentes de la Contabilidad General y las Cuentas por pagar y por cobrar.

- **SAP**

Fue desarrollado en la Ciudad de Mannheim, Alemania, por antiguos empleados de IBM. SAP puede configurar cada uno de los procesos según las necesidades de las empresas con el fin de obtener una ventaja competitiva en su sector. Éste software está diseñado para incrementar la eficiencia de la planificación y la gestión de procesos a lo largo de las empresas. Ofrece dos conjuntos de funcionalidades: uno centrado en el soporte a las operaciones y otro centrado en la generación de valor. El primero permite gestionar operaciones logísticas, satisfacer requisitos de calidad y cumplir con la normativa y estándares de su industria. También cuenta con las herramientas para el desarrollo y la introducción de nuevos productos con cobertura para el ciclo de vida completo del producto. El segundo conjunto de funcionalidades está diseñado para incrementar la eficiencia en los procesos de producción. Permite mejorar la totalidad de las operaciones logísticas relacionadas con la creación y mejora de sus productos y servicios.

SAP da soporte a procesos de negocios globales en las áreas de Finanzas, Desarrollo de productos y Ventas y Servicio. Entre sus áreas funcionales podemos encontrar:

- ✓ Contabilidad general.
- ✓ Presupuesto.
- ✓ Bancos.
- ✓ Informes financieros.

Se caracteriza por ser un sistema totalmente preparado para trabajar con él mediante la web debido la nueva plataforma tecnológica denominada SAP NetWeaver5. Implementado en .NET. Funciona sobre el sistema operativo Windows y presenta soporte para bases de datos Oracle.

[1] **Compiere**: Aplicación para negocios de código abierto, ERP y CRM destinada para las empresas de pequeño y mediano tamaño.

1.3.2 Sistemas nacionales

- **RODAS XXI**

Es un Sistema multiempresa y multiusuario creado por la empresa cubana CITMATEL para la automatización de la gestión empresarial. Contiene diferentes módulos que pueden usarse integrados o independientes. RODAS XXI usa como gestor de base de datos SQL de Microsoft, lo que hace menos viable al tener que pagar licencias por su utilización. Dentro de dichos módulos se encuentra Finanzas, donde se ven reflejadas funcionalidades orientadas al proceso de caja, tales como reembolsos para pagos menores, registro de ingresos, control de las dietas, y la realización de arqueos

- **Versat-Sarasola**

Software desarrollado por la entidad cubana TEICO de Villa Clara, constituye un sistema que automatiza prácticamente todas las actividades de Planificación, Control y Análisis Económico de cualquier tipo de Entidad, ya que es configurable. Presenta dos variantes para su instalación: la variante típica que incluye todos los subsistemas del VERSAT-Sarasola y la variante personalizada en la que solo estarán los subsistemas seleccionados por el usuario. Permite llevar el control y el registro contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades y obtener los Estados Financieros y Análisis Económicos y Financieros en estos niveles. Aparece el Subsistema de Contabilidad General como rector del sistema que maneja el resto de los subsistemas:

- ✓ Costos y Procesos
- ✓ Cobros y pagos
- ✓ Activos Fijos
- ✓ Finanzas
- ✓ Banco

Actualmente lo utilizan más de 200 entidades de varias provincias en todo el país, entre las que figuran organismos de la Administración Central del Estado, las direcciones municipales de finanzas, tesorerías, la ONAT y otros. Dicho sistema requiere como sistema operativo Windows, por lo que no es multiplataforma, reduciendo de esta manera su viabilidad en nuestro país. En su módulo de finanzas el flujo de caja está integrado con algunos de los demás subsistemas del mismo, pero no tiene funcionalidades tales como arqueo de fondos, lo que hace complejo la evaluación al listado de movimientos de los fondos en un período de tiempo.

- **Subsistema actual de Caja en el Sistema de Gestión Cedrux**

Cedrux asume como características propias las principales desventajas que presentan los sistemas antes mencionados -es un sistema multiplataforma, completamente web,

implementado mediante software libre, tiene en cuenta la integración de sus componentes, entre otras-, en vistas a ser la alternativa cubana a la necesidad de una herramienta estándar que agilice los procesos empresariales. A pesar de esto, particularmente su módulo Caja, como fue expresado anteriormente, no contempla todos los **métodos de Caja,** afectando su escalabilidad como sistema.

1.3.3 Valoración del estado del arte.

Después de realizar un estudio a los sistemas contables antes mencionados donde se tuvieron en cuenta no sólo los aspectos fundamentales desde el punto de vista del software sino también del producto, *se evidencia* que los sistemas extranjeros *no son compatibles con las políticas de la economía cubana,* además en algunos casos requieren del pago de *costosas licencias para su utilización.* En el caso de los sistemas nacionales *se puede apreciar la no existencia de un software que responda cabalmente a lo que en realidad se necesita para la gestión eficaz y eficiente de los procesos de Caja*, lo que imposibilita que sea utilizado en todas las entidades nacionales. Sería entonces factible en la lucha por lograr la soberanía tecnológica, *implementar* a partir de la versión inicial de Cedrux -la cual trabaja con herramientas y tecnologías que no son propietarias-, una nueva versión que gestione de manera eficiente los procesos de las cajas cubanas.

1.4 TENDENCIAS Y TECNOLOGÍAS ACTUALES USADAS

1.4.1 Modelo de desarrollo

La propuesta de modelo de desarrollo que a continuación aparece fue elaborada por el equipo de producción en colaboración con las Líneas de desarrollo del proyecto ERP-Cuba de acuerdo con las necesidades presentadas por cada una de ellas y donde se tuvieron en cuenta los principales riesgos con los que se cuentan en el proyecto. Sus características son las siguientes:

- ✓ **Centrado en la arquitectura:** La arquitectura determina la línea base y los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades en la producción y resuelve las necesidades tecnológicas y de soporte para el desarrollo.
- ✓ **Orientado a componentes:** Las iteraciones son orientadas según la significación arquitectónica de los componentes, los mismos son abstracciones

arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

- ✓ **Iterativo e incremental:** Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.
- ✓ **Ágil y adaptable al cambio:** El desarrollo de las partes formaliza solamente 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. (Vega, 2009)

1.4.2 Marco de Trabajo Sauxe

Según el concepto citado por la Ingeniera Yadira Piñera Andux "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". A la par del paradigma de independencia tecnológica que es impulsado por nuestro país, Sauxe reutiliza las siguientes tecnologías que serán descritas a continuación.

1.4.3 Frameworks

1) Zend Framework

Es un framework de alta calidad y de código abierto para el desarrollo de aplicaciones y servicios web con PHP.

Zend Framework brinda facilidades de uso y poderosas funcionalidades. Proporciona soluciones para construir modernos, robustos y seguros sitios web, está diseñado para php5 y posee buenas capacidades de ampliación. Presenta entre otras, las siguientes características:

- ✓ Proporciona un sistema de caché de forma que se puedan almacenar diferentes datos.
- ✓ Proporcionan los componentes que forma la infraestructura del patrón MVC.

- ✓ Proporciona una capa de acceso a base de datos, construida sobre PDO[2] pero ampliándola con diferentes características.
- ✓ Proporciona mecanismos de filtrado y validación de entradas de datos.
- ✓ Permite convertir estructuras de datos PHP a JSON[3] y viceversa, para su utilización en aplicaciones AJAX.
- ✓ Proporciona capacidades de búsqueda sobre documentos y contenidos.

2) Doctrine Framework

Doctrine es un potente y completo sistema ORM[4] (en inglés Object Relational Mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientada a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL manteniendo un máximo de flexibilidad sin requerir la duplicación del código innecesario. Funcionalidades:

1- Exporta una base de datos existente a sus clases correspondientes.

2- Convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. (Doctrine, 2008)

3) ExtJS

Librería construida empleando JavaScript con el fin de desarrollar aplicaciones web interactivas usando tecnologías como AJAX y DOM. Ext es muy potente ya que contiene rica colección de componentes para el diseño de Interfaces Gráficas de Usuario (GUI's) del lado del cliente haciendo uso extensivo de AJAX. Dispone de un conjunto de componentes gráficos como:

- ✓ Cuadros y áreas de texto.
- ✓ Campos para fechas.
- ✓ Campos numéricos.
- ✓ Selectores estáticos y dinámicos.
- ✓ Botones.
- ✓ Editor HTML.
- ✓ Elementos de datos (con modos de sólo lectura, datos ordenables, columnas que se pueden bloquear y arrastrar, etc.).
- ✓ Árbol de datos.
- ✓ Pestañas.
- ✓ Barra de herramientas.
- ✓ Menús al estilo de Windows.

[2] PDO: (en inglés: PHP Data Objects) es una interface de acceso a datos que permite la conexión a diferentes bases de datos utilizando tecnología orientada a objetos. (14)
[3] JSON: es un formato ligero para el intercambio de datos. (14)
[4] ORM: Componente de software que permite trabajar con los datos persistidos como si fueran parte de una base de datos orientada a objetos

1.4.4 Navegador

- **Mozilla Firefox**

Firefox es un navegador de código abierto y multiplataforma, disponible para la mayoría de los sistemas operativos actuales. Incluye la navegación por pestañas, un administrador de tareas, un sistema de búsqueda integrado mediante el motor de búsqueda escogido por el usuario. Como una de sus principales características está la de aceptar complementos creados por terceros.

Este navegador es compatible con la mayoría de los estándares web, además de prever distintos cambios en éstos. Protege su comunicación con los servidores web mediante el sistema SSL/TLS, y cuenta con una protección antipishing, antimalware e integración con el antivirus local.

1.4.5 Lenguajes de Programación para la Web

- **PHP**

PHP es un lenguaje de programación open source ampliamente utilizado, diseñado originalmente para ser usado en el desarrollo de sitios Web. Es nombrado por sus siglas en inglés, Hypertext Preprocessor (Preprocesador de Hipertexto).

Algunas de sus características que influyen en su popularidad son:

- ✓ Es rápido: en las páginas web, a causa de estar embebido en el código HTML, el tiempo de procesamiento y carga de la página web es pequeño.
- ✓ Es libre (…)
- ✓ Es fácil de usar: la sintaxis es simple y fácil de entender y usar, aún para los no programadores. Para su uso en las páginas web, el código PHP es diseñado para ser incluido fácilmente en un archivo HTML.
- ✓ Es versátil: PHP corre en una amplia variedad de sistemas operativos- Windows, Linux, MacOS, y muchas variedades de Unix.
- ✓ El apoyo técnico está ampliamente disponible: Uno se puede añadir a cualquiera de los muchas listas de discusión ofrecidas en el sitio web PHP (...)
- ✓ Es seguro: Mientras los scripts sean diseñados correctamente, el usuario no debe ver el código PHP.
- ✓ Es adaptable: la licencia open source permite a los programadores modificar el software PHP, añadiendo o modificando propuestas (...) (Valade, 2004)

- ✓ **HTML**

HTML es un lenguaje de composición de documentos y especificación de ligas de hipertexto que define la sintaxis y coloca instrucciones especiales que no muestra el

navegador, aunque si le indica cómo desplegar el contenido del documento, incluyendo texto, imágenes y otros medios soportados. HTML también indica cómo hacer un documento interactivo a través de ligas especiales de hipertexto.

PHTML es una extensión para un tipo de páginas web que llevan código PHP y HTML para ser generadas. Cuando una página está escrita en PHP podemos encontrarla con varios tipos de extensiones como por ejemplo .php, .php4, .php3, o .phtml.

- ✓ **JavaScript**

JavaScript es un lenguaje interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C. Al contrario que Java, JavaScript no es un lenguaje orientado a objetos propiamente dicho, ya que no dispone de herencia, es más bien un lenguaje basado en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM[5]. JavaScript es un lenguaje con muchas posibilidades, utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos mucho más complejos. Con JavaScript se puede crear diferentes efectos e interactuar con los usuarios. JavaScript es soportado por la mayoría de los navegadores como Internet Explorer, Netscape y Mozilla Firefox. (Territorio PC, 2001)

1.4.6 Sistemas gestores de Base de Datos

Se denomina Sistema Gestor de Base de Datos (siglas: SGBD) al conjunto de programas que permiten definir, construir y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad.

- ✓ **PostgreSQL**

PostgreSQL es un servidor de base de datos relacional, libre. Se destaca en ejecutar consultas complejas, consultas sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. Como toda herramienta de software libre PostgreSQL tiene entre otras ventajas las de contar con una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante es que es multiplataforma. Fue diseñado para ambientes de alto volumen. Escala muy bien al

[5] DOM: El Modelo de Objetos de Documento (en inglés Document Object Model), Es una plataforma que proporciona un conjunto estándar de objetos, que permite crear documentos HTML y XML, navegar por su estructura, modificar, añadir y borrar tanto elementos como contenidos. No se ajusta a un lenguaje de programación específico, lo que facilita el diseño de páginas web activas, de manera tal que proporciona una interfaz estándar para que otro software manipule los documentos.

aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones y desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas procedimientos almacenados en el servidor, transacciones, almacenamiento de objetos de gran tamaño y además tiene ciertas características orientadas a objetos.

1.4.7 Tecnología AJAX

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. Permite realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. AJAX es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting lenguaje) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. (Abarta, 2006)

1.4.8 Herramientas de Desarrollo

- ✓ **Subversion**

También conocido como SVN, es un sistema de control de versiones que se ha popularizado bastante, en especial dentro de la comunidad de desarrolladores de software libre. Está preparado para funcionar en red y se distribuye bajo licencia libre.

- ✓ Mantiene versiones no sólo de archivos, sino también de directorios
- ✓ Mantienen versiones de los metadatos asociados a los directorios.
- ✓ Además de los cambios en el contenido de los documentos, se mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- ✓ Atomicidad de las actualizaciones, una lista de cambios constituye una única transacción o actualización del repositorio, esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- ✓ Soporte tanto de ficheros de texto como de binarios.
- ✓ Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos

- ✓ **Zend Studio para Eclipse**

Es un IDE (Entorno integrado de desarrollo) para el lenguaje de PHP escrito en Java destinado a desarrolladores profesionales, propietario, compatible con las plataformas Linux, MAC y Windows. Incluye todos los componentes necesarios durante el ciclo de vida de una aplicación en PHP. Incluye editor, análisis, depuración, optimizadores de código y herramientas de base de datos. Zend Studio permite agilizar el desarrollo web y permite simplificar proyectos complejos. Posibilita un excelente completamiento de código, coloreado en la sintaxis del código, administración avanzada de proyectos, múltiples lenguajes, incorpora el Framework de Zend, integración con subversión, integración avanzada con FTP, soporte para Web Services, PHP4 y PHP5, entre otras características están:

- ✓ No requiere la instalación previa de PHP ni del entorno de ejecución de Java.
- ✓ Detección de errores de sintaxis en tiempo real.
- ✓ Manual de PHP integrado.
- ✓ Ofrece soporte básico para otros lenguajes web, como HTML, JavaScript y XML.
- ✓ Acceso al paquete de plug-ins de Eclipse.
- ✓ Mecanismo de actualización automática. (Alonso, 2001)

- ➢ **Visual Paradigm**

Visual Paradigm es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Entre sus principales características podemos encontrar:

- ✓ Soporta aplicaciones Web.
- ✓ Es un producto de calidad.
- ✓ Fácil de instalar y actualizar.
- ✓ Compatibilidad entre ediciones. (Visual, 205)

- ➢ **Servidor web Apache**

Es una tecnología gratuita de código fuente abierta, puede ser usado en varios sistemas operativos, lo que lo hace prácticamente universal. Es un servidor altamente configurable es decir se pueden elegir qué características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor. (Apache, 2009)

1.5 CONCLUSIONES

Fueron definidos primeramente conceptos claves para el entendimiento de los propósitos planteados en este documento. Las investigaciones realizadas en este capítulo referente al estado del arte, de la implementación de los Sistemas de Gestión fueron fundamentales para concluir que la soberanía tecnológica, su adaptabilidad a las características de la economía cubana y la lograda integración entre sus componentes, hacen de Cedrux el sistema ideal sobre el cual implementar la nueva versión del modulo caja, por delante de las otras versiones estudiadas en esta sección.

También se estudio las tecnologías definidas por el marco de trabajo, que anteriormente habían logrado sus propósitos al ser escogidas por la dirección del proyecto en la anterior versión.

CAPÍTULO II. DISEÑO E IMPLEMENTACIÓN.

2.1 INTRODUCCIÓN.

El siguiente capítulo abarca el desarrollo de elementos fundamentales de la implementación de la solución, encontrándose primeramente una valoración del diseño propuesto y una explicación de la arquitectura general del subsistema. A continuación, se detallará los procesos que se requieren para implementar la solución, incluyendo una descripción de las clases y operaciones más relevantes.

2.2 DISEÑO DE LA SOLUCIÓN

2.2.1 Valoración crítica del diseño propuesto por el analista.

El artefacto Especificación de Requisitos del Software para el Subsistema Caja constituye un elemento clave para el diseño y la posterior implementación. Se caracteriza por presentar los requisitos de forma completa, definiendo todas las responsabilidades del sistema con relación a los datos de entrada, válidos o no y respecto a los datos de salida. Presenta una adecuada organización y documentación; los términos, las tablas y los prototipos en él comprendidos están correctamente descritos y referenciados. Cada requisito que comprende tiene una única interpretación evitando la ambigüedad en las definiciones y funcionalidades, están clasificados por la importancia arquitectónica y desde el punto de vista del cliente. Viabiliza las modificaciones, la comprobación y la trazabilidad (origen- implementación) de los requisitos especificados.

2.2.2 Descripción de las funcionalidades de los componentes.

De forma general, la descripción del funcionamiento del componente creado es la siguiente. El usuario realiza una petición en el navegador, la cual genera un evento mediante un formulario JS, que envía una notificación a la clase controladora mediante AJAX. El controlador entonces es el encargado de decidir quién va a ser el responsable de obtener los datos de la petición, una clase Bussines si la petición realizada inserta modifica o elimina (CRUD) o una clase Domain si lo que se desea es consultar, esta última es mapeada por el ORM Doctrine y se conecta mediante PDO (Doctrine Record) a la DBO (Capa interna de doctrine basada en PDO nativo) y la DBO es la que se conecta directamente a la base de datos. La clase controladora es la encargada de la lógica propia del negocio, cálculos y procesamientos

A continuación se describirán los componentes que están incluidos en el módulo Caja junto con cada una de sus respectivas funcionalidades y se listarán los requisitos funcionales identificados por el analista. (Más detalles en el Anexo 3 Diagrama de Componentes del módulo, donde podrán visualizarse las integraciones entre las funcionalidades del modulo)

- **Componente Configuraciones**

Este es el componente encargado de realizar la nomenclatura de los elementos fundamentales en el funcionamiento de los fondos que disponga el módulo. La funcionalidad que más transformaciones sufrió en interés de la nueva versión fue **Tipo de Fondo** (encargada, como su nombre lo dice de gestionar los diferentes tipos de fondos existentes en la Caja), a la cual añadiéndosele la opción de crear el **tipo de fondo para Caja (Fondo Fijo)**, el cual, como característica fundamental presenta la propiedad de admitir todos las operaciones contables de entrada y salida. **Gestionar Fondo**, **Gestionar Caja** y **Gestionar Tipo de Movimiento** son el resto de las funcionalidades que completan el componente, en el caso de Gestionar Fondo, se agregó la particularidad de los fondos de pagos planificados.

- **Componente Movimientos**

Con la mayor cantidad de funcionalidades dentro del módulo, el componente Funcionalidades (Movimientos) es el encargado de dar solución a la mayoría de los procesos de Caja. Está compuesto por las funcionalidades **Anticipos (nacional y exterior), Depósito, Liquidación de Anticipos, Reembolso y Movimientos Genéricos** -estas fueron adaptadas a los nuevos requerimientos propios del negocio de la presente versión-. Los mayores cambios se hicieron en los Anticipos, a los cuales se le añadieron algunas particularidades propias del negocio y en Movimiento Genérico, la cual cambió casi completamente con el objetivo de permitir que a un mismo fondo se le pudieran realizar operaciones de entrada y salidas.

Fueron añadidas en la nueva versión, ***Completitud de Fondo Fijo*** **para Caja**, sólo disponible para los fondos de nuevo tipo, ya que ellos no realizan el movimiento de reembolso como estaba predefinido anteriormente y ***Movimiento entre Cajas*****, la cual se encarga de realizar todos los movimientos de los pagos planificados.**

- **Componente Arqueos**

La única funcionalidad comprendida en este componente (**Realizar Arqueo**[6]) es la encargada de realizar el arqueo de los fondos. Esta es una de las funcionalidades más complejas y más útiles del trabajo con la caja pues permite al usuario realizar un cuadre de Caja (verifica que el saldo del fondo coincida con el de la cuenta del fondo en contabilidad) en cualquier momento, siendo una funcionalidad muy utilizada para las auditoría y para el control en los cuadres diarios.

- **Identificación de los requisitos funcionales**

Los requisitos funcionales que fueron identificados y aprobados en el artefacto del que se hace mención en el epígrafe anterior, fueron agrupados en un conjunto de funcionalidades con un objetivo en común.

1. Gestionar tipo de fondo.
    a. Adicionar tipo de fondo.
    b. Modificar tipo de fondo.
    c. Eliminar tipo de fondo.
    d. Listar tipo de fondo
2. Gestionar fondo
    a. Adicionar fondo.
    b. Modificar fondo.
    c. Eliminar fondo.
    d. Listar fondo
3. Gestionar caja
    a. Adicionar caja.
    b. Modificar caja.

[6] El Arqueo de Caja consiste en el análisis de las transacciones del efectivo, durante un lapso determinado, con el objeto de comprobar si se ha contabilizado todo el efectivo recibido y por tanto el Saldo que arroja esta cuenta, corresponde con lo que se encuentra físicamente en Caja en dinero efectivo, cheques o vales. Sirve también para saber si los controles internos se están llevando adecuadamente.

c. Eliminar caja.

d. Asociar/desasociar fondo

e. Apertura de fondo

f. Asignar talonario al fondo

g. Listar caja

4. Gestionar anticipo de dieta para viaje al exterior

a. Adicionar anticipo de dieta para viaje al exterior

b. Modificar anticipo de dieta para viaje al exterior

c. Cancelar anticipo de dieta para viaje al exterior

d. Confirmar anticipo de dieta para viaje al exterior

e. Contabilizar anticipo de dieta para viaje al exterior

f. Listar anticipo de dieta para viaje al exterior

5. Liquidar anticipo de dieta para viaje al exterior

a. Liquidar anticipo de dieta para viaje al exterior

b. Listar anticipo de dieta para viaje al exterior liquidados

6. Realizar depósito.

a. Adicionar depósito

b. Cancelar depósito

c. Confirmar depósito

d. Contabilizar depósito

7. Realizar reembolso

a. Adicionar reembolso

   b. Cancelar reembolso
   c. Contabilizar reembolso

8. Gestionar anticipo de dieta
   a. Adicionar anticipo de dieta.
   b. Modificar anticipo de dieta.
   c. Cancelar anticipo de dieta.
   d. Confirmar anticipo de dieta.
   e. Contabilizar anticipo de dieta.
   f. Listar anticipo de dieta.
9. Liquidar anticipo de dieta
   a. Liquidar anticipo de dieta.
   b. Listar anticipos de dieta liquidados.
10. Realizar movimiento genérico.
   a. Adicionar movimiento genérico
   b. Modificar movimiento genérico
   c. Cancelar movimiento genérico
   d. Confirmar movimiento genérico
   e. Contabilizar movimiento genérico
11. Completitud de fondo fijo para Caja
   a. Realizar completitud
   b. Listar completitudes realizadas
12. Realizar arqueo

a. Adicionar arqueo

b. Modificar arqueo

c. Cancelar arqueo

d. Contabilizar arqueo

e. Cancelar diferencia

f. Listar arqueos realizados

2.2.3 Arquitectura del diseño del módulo.

"La Arquitectura del Software es la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución" (Desarrollo, 2010). En la medida que sea concebida la arquitectura basada en los principios de cohesión, utilidad y flexibilidad de los componentes se obtendrá un mejor acabado del producto, y será la calidad final y la utilidad del software.

La arquitectura de software se selecciona y diseña con base en objetivos y restricciones, pues es el diseño de más alto nivel de la estructura de un sistema. Sobre ella se sustentan todos los mecanismos de diseño y representaciones de la estructura general de la aplicación a desarrollar. De la cohesión, utilidad y flexibilidad de los componentes de la arquitectura dependerán la calidad final y la utilidad del software.

La arquitectura de una aplicación comprende la vista conceptual de la misma; en este marco se establecen los fundamentos en los que analistas, diseñadores y programadores trabajan en una línea común con la meta de alcanzar los objetivos del sistema en relación con las necesidades del cliente.

En la solución del módulo Caja se adoptó la misma arquitectura definida en el proyecto CEDRUX, con el objetivo de que el mismo forme parte del subsistema Finanzas.

- **Orientado a Componentes**

Con la introducción de sistemas concurrentes, independientemente extensibles, con componentes heterogéneos que pueden ingresar o abandonar el sistema y que los mismos poseen una vida más corta que la del sistema en sí, la Programación Orientada a Objetos se muestra insuficiente en sus técnicas en el desarrollo de aplicaciones de este

tipo. A partir de ello nace la Programación Orientada Componentes, como extensión natural de la anterior para los sistemas creados en entornos abiertos. Este paradigma propugna el desarrollo y utilización de componentes reutilizables dentro del posible mercado de software.

Con la utilización de esta metodología de desarrollo, la implementación de cualquier componente del Sistema de Gestión será más sencillo y en un tiempo y con un coste menores y se disminuirá la posibilidad de errores a un grado mínimo puesto que éstos son sometidos a un control de calidad de forma independiente al producto en general.

En el Sistema de Gestión Cedrux, se utilizó dentro de cada componente el patrón arquitectónico Modelo-Vista-Controlador, explicado a continuación.

- **Patrón(o estilo) Modelo-Vista-Controlador**

El patrón Modelo Vista Controlador MVC separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, permitiendo mayor independencia, mantenimiento y reutilización de los mismos.

- ✓ Modelo: representa los datos del programa y controla todas sus transformaciones.
- ✓ Vista: genera la presentación visual de los datos representados por el Modelo y muestra los datos al usuario.
- ✓ Controlador: maneja las entradas del usuario, actuando sobre los datos representados por el Modelo.

Algunos de los beneficios que brinda este patrón, según Febe Ángel Ciudad son los siguientes:

1. Menor acoplamiento.
2. Desacopla las vistas de los modelos.
3. Desacopla los modelos de la forma en que se muestran e ingresan los datos.
4. Mayor cohesión.
    a. Cada elemento del patrón está altamente especializado en su tarea (la vista en mostrar datos al usuario, el controlador en las entradas y el modelo en su objetivo de negocio).
5. Las vistas proveen mayor flexibilidad y agilidad.

a. Se puede crear múltiples vistas de un modelo.
b. Las vistas pueden anidarse.
c. Se puede cambiar el modo en que una vista responde al usuario sin cambiar su representación visual.
d. Se puede sincronizar las vistas.
e. Las vistas pueden concentrarse en diferentes aspectos del modelo.

6. Mayor facilidad para el desarrollo de clientes ricos en múltiples dispositivos y canales
    a. Una vista para cada dispositivo que puede variar según sus capacidades.
    b. Una vista para la Web y otra para aplicaciones de escritorio.
7. Más claridad de diseño.
8. Facilita el mantenimiento.
9. Mayor escalabilidad.

2.2.4 Patrones utilizados en el diseño

El diseño fue elaborado siguiendo patrones basados en la experiencia, que de manera general constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. En este caso se emplearon los patrones GRASP (en inglés General Responsability Assignment Software Patterns), los que describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Los patrones GRASP que se utilizaron son los siguientes:

Experto: Dicho patrón es evidenciado en la definición de las clases de acuerdo con las funcionalidades que deben realizar a partir de la información manejada dentro del componente, como por ejemplo las clases controladoras y las del modelo.

Creador: Este patrón es adaptable a las clases del paquete Domain, quienes son las encargadas de crear los objetos de tipo Doctrine_Query, para permitir el acceso a la información almacenada a nivel de datos.

Alta cohesión: Este patrón fue utilizado en el diseño del componente de manera general; donde se agruparon las clases en dependencia de los requerimientos a los que se les

debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Durante el diseño del componente se emplearon patrones GOF, específicamente:

Fachada: La aplicación de este patrón en el componente Caja se evidencia en la interfaz de servicios simple que se proporciona para establecer la comunicación con otros componentes dentro y fuera del Subsistema.

Mediador: La comunicación en la base de datos se puede tornar compleja debido a las dependencias marcadas entre las tablas que la componen, esto resulta engorroso a la hora de acceder a un determinado valor. La solución a este inconveniente viene dada por la utilización de este patrón; específicamente; creando una nueva tabla entre todas las tablas unidas mediante una relación de muchos a muchos. La tabla mediadora posee una relación de uno a muchos con las vinculadas a ella. De esta forma, el comportamiento distribuido entre las clases queda adaptado a las circunstancias y necesidades del diseño.

Cadena de Responsabilidad: Está concebido que ante la ocurrencia de un error al realizarse una determinada consulta a la base de datos el mismo sea manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error y sin especificar detalles del mismo. De esta manera, se distribuyen las responsabilidades entre las diferentes componentes, evidenciándose por lo tanto el empleo de este patrón.

La Arquitectura Base y el diseño flexible y escalable a través del correcto uso de patrones de diseño en la generación de los artefactos necesarios para el desarrollo, posibilitaron crear una entrada apropiada como punto de partida a las actividades de implementación, con la máxima de lograr una mayor calidad del producto y la satisfacción del cliente.

2.2.5 Estrategia de Integración entre componentes

En cada uno de los componentes del sistema el flujo de datos que va desde la vista hacia el modelo y viceversa, responde completamente a una estrategia de integración vertical concebida sobre 4 nodos y a partir de cada uno de los elementos arquitectónicos definidos.

El primer nodo se sitúa entre la vista y el controlador, el segundo está entre el controlador y el modelo, el tercero vincula el modelo con el framework doctrine y el último se encuentra entre la base de datos y el doctrine.

Vista – Controlador: Los datos recogidos en un formulario son enviados al Controlador haciendo uso del protocolo de comunicación HTTP a través del método "post" para ser procesados y los resultados son enviados por el controlador a la vista en un JSON a través del método "echo".

Controlador – Modelo: El Controlador toma los datos recibidos desde la vista, instancia una determinada clase del modelo y llama a uno de sus métodos, pasándole como parámetros los datos recibidos.

Modelo – Doctrine: El Modelo utiliza llamadas a métodos de Doctrine que le permitan crear, modificar, eliminar o actualizar los datos almacenados en las tuplas de la base de datos.

Doctrine – Base de Datos: Doctrine ejecuta las consultas a la Base de Datos utilizando programación orientada a objetos.

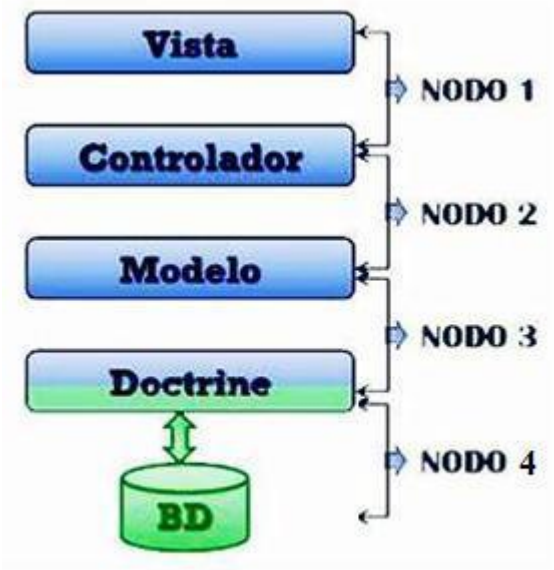


Ilustración 1 Nodos involucrados en la integración.

La comunicación dentro de un mismo componente se ejecuta de forma directa, sin embargo, la que se establece entre los diferentes componentes y subsistemas de la aplicación va mas allá de un simple llamado a un servicio, esta se basa en el empleo de

un registro de datos de los subsistemas contenidos en un fichero xml mapeado por el framework para el funcionamiento del componente llamado: inversión de control (IoC). El IoC registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema, especifica respuestas deseadas a sucesos o solicitudes de datos concretas, orden necesario y el conjunto de sucesos que tienen que ocurrir según los parámetros requeridos para poder hacer uso de un determinado servicio.

2.2 IMPLEMENTACIÓN DE LA SOLUCIÓN.

2.3.1 Estructura del Marco de Trabajo

Cumpliendo con las decisiones arquitectónicas de la dirección del proyecto y del equipo de arquitectura se define una estructura contenedora donde van a estar ubicados cada uno de los subsistemas implementados dentro de la aplicación, de manera que facilite la organización y claridad durante el desarrollo, la especificación que a continuación se presenta está enfocada al componente Caja del Subsistema Finanzas.

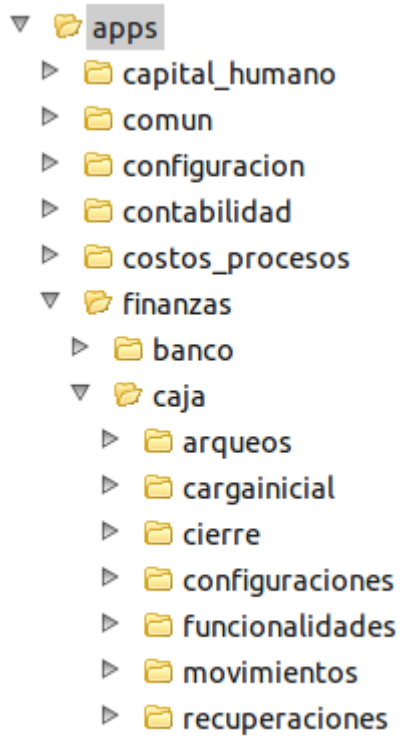


Ilustración 2: Módulo Caja en la carpeta Apps.

App: es la carpeta donde se almacenarán los controladores y el modelo de cada uno de los componentes que le corresponden al los subsistemas.

Común: En el está comprendido todo lo referente a la configuración específica del subsistema. Él contiene en sí la carpeta **recursos**, la cual a su vez guarda otra denominada **xml.** Ésta incluye los ficheros explicados a continuación.

- ✓ **Ioc:** En él están publicados los servicios que brindan cada uno de los componentes del Subsistema.
- ✓ **Validator**: Chequea las precondiciones antes de ejecutar una determinada función en el servidor, en dependencia del tipo de parámetro, la acción y el usuario que la ha realizado
- ✓ **Exepction**: Se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción en el servidor.

Dentro del módulo cada componente cuenta con una carpeta la cual se estructura de la siguiente forma:

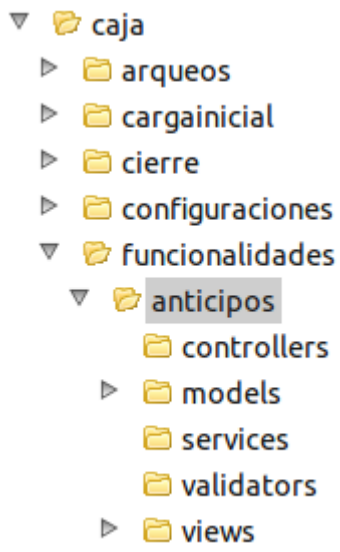


Ilustración 3: Paquete correspondiente a la funcionalidad Anticipo.

Controllers: En esta carpeta se almacenarán las clases controladoras encargadas de la gestión de las funcionalidades del componente y el flujo de información entre las vistas y las modelos. Su responsabilidad principal es comunicar las interfaces de usuario con la lógica de negocio de la aplicación.

Models: En dicha carpeta se programa toda la lógica de negocio del componente en cuestión y se gestiona lo referente a la información física de la base de datos que se archiva en las carpetas **bussines** y **domain**.

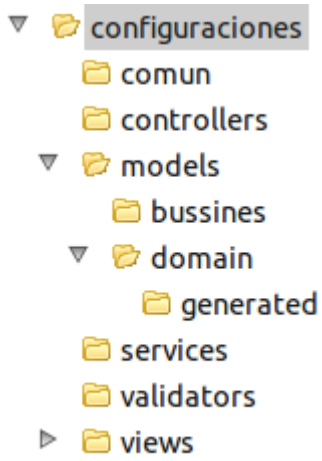


Ilustración 4: Formato de la carpeta models en la funcionalidad Configuraciones.

- ✓ **Bussines**: En ella se generan las clases modelo, se programa toda la lógica de negocio y las acciones de modificación que se van a realizar con determinada entidad de la base de datos, como insertar, actualizar y eliminar.
- ✓ **Domain**: Se guardan archivos generados por el framework Doctrine; esto incluye las clases encargadas de gestionar las consultas a las tablas de la base de datos. Estas clases heredan de ficheros base definidos como clases php que tienen el mismo nombre de las tablas y se ubican en la subcarpeta generated.

2.2.2 Estándares de código.

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Los estándares de codificación en el marco del proyecto CEDRUX van a permitir una mejor integración entre las líneas de producción y se establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su capacidad de mantenimiento a lo largo del tiempo. Nuestro módulo se rige por estos estándares, pues son una guía para el desarrollo y desde el punto de vista arquitectónico estandarizan el código a implementar.
Se definen al menos tres partes principales dentro de un estándar de programación:

- ✓ Convención de nomenclatura o la forma de nombrar las variables y funciones a utilizar.
- ✓ Convención de legibilidad de código: Forma de organizar el código
- ✓ Convención de documentación: Cómo se establecen los comentarios, ayudas, etc.

2.3.3 Estructura de datos a utilizar.

Se define como estructura de datos en programación, a la forma de organizar un conjunto de datos elementales con el fin de facilitar su manipulación. Un dato elemental es la mínima información que se tiene en un sistema. Existen diversas estructuras que en sí poseen ventajas y desventajas según la simplicidad y eficiencia para la realización de cada operación. Sin embargo, a la hora de elegir la estructura de datos más conveniente es preciso evaluar factores básicos como la frecuencia y el orden en que se realiza cada operación sobre los datos. Durante la realización de una sintaxis de código en PHP; lenguaje definido para la implementación de los diferentes componentes de la aplicación debido a las facilidades que brinda; aparecen variables que tienen información similar y que se procesan de forma semejante, concretamente, se está hablando de los arrays como estructura de datos. Un array o como también se le conoce: arreglo, es un conjunto de variables (elementos) agrupadas bajo un único nombre en distintas casillas. En dicho lenguaje de programación existen dos tipos de arreglos: los de índices numéricos y los de índices asociativos, ambos poseen una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, para contar el número de elementos que comprende y movernos por dentro de él hacia delante o atrás. A partir de lo antes descrito y por decisión del equipo de arquitectura la siguiente vendría siendo la estructura de datos a utilizar de forma general para el desarrollo en todos los subsistemas.

- ➢ Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, encaso de que sea un nombre compuesto se empleará notación PascalCasing*. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: *GestionarCaja*

Nomenclatura según el tipo de clases

1. Clases controladoras

Las clases controladoras después del nombre llevan la palabra:”Controller”.

Ejemplo: GestionarCajaController

2. Clases de los modelos

Business (Negocio)

Las clases que se encuentran dentro de Business después del nombre llevan la palabra: “Model”.

Ejemplo: NomCajaModel

- ✓ Domain (Dominio)

Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base deDatos.

Ejemplo: NomCaja

- ✓ Generated (Dominio bases)

Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: “Base” y seguido el nombre de la tabla en la Base de Datos.

Ejemplo: BaseNomCaja

- ➢ Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing. Esta es similar a la PascalCasting con la diferencia de la primera letra..

Ejemplo: insertarCaja

En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra:”Action”

Ejemplo: insertarCajaAction

- ➢ Nomenclatura de las variables

El nombre a emplear para las variables se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing, y comenzando con un prefijo según el tipo de datos.

Ejemplo: arrCaja

- ➢ Nomenclatura de los atributos

El nombre a emplear para los atributos se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing*. Además, en caso de ser un objeto se comienza con:"_" y después se escribe el nombre.

Ejemplo: intMoneda = dinero

objMoneda= _dinero

El uso de los estándares que estableció el proyecto facilita el trabajo y mejora el entendimiento entre el equipo del módulo, ya que se crea una mejora en la comprensión del código y el futuro mantenimiento del mismo.

2.3.4 Descripción de la Estructura e Implementación por Componentes.

La presente versión de Cedrux no presenta variación en cuanto a la versión anterior con relación a la cantidad de Componentes del módulo, pero cada uno de ellos sufrió modificaciones en relación a las nuevas funcionalidades del fondo fijo y al fondo de pagos planificados que lo hacen mucho más eficiente, a la vez que se beneficiarán un mayor número de entidades y permitirá gestionar todos los procesos de Caja de forma automatizada.

Se ampliaron la cantidad de tipos de fondos definidos, añadiéndose en la funcionalidad Gestionar tipo de Fondo del Componente Configuración el Fondo para Caja, el cual admite movimientos de entrada y salida, viabilizando la posibilidad de manejar el método para Caja de Fondo Fijo. A este tipo de fondo se le podrán asociar todas las operaciones que sean de entrada y salida, así como las operaciones predefinidas por el sistema como son las operaciones de Arqueo y las de completitud del fondo. También es nueva la opción de pago planificado en Gestionar Fondos. Mediante esta opción el fondo queda listo para que se efectúen en el todas las operaciones de pago de salario a los trabajadores que se explicarán más adelante.

Dentro del componente Movimientos la funcionalidad Anticipo de viaje Nacional es presentada como Anticipo de dieta, una solución más genérica que engloba todas las posibles opciones por las cuales se puede realizar en cualquier entidad empresarial un reembolso de efectivo por cuestión de anticipo. La funcionalidad Movimiento genérico del mismo componente fue adaptada para poder crear este tipo de movimiento sobre un fondo que pueda realizar operaciones de entrada y salida al mismo tiempo, siendo el fondo más operable.

Se le añadió al componente la funcionalidad Movimiento entre Caja, la cual realiza sus operaciones solamente sobre el fondo de tipo pago anticipado que posea la caja, el cual debe ser único. Se realizarán las siguientes operaciones en el mismo:

- Entrada de efectivo por motivo de pagos de salarios a los trabajadores: mediante este movimiento se le da entrada a la caja de un monto de efectivo a partir del saldo de la nómina que se lleva en el Capital Humano de las empresas de forma manual.

- Liquidación del efectivo que fue entregado como salario: Al no existir aún una solución de nóminas en el sistema, esta liquidación se realiza de manera general de lo pagado a todos los trabajadores por motivo de salarios al finalizar los tres días hábiles para su entrega.

- Movimiento entre Cajas: este movimiento da por cerrado el fondo al mover el importe que no fue cobrado por los trabajadores a otro fondo, permitiendo que se continúe operando con este dinero sin necesidad de hacer su entrega al Banco. El sistema se encarga de ajustar esos valores en las cuentas de fondos, gastos e ingresos.

Por último la funcionalidad Completitud de Fondo para Caja, que será la encargada en el caso de que el saldo actual del fondo sobrepase el saldo límite definido previamente. Para completar la diferencia se realizará un movimiento de depósito (salida) hasta el saldo límite, y en caso contrario, en el momento en el que el cajero identifique que tiene muy poco efectivo en el fondo puede realizar un movimiento de completitud (entrada) hasta el límite del fondo. En el primer caso el sistema va emitir una alerta de forma permanente hasta que el cajero realiza la acción contable mediante la operación predefinida.

2.3.5 Descripción de clases y operaciones del módulo.

Clases incluidas en el componente Configuración.

✓ **Generalización de las clases del Modelo**

Nombre: *NombreClaseModel*	
Tipo de clase: *Modelo*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
NombreClaseModel ()	Constructor de la clase.
Insertar (& $instance)	Responsabilidad encargada de insertar un objeto en la base datos.
Actualizar (& $instance)	Responsabilidad encargada de modificar un objeto en la base datos.

Eliminar (& $instance)	Responsabilidad encargada de eliminar un objeto en la base datos.

- ✓ **Clases del tipo Entidad**

Descripción de la clase Tipo Movimiento Caja

Nombre: *NomTipomovimientocaja*	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarMovimientos ()	Encargado de devolver todos los movimientos.
cargarTiposMovimientos ()	Encargado de devolver todos los tipos de movimientos.
obtenerMovimiento ($idmovimiento)	Encargado de devolver un movimientos.
buscar(arreglo $condiciones)	Encargado de devolver todos los movimientos.
obtenerDependencia($idtipomov)	Encargado de devolver todos los movimientos.

Descripción de la clase Nomenclador de Tipos de fondos

Nombre: TipoFondo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarTipoFondos ()	Encargado de devolver todos los fondos.
existeCodigo ()	Encargado de verificar si hay un tipo de fondo existente
obtenerTipoFondo ($idtipofondo)	Encargado de devolver un tipo de fondo específico.
dameTipoFondo($idfondo)	Encargado de devolver todos los tipos de fondo de un fondo.
existeTipoMovimiento($idfondo,$idmov)	Encargado de verificar si existe un tipo de movimiento.

Descripción de la clase Nomenclador de fondos

Nombre: NomFondo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarFondos($idcaja)	Encargado de devolver todos los fondos de un caja.
buscar(arreglo $condiciones)	Encargado de buscar los fondos
obtenerFondos($idfondo)	Encargado de obtener los fondos por un id
verificarFondos($idtipofondo)	Encargado de verificar si existe un fondo que coincida con el tipo de fondo pasado por parámetros.

obtenerDependencia($idfondo)	Encargado de verificar si está asociado el fondo a una caja

Descripción de la clase Nomenclador de cajas

Nombre: NomCaja	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarCaja()	Encargado de devolver todas las cajas de la entidad
obtenerCaja($idcaja)	Encargado de obtener una caja dado el id
buscar(arreglo $condiciones)	Encargado de buscar las cajas de la entidad
cajaAsociada($idcaja)	Encargado de devolver una caja si esta está asociada

Descripción de la clase Nomenclador tipo de fondos-tipos de movimientos

Nombre: DatTipofondotipomovimiento	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
dameFondoDocumento($idfondo)	Encargado de devolver todos los documentos del fondo.

Descripción de la clase Apertura de fondo

Nombre: DatAperturafondo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
existeApertura($idfondo)	Encargado de verificar si el fondo tiene apertura realizadas.

Descripción de la clase Caja y fondos.

Nombre: DatCajafondo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarCajaFondo($idcaja)	Encargado de devolver todos los fondos asociados a una caja.
fondoSinApertura($idcaja)	Encargado de devolver los fondos que no tienen realizadas aperturas.

getIdFondoCaja($idfondo)	Encargado de devolver todos los ID de fondoCaja
obtenerFondosConApertura()	Encargado de obtener todos los fondos con aperturas realizadas.

Clases incluidas en el componente Movimiento.

- ✓ **Generalización de las clases del Modelo**

Nombre: *NombreClaseModel*	
Tipo de clase: *Modelo*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
NombreClaseModel ()	Constructor de la clase.
Insertar (& $instance)	Responsabilidad encargada de insertar un objeto en la base datos.
Actualizar (& $instance)	Responsabilidad encargada de modificar un objeto en la base datos.
Eliminar (& $instance)	Responsabilidad encargada de eliminar un objeto en la base datos.

- ✓ **Clases del tipo Entidad**

Descripción de la clase Reembolso

Nombre: DatReembolso	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarDocumentosReembolso($idfondo)	Encargado cargar los documentos de reembolso por el id de fondo
reembolsar()	Encargado de reembolsar

Descripción de la clase Anticipo de Dieta

Nombre: DatAnticipoviajenacional	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarAnticiposPendienteEntrega()	Encargado de cargar los anticipos pendientes a entregar.

Descripción de la clase liquidación del anticipo de Dieta

Nombre: DatLiquidacionviajenacional	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	

Nombre.	Descripción.
setUp ()	Constructor de la clase.
cargarAnticiposPendienteLiquidar()	Encargado de cargar los anticipos nacionales pendientes a liquidar.

Descripción de la clase anticipo de viaje extranjero

Nombre: DatLiquidacionviajeextranjero	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarAnticiposPendienteLiquidar()	Encargado de cargar los anticipos extranjeros pendientes a liquidar.

Descripción de la clase Detalles del Movimiento

Nombre: DatDetallesmovimiento	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarDetalles($idmov)	Encargado de devolver todos los detalles de los movimientos.

Descripción de la clase Depósito

Nombre: DatDeposito	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarDocumentosDeposito($idfondo)	Encargado cargar los documentos de deposito por el id de fondo
depositar()	Encargado de depositar
desglosarDinero()	Encargado de desglosar el dinero.

Descripción de la clase Anticipo

Nombre: DatAnticipo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtenerAnticipo($idanticipo)	Encargado de obtener un objeto anticipo.

Descripción de la Clase Movimiento Genérico

Nombre: ContabilizarMov	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
Contabilizar ()	Devuelve los documentos a contabilizar

Nombre: DatMovimientogenericoModel	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
ObtenerInstByIdFondo ()	Devuelve los instrumentos de los movimientos
CargarMovGenericosCriterio()	Carga los movimientos genéricos del fondo
BuscarDatosAsociados()	Carga los datos asociados al movimiento
InsertarInstrumentosdeMovGen()	Inserta los instrumentos del movimiento

Clases incluidas en el componente Arqueo.

- ✓ **Clases del tipo Modelo**

Descripción de la clase Arqueo de Caja

Nombre: DatArqueosModel	
Tipo de clase: *Model*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
DatArqueosModel ()	Constructor de la clase.
guardarArqueo()	Encargado de guardar los arqueos realizados.
eliminarArqueo($idarqueo)	Encargado de eliminar un arqueo realizado por id.

- ✓ **Clases del tipo Entidad**

Descripción de la clase Arqueo

Nombre: DatArqueo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarArqueosRealizados($idarqueo)	Encargado de cargar un arqueo por su id.
buscarArqueo($idarqueo)	Encargado de buscar un arqueo por id

Descripción de la clase Documento de Arqueo

Nombre: DatDocumentoarqueo	
Tipo de clase: *Entidad*.	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
cargarDocArqueo()	Encargado de cargar todos los documento generados por un arqueo.
buscarDocArqueo($iddocArqueo)	Encargado de buscar los documentos generados por un arqueo.
eliminarDocArqueo($iddocarqueo)	Encargado de eliminar un documento generado por un arqueo

2.4 CONCLUSIONES

El diseño brindado por el analista ayudó en gran medida a la obtención correcta de los componentes a implementar, y en lo referente a los requisitos no funcionales del sistema. A partir de ellos se llevó a cabo la implementación de las nuevas funcionalidades del módulo Caja.

CAPÍTULO III. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 INTRODUCCIÓN

La calidad de un producto de software es el indicador que permite determinar si los procesos de construcción de software fueron apropiados. Pero, aunque a este renglón son dedicados grandes cantidades de esfuerzos, el software en sí nunca es perfecto. Es por esto que debe indagarse sobre los métodos y técnicas que garantizan calidad en los productos, con miras a generar propuestas concretas para aplicaciones con características específicas. Estos deben aplicarse desde las más tempranas etapas del desarrollo, en aras de detectar y corregir a tiempo una mayor cantidad de errores.

También se tendrá en cuenta para la búsqueda de la calidad y a la hora de emitir un criterio de valoración del producto final factores tales como la funcionalidad, la usabilidad, la eficiencia y la escalabilidad.

En el capítulo siguiente se le aplicará a la propuesta de solución el método de la caja negra, la cual, olvidando el comportamiento interno del sistema se enfocará en su interfaz. Se validará también el diseño propuesto en el capítulo anterior para el desarrollo de la nueva versión de Caja mediante las métricas de calidad Tamaño Operacional de la Clase (TOC) y Relaciones entre Clases (RC), teniendo en cuenta las clases y operaciones definidas con antelación.

3.2 PRUEBAS DE SOFTWARE.

Las pruebas de software existen debido a nuestra capacidad innata de provocar errores. Una vez generado el código fuente, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de su entrega al cliente. El objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores. Aquí es donde se aplican las técnicas de pruebas del software. Estas técnicas facilitan una guía sistemática para diseñar pruebas que:

- ✓ Comprueban la lógica interna de los componentes software.
- ✓ Verifican los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento. (Pressman, 2002).

También deberán evaluarse durante esta fase todos los artefactos generados, lo que incluye especificaciones de requisitos, diagramas de diversos tipos, el código fuente y el resto de productos que forman parte de la aplicación, ejemplo: la base de datos.

3.2.1 Tipos de pruebas

- **Prueba de Caja Blanca**

Las pruebas de caja blanca, también conocidas como *pruebas de caja de cristal* o *pruebas estructurales*, se centran en los detalles procedimentales del software, por lo que su diseño está fuertemente ligado al código fuente.

Este método intenta garantizar que se ejecutan al menos una vez todos los caminos independientes que presente el módulo, que las decisiones que presenta el código, tanto en su versión verdadera como en la falsa, sean cumplidas, que se ejecutarán todos los bucles en sus límites y que todas las estructuras de datos internas serán usadas.

- **Prueba de Caja Negra**

Esta prueba se encarga de verificar la operatividad de las funciones del sistema. Obviando el comportamiento interno del programa y su estructura, esta se lleva a cabo sobre la interfaz del software.

Como pretensión fundamental de este tipo de prueba está encontrar los errores enumerados a continuación:

- ✓ 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

3.3 PRUEBAS APLICADAS AL MÓDULO.

3.3.1 Aplicación de la prueba de Caja Negra

Dentro de la prueba de caja negra se incluyen las Técnicas de Pruebas que serán descritas a continuación:

Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

A continuación se aplica la prueba de partición de equivalencia como parte de la realización de la prueba de caja negra sobre la interfaz que responde al requisito

funcional Anticipos La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Luego de aplicados el método de prueba por el equipo de desarrollo al requisito funcional Anticipo, es válido señalar que los resultados obtenidos hasta el momento han sido satisfactorios desde el punto de vista funcional del componente, atendiendo al correcto comportamiento del mismo ante diferentes situaciones (entradas válidas y no válidas).

3.4 VALIDACIÓN DEL MODELO DE DISEÑO PROPUESTO

Una de las principales condiciones en la evaluación de una propuesta de solución es la validación de su diseño e implementación mediante la aplicación de métricas. Según Jaime Ramírez, “Las métricas de diseño a nivel de componentes se concentran en las características internas de los componentes del software e incluyen entre otras medidas la cohesión, acoplamiento y complejidad del módulo, medidas que pueden ayudar al desarrollador de software a juzgar la calidad de un diseño a nivel de los componentes.” (Ramírez, 2005)

En tal sentido, a continuación se realizara una evaluacion del diseño propuesto para la nueva versión de Caja, teniendo en cuenta los siguientes atributos de calidad:

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

- ✓ **Cantidad de pruebas**: Número o grado de esfuerzo necesario para realizar las pruebas de calidad al producto (componente) diseñado.

Las métricas necesarias para evaluar los siguientes artefactos son las mencionadas a continuación:

- **Tamaño operacional de clase (siglas: TOC):** Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, existiendo una relación directa con los dos primeros e inversa con el último antes

Resultados:

La parte de imagen con el identificador de relación rId28 no se encontró en el archivo.

Ilustración 5 Resultados obtenidos en la métrica agrupados en los intervalos definidos.

La ilustración siguiente expresa los valores del resultado anterior en %.

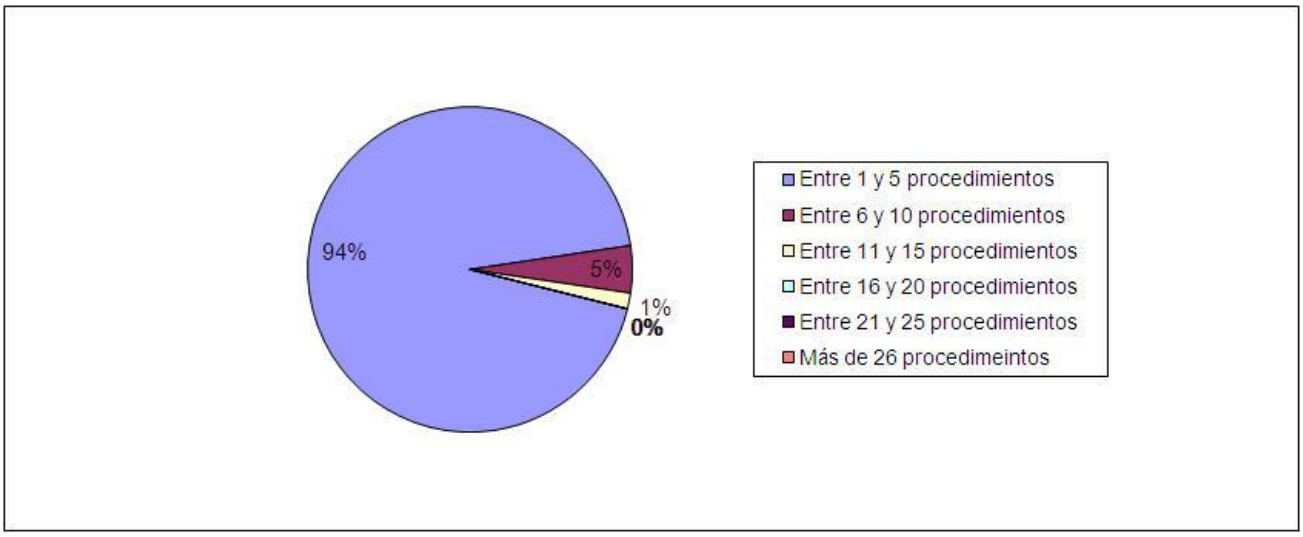


Ilustración 6: Resultados obtenidos en la métrica en % agrupados en los intervalos definidos.

Los resultados obtenidos en los atributos de calidad fueron los siguientes:

Responsabilidad

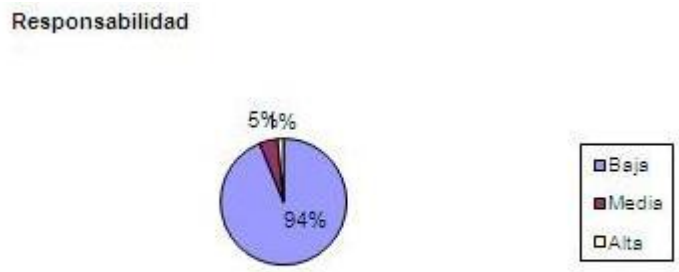


Ilustración 7: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

Complejidad del Implementación

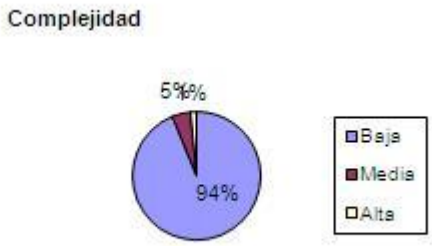


Ilustración 8: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

Reutilización

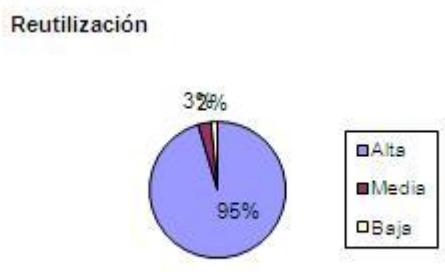


Ilustración 9: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Las conclusiones que podemos sacar de la aplicación de esta métrica son que la nueva versión del componente Caja se encuentra dentro de los límites aceptables de calidad pues una gran mayoría de sus clases, exactamente el 94% del total poseen

una cantidad de operaciones menor que la media. Se fomenta la reutilización del código y se hace un esfuerzo por reducir la responsabilidad y la complejidad de la implementación de las clases.

- **Relaciones entre clases (siglas: RC):** Dado por el número de relaciones de uso de una clase. Está determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

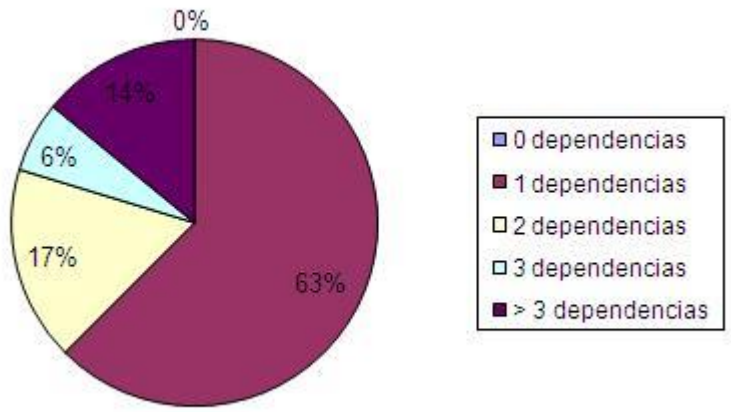


Ilustración 10: Representación de los resultados obtenidos en la métrica agrupados por intervalos específicos.

Representación de la incidencia de los resultados obtenidos en esta métrica en el atributo Complejidad de Mantenimiento.

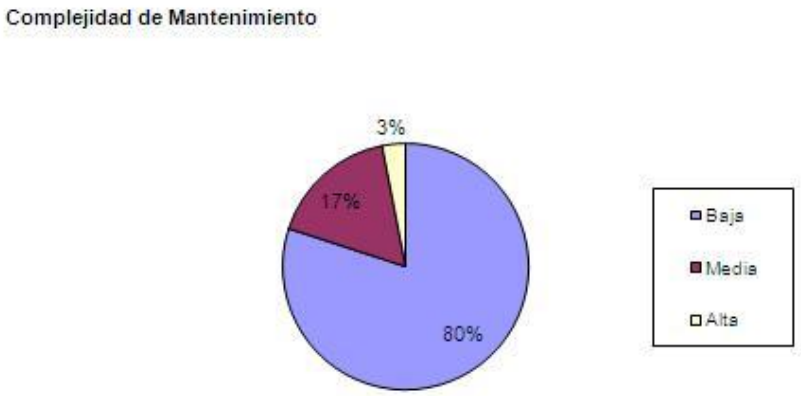


Ilustración 11 Representación de la incidencia de los resultados obtenidos en esta métrica en el atributo Complejidad de Mantenimiento

La incidencia en el atributo Acoplamiento de los resultados en la métrica fue la siguiente:

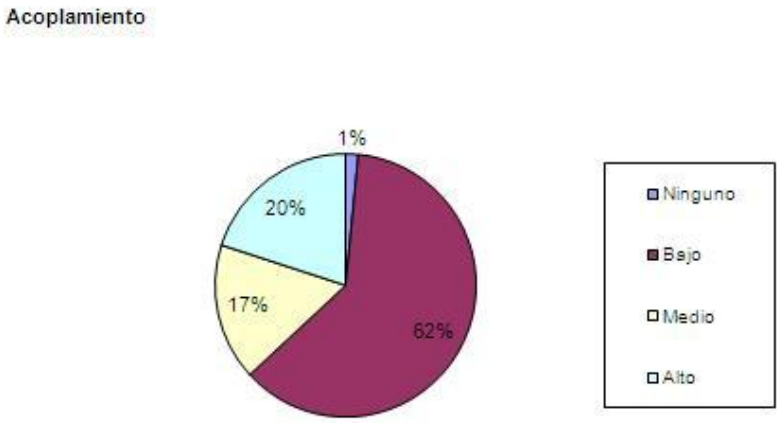


Ilustración 12: Representación de la incidencia de los resultados obtenidos en esta métrica en el atributo Acoplamiento

Los resultados en el atributo Cantidad de Prueba fueron los siguientes

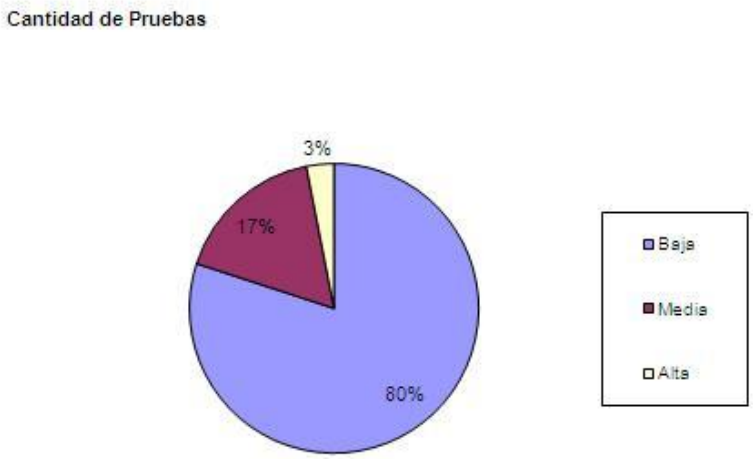


Ilustración 13: Representación de la incidencia de los resultados obtenidos en esta métrica en el atributo Cantidad de Prueba

Luego del análisis de los resultados obtenidos en la aplicación de la métrica Relación de Clases, se concluye que el componente esta en los límites aceptables de calidad, ya que la gran mayoría de sus clases posee menos de 3 dependencias respecto de otras

3.5 CONCLUSIONES

La validación de la calidad de la nueva versión del componente Caja fue el elemento clave del capítulo que recién concluye. En tal sentido se efectuaron pruebas de software mediante casos de pruebas; se aplicaron además las métricas: Relaciones entre Clases y Tamaño Operacional de la Clase para validar y evaluar el diseño, las cuales arrojaron valores satisfactorios para cada uno de los indicadores correspondientes. Se concluye, pues que la solución desde el punto de vista funcional cumple con los requerimientos capturados y especificados en las primeras etapas de desarrollo a partir de las expectativas del cliente.

CONCLUSIONES GENERALES

Dada la no idoneidad de los sistemas que gestionan en la actualidad los recursos de Caja en las empresas financieras cubanas -incluyendo la versión original de Cedrux- , en cubrir todas las características propias de estas últimas, se hizo necesario de implementar una nueva versión de este módulo que abarque todas las no conformidades pendientes en el componente.

Enfocado en esto se realizó un estudio del estado actual del arte en Cuba y el mundo analizando las deficiencias y las ventajas de los sistemas más utilizados. Posteriormente se realizó la implementación del componente Ajuste al Costo correspondiente al Subsistema Costos y Procesos del Sistema Integral de Gestión de Entidades CEDRUX, con el objetivo de erradicar los problemas de los sistemas existentes y fusionar sus mejores prácticas.

Por último se evaluó la viabilidad del componente por medio de métricas realizadas al diseño del mismo y pruebas realizadas a la propuesta de solución, arrojando las mismas resultados favorables al cumplimiento de los requisitos planteados por el analista

Todo esto ayudó al tomar la conclusión de que la presente propuesta de solución cumple todos los requisitos que se plantearon en un inicio para ella en la búsqueda del correcto gestionar de las finanzas correspondientes a Caja, manteniendo la integración propia con el resto de los módulos del Sistema Cedrux

RECOMENDACIONES

Partiendo de lo obtenido hasta el momento se recomienda

- ✓ Continuar realizando pruebas de calidad al componente.
- ✓ Realizar el despliegue de la aplicación en varias entidades con el fin de comprobar si realmente cumple con las necesidades del cliente desde el punto de vista tecnológico.
- ✓ Realizar nuevas investigaciones con el fin de añadirle nuevas funcionalidades al componente en aras de su optimización.

BIBLIOGRAFIA REFERENCIADA

AbartiaTeam. (2006). Recuperado el 11 de Marzo de 2011, de http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web

Alonso, J. (2001). *TerritorioPC.* Recuperado el 2011, de http://www.territoriopc.com/javascript/tutorial_javascript_introduccion.php.

Alvarez, M. A. (2010). *Arquitectura de Software. Desarrollo web.* Recuperado el 7 de Mayo de 2011, de Desarrollo web.: http://www.desarrolloweb.com/articulos/392.php

Apache Documentation. (s.f.). Recuperado el 15 de marzo de 2011, de http://httpd.apache.org/docs/2.0/es/

Díaz Cobas, W., & Martinez, R. (2009). *Implementación del módulo Caja del Sistema Integral de Gestión CEDRUX.* Habana.

Diseño de base de datos relacional. (2010). Obtenido de http://usuarios.multimania.es/cursosgbd/UD4.htm

Guillerón, G. (12 de Julio de 2009). *GLN Ingeniería & Consultoría.* Obtenido de http://glnconsultora.com/blog/?tag=inversion-de-control

Introducing JSON. Sitio oficial de JSON. (s.f.). Obtenido de http://www.json.org/

Johanns, A. (13 de noviembre de 2008). Obtenido de N-capas: http://n-capas.blogspot.com/2008/11/modelo-n-capas_13.html

Los datos de Objetos en PHP. (2010). Obtenido de sitio oficial de PHP: http://php.net/manual/es/book.pdo.php

OpenBravo. Opening the ERP future. (2007). Obtenido de OpenBravo: http://www.openbravo.com/es/product/erp/features

Pressman, R. S. (2002). *Un enfoque práctico.* Madrid: Editorial Mc Graw Hill.

Ramírez, J. (2005). *Unidad de Programación. Métodos de Prueba del Software.*

Sistemas ERP en las PYMES. Claves para implementar un ERP de manera exitosa. (2011). Obtenido de Diario Pyme: http://www.diariopyme.cl/newtenberg/1805/article-70424.html

Valade, J. (2004). *PHP for Dummnies.* Wiley Publishing, Inc.

Vega, Y. (2009). *Definición del ciclo de vida del proyecto.* La Habana.

Visual Paradigm. (2005). Recuperado el 2011 de marzo de 15, de Sitio web de Visual Paradigm: www.visual-paradigm.com.

Zend Studio. (2010). Obtenido de The Professional PHP IDE – Zend: http://www.zend.com/products/studio

GLOSARIO DE TÉRMINOS

Arqueo: El Arqueo de Caja consiste en el análisis de las transacciones del efectivo, durante un lapso determinado, con el objeto de comprobar si se ha contabilizado todo el efectivo recibido.

Código abierto: Término con el que se conoce al software distribuido y desarrollado libremente.

Componente: Un componente es una clase de uso específico, que puede ser configurada o utilizada de forma visual desde el entorno de desarrollo.

DOM: Document Object Model (una traducción al español no literal, pero apropiada, podría ser Modelo en Objetos para la representación de Documentos), abreviado DOM, es esencialmente una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

ERP: Sistemas de planificación de recursos empresariales (Enterprise Resource Planning, ERP por sus siglas en inglés).

Escalabilidad: La escalabilidad es la capacidad de mejorar recursos para ofrecer una mejora (idealmente) lineal en la capacidad de servicio. La característica clave de una aplicación es que la carga adicional sólo requiere recursos adicionales en lugar de una modificación extensiva de la aplicación en sí.

Herramientas: Es un objeto elaborado a fin de facilitar la realización de una tarea mecánica que requiere de una aplicación correcta de energía.

JAVA: Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

JSON: acrónimo de "JavaScript Object Notation", es un formato ligero para el intercambio de datos. .

Metodología: Se refiere a los métodos de investigación que se siguen para alcanzar una gama de objetivos.

Módulo: Es un software que agrupa un conjunto de subprogramas y estructuras de datos.

Navegador web: Programa que permite ver la información que contiene una página web

ORM: El Mapeo Objeto Relacional es una técnica de programación para 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

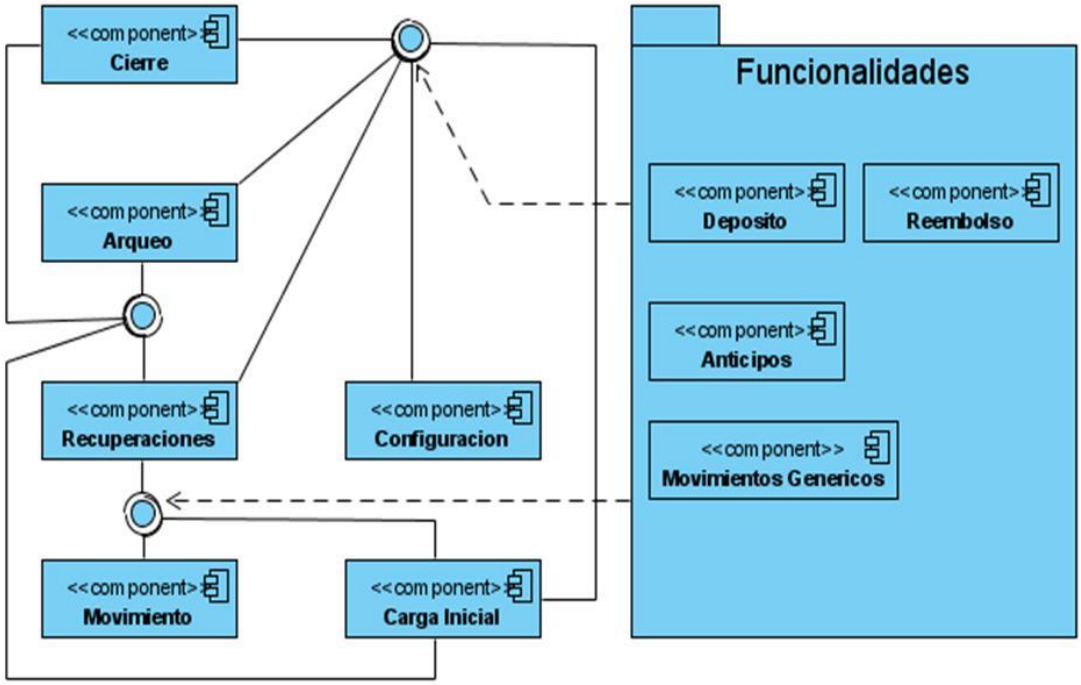
Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente

Subsistema: Son las partes que forman un sistema. Cada sistema está compuesto de subsistemas, los cuales a su vez son parte de otros subsistemas; cada subsistema es delineado por sus límites.

XHTML: Reformulación del lenguaje HTML utilizado para crear páginas Web.

ANEXOS

Anexo 1 Diagrama de Componentes Módulo Caja



Anexo 2 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
ArqueosModel	3	Media	Media	Alta
DatArqueodocumentoModel	2	Baja	Baja	Alta
DatArqueo	3	Baja	Baja	Alta
DatMovimientoinstrumento	1	Baja	Baja	Alta
DatReembolso	3	Baja	Baja	Alta
NomClasificacion	1	Baja	Baja	Alta
DatAnticipo	2	Baja	Baja	Alta
DatAnticipoviajeextranjero	1	Baja	Baja	Alta
DatAnticipoviajeextranjeropais	1	Baja	Baja	Alta
DatAnticipoviajenacional	2	Baja	Baja	Alta
DatDocumentoreembolso	1	Baja	Baja	Alta
NomEstadoanticipo	1	Baja	Baja	Alta
DatAnticipoviajenacionalentregado	1	Baja	Baja	Alta
NomEstadoanticipoModel	2	Baja	Baja	Alta
DatAnticipoviajenacionalentregadoModel	2	Baja	Baja	Alta
DatCantidaddenominacion	1	Baja	Baja	Alta
DatDeposito	4	Baja	Baja	Alta
DatInstrumentodeposito	1	Baja	Baja	Alta

DatDetallesmovimiento	2	Baja	Baja	Alta
DatLiquidacion	1	Baja	Baja	Alta
DatLiquidacionviajeextranjero	1	Baja	Baja	Alta
DatLiquidacionviajeextranjeropais	2	Baja	Baja	Alta
DatLiquidacionviajenacional	2	Baja	Baja	Alta
DatMovimiento	1	Baja	Baja	Alta
DatMovimientogenerico	1	Baja	Baja	Alta
DatAnticipoModel	3	Baja	Baja	Alta
DatAnticipoviajeextranjeroModel	2	Baja	Baja	Alta
DatAnticipoviajeextranjeropaisModel	2	Baja	Baja	Alta
DatAnticipoviajenacionalModel	2	Baja	Baja	Alta
DatDocumentoreembolsoModel	2	Baja	Baja	Alta
NomFondoModel	4	Baja	Baja	Alta
NomTipoFondoModel	4	Baja	Baja	Alta
NomTipoMovimientocajaModel	4	Baja	Baja	Alta
DatAperturafondointrumento	1	Baja	Baja	Alta
DatAperturafondo	2	Baja	Baja	Alta
DatCajafondo	5	Baja	Baja	Alta
DatTipoinstrumentotipomovimiento	1	Baja	Baja	Alta
DatTipofondotipomovimiento	2	Baja	Baja	Alta
DatTipoinstrumentotipofondo	1	Baja	Baja	Alta
NomCaja	5	Baja	Baja	Alta
NomFondo	9	Media	Media	Media
NomTipoFondo	8	Media	Media	Media
NomTipoMovimientocaja	6	Baja	Baja	Alta
DatCantidaddenominacionModel	2	Baja	Baja	Alta
DatDepositoModel	3	Baja	Baja	Alta
DatInstrumentodepositoModel	2	Baja	Baja	Alta
DatDetallesmovimientoModel	2	Baja	Baja	Alta
DatLiquidacionModel	4	Baja	Baja	Alta
DatLiquidacionviajeextranjeroModel	2	Baja	Baja	Alta
DatLiquidacionviajeextranjeropaisModel	2	Baja	Baja	Alta
DatLiquidacionviajenacionalModel	2	Baja	Baja	Alta
DatMovimientoModel	3	Baja	Baja	Alta
DatMovimientogenericoModel	4	Baja	Baja	Alta
DatMovimientoinstrumentoModel	2	Baja	Baja	Alta
DatReembolsoModel	3	Baja	Baja	Alta
NomClasificacionModel	2	Baja	Baja	Alta
CompletamientoController	15	Alta	Alta	Baja
DatAperturafondoModel	3	Baja	Baja	Alta
DatCajafondoModel	3	Baja	Baja	Alta
DatTipoinstrumentotipomovimientoModel	2	Baja	Baja	Alta
DatTipofondotipomovimientoModel	2	Baja	Baja	Alta
DatTipoinstrumentotipofondoModel	2	Baja	Baja	Alta
NomCajaModel	4	Baja	Baja	Alta
DatAperturafondointrumentoModel	2	Baja	Baja	Alta

Anexo 3 Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC

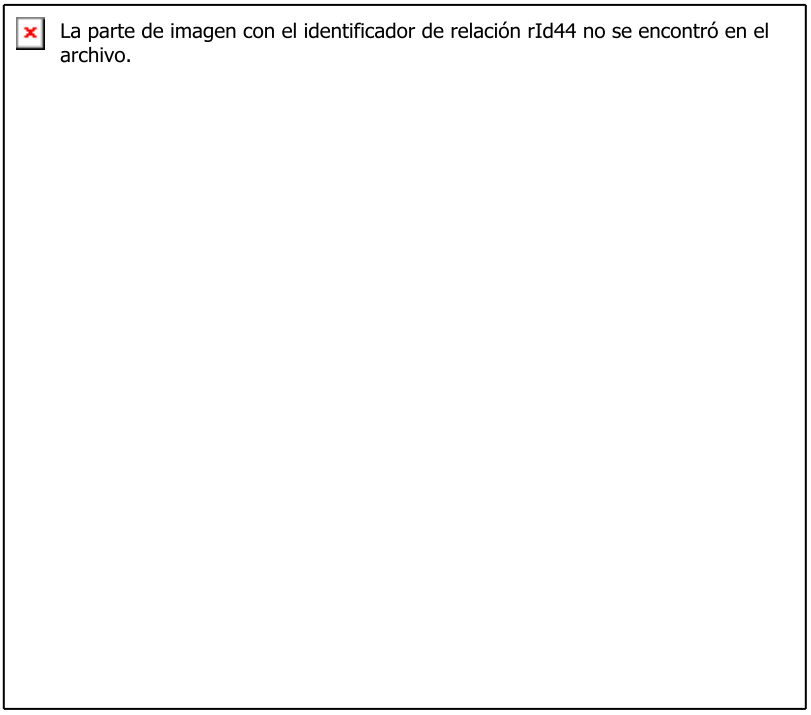
	Categoria	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

Anexo 4 Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
DatAperturafondointrumentoModel	1	Bajo	Baja	Alta	Baja
DatAperturafondoModel	2	Medio	Baja	Alta	Baja
DatCajafondoModel	2	Medio	Baja	Alta	Baja
DatTipoinstrumentotipomovimientoModel	1	Bajo	Baja	Alta	Baja
DatTipofondotipomovimientoModel	1	Bajo	Baja	Alta	Baja
DatTipoinstrumentotipofondoModel	1	Bajo	Baja	Alta	Baja
NomCajaModel	3	Alto	Media	Media	Media
NomFondoModel	4	Alto	Media	Media	Media
NomTipoFondoModel	4	Alto	Media	Media	Media
NomTipoMovimientocajaModel	2	Medio	Baja	Alta	Baja
DatAperturafondointrumento	1	Bajo	Baja	Alta	Baja
DatAperturafondo	3	Alto	Media	Media	Media
DatCajafondo	4	Alto	Media	Media	Media
DatTipoinstrumentotipomovimiento	1	Bajo	Baja	Alta	Baja
DatTipofondotipomovimiento	1	Bajo	Baja	Alta	Baja
DatTipoinstrumentotipofondo	1	Bajo	Baja	Alta	Baja
NomCaja	3	Alto	Media	Media	Media
NomFondo	5	Alto	Alta	Baja	Alta
NomTipoFondo	4	Alto	Media	Media	Media
NomTipoMovimientocaja	4	Alto	Media	Media	Media
DatCantidaddenominacionModel	1	Bajo	Baja	Alta	Baja
DatDepositoModel	3	Alto	Media	Media	Media
DatInstrumentodepositoModel	1	Bajo	Baja	Alta	Baja
DatDetallesmovimientoModel	1	Bajo	Baja	Alta	Baja
DatLiquidacionModel	2	Medio	Baja	Alta	Baja
DatLiquidacionviajeextranjeroModel	1	Bajo	Baja	Alta	Baja

DatLiquidacionviajeextranjeropaisModel	1	Bajo	Baja	Alta	Baja
DatLiquidacionviajenacionalModel	1	Bajo	Baja	Alta	Baja
DatMovimientoModel	2	Medio	Baja	Alta	Baja
DatMovimientogenericoModel	2	Medio	Baja	Alta	Baja
DatMovimientoinstrumentoModel	1	Bajo	Baja	Alta	Baja
DatReembolsoModel	2	Medio	Baja	Alta	Baja
NomClasificacionModel	1	Bajo	Baja	Alta	Baja
DatAnticipoModel	2	Medio	Baja	Alta	Baja
DatAnticipoviajeextranjeroModel	1	Bajo	Baja	Alta	Baja
DatAnticipoviajeextranjeropaisModel	1	Bajo	Baja	Alta	Baja
DatAnticipoviajenacionalModel	1	Bajo	Baja	Alta	Baja
DatDocumentoreembolsoModel	1	Bajo	Baja	Alta	Baja
NomEstadoanticipoModel	1	Bajo	Baja	Alta	Baja
DatAnticipoviajenacionalentregadoModel	1	Bajo	Baja	Alta	Baja
DatCantidaddenominacion	1	Bajo	Baja	Alta	Baja
DatDeposito	4	Alto	Media	Media	Media
DatInstrumentodeposito	1	Bajo	Baja	Alta	Baja
DatDetallesmovimiento	1	Bajo	Baja	Alta	Baja
DatLiquidacion	1	Bajo	Baja	Alta	Baja
DatLiquidacionviajeextranjero	1	Bajo	Baja	Alta	Baja
DatLiquidacionviajeextranjeropais	1	Bajo	Baja	Alta	Baja
DatLiquidacionviajenacional	1	Bajo	Baja	Alta	Baja
DatMovimiento	1	Bajo	Baja	Alta	Baja
DatMovimientogenerico	1	Bajo	Baja	Alta	Baja
DatMovimientoinstrumento	1	Bajo	Baja	Alta	Baja
DatReembolso	2	Medio	Baja	Alta	Baja
NomClasificacion	1	Bajo	Baja	Alta	Baja
DatAnticipo	4	Alto	Media	Media	Media
DatAnticipoviajeextranjero	1	Bajo	Baja	Alta	Baja
DatAnticipoviajeextranjeropais	1	Bajo	Baja	Alta	Baja
DatAnticipoviajenacional	1	Bajo	Baja	Alta	Baja
DatDocumentoreembolso	1	Bajo	Baja	Alta	Baja
NomEstadoanticipo	1	Bajo	Baja	Alta	Baja
DatAnticipoviajenacionalentregado	1	Bajo	Baja	Alta	Baja
ArqueosModel	2	Medio	Baja	Alta	Baja
DatArqueodocumentoModel	1	Bajo	Baja	Alta	Baja
DatArqueo	5	Alto	Alta	Baja	Alta
CompletamientoController	2	Medio	Baja	Alta	Baja

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



Anexo 6 Requisitos a probar en la prueba de Caja Negra en la funcionalidad Gestionar Fondo

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar fondo.	*Se registra un nuevo fondo con los datos nombre, código, tipo, saldo límite y descripción.*	EP 1.1: Adicionar un fondo correctamente.	– Se presiona el botón **Adicionar.** – Se abre la ventana **Adicionar fondo.** – Se introducen correctamente los datos. – Se presiona el botón **Aceptar.** – Se presiona el botón **Aceptar** de la ventana de información: "Los datos han sido guardados satisfactoriamente.".
		EP 1.2: Adicionar un fondo introduciendo datos inválidos.	– Se presiona el botón **Adicionar.** – Se abre la ventana **Adicionar**

			fondo. – Se introducen datos inválidos. – Se muestra el mensaje: “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”.
		EP 1.3: Adicionar un fondo dejando campos vacíos.	– Se presiona el botón **Adicionar.** – Se abre la ventana **Adicionar fondo.** – Se introducen los datos dejando campos vacíos. – Se presiona el botón **Aceptar.** – Se muestra el mensaje: “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”.
		EP 1.4: Adicionar un fondo registrando código igual a uno ya registrado.	– Se presiona el botón **Adicionar.** – Se abre la ventana **Adicionar fondo.** – Se introducen los datos. – Se presiona el botón **Aceptar.** – Se presiona el botón **Aceptar** de la ventana de información: “El código ya existe, seleccione uno nuevo.”.
		EP 1.5: Cancelar.	– Se presiona el botón **Adicionar.** – Se abre la ventana **Adicionar fondo.** – Se introducen o no los datos. – Se presiona el botón **Cancelar.**

Anexo 7 Juegos de datos a probar en la Prueba de Caja Negra a la funcionalidad Gestionar Fondo

Id del escenario	Escenario	Nombre	Código	Tipo	Saldo límite	Descripción	Respuesta del sistema	Resultado de la prueba
EP 1.1.	Adicionar un fondo correctamente.	V(Fondo para pagos menores)	V(54121)	V(Fondo CUP)	V(5000)	V(Fondo en CUP para pagos menores)	El sistema muestra el mensaje: “Los datos han sido guardados satisfactoriamente.”.	
EP 1.2.	Adicionar un fondo introduciendo datos inválidos.	I(Fondo#2136 para pagos//// menores)	V(54121)	V(Fondo CUP)	V(5000)	V(Fondo en CUP para pagos menores)	El sistema no permite registrar datos inválidos. Se muestra la alerta: “Este campo es obligatorio.”. Se muestra el mensaje: “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”.	
		V(Fondo para pagos menores)	I(#Treinta y cinco)	V(Fondo CUP)	V(5000)	V(Fondo en CUP para pagos menores)		
		V(Fondo para pagos menores)	V(54121)	V(Fondo CUP)	I(5mil)	V(Fondo en CUP para pagos menores)		
EP 1.3.	Adicionar un fondo dejando campos vacíos.	I(Vacío)	V(54121)	V(Fondo CUP)	V(5000)	V(Fondo en CUP para pagos menores)	El sistema muestra la alerta “Este campo es obligatorio.”. Si no se	
		V(Fondo para pagos	I(Vacío)	V(Fondo CUP)	V(5000)	V(Fondo en CUP para pagos		

		menores)				menores)	registran los datos obligatorios se muestra el mensaje: “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”.	
		V(Fondo para pagos menores)	V(54121)	I(Vacío)	V(5000)	V(Fondo en CUP para pagos menores)		
		V(Fondo para pagos menores)	V(54121)	V(Fondo CUP)	I(Vacío)	V(Vacío)		
EP 1.4.	Adicionar un fondo registrando código igual a uno ya registrado.	V(Fondo para pagos menores)	I(54121)	V(Fondo CUP)	V(5000)	V(Fondo en CUP para pagos menores)	El sistema muestra el mensaje: “El código ya existe, seleccione uno nuevo.”.	
EP 1.5.	Cancelar.	NA	NA	NA	NA	NA	Se cancela la operación y se cierra la ventana.	