Título: Implementación del módulo Caja del Sistema Integral de Gestión CEDRUX

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

Autores: Rigoberto Martínez Álvarez

Walberto Díaz Cobas

Tutores: Ing. Víctor Luis Borroto Alvariño

Ing. Julio Cardoso Ventura

Ciudad de la Habana

Mayo 2009

"...la ciencia no es ni misterio de iniciados, ni privilegio de los aristócratas de la mente, sino el medio único que tiene el hombre de explicarse las leyes de la vida..."

José Martí

A nuestra familia

RESUMEN

Durante años la contabilidad ha sido el método que dirige la economía de los países, pero el siglo XX dio un cambio en la forma en que se almacenaban los datos, hasta ese momento en hoja y papel, haciendo que los trabajadores encargados de contabilizar pasaran gran cantidad de horas en tareas engorrosas por el gran volumen de información. Para ello se fueron creando software de sistemas de planificación de recursos que se encargan de llevar en tiempo y forma todos los procesos empresariales de manera confiable, precisa y oportuna.

Nuestro país no está ajeno a este desarrollo y en las dos últimas décadas a dado un salto cuantitativo y cualitativo en busca de la informatización de la sociedad. Las empresas nacionales han incorporado software de gestión para la planificación de los recursos, los cuales no poseen sistemas que lleven a cabo los procesos de Caja, por lo que las labores se realizan de forma manual, provocando pérdida de datos, lentitud en el proceso y haciendo que un mismo documento se opere varias veces. Por tales motivos se decidió realizar un sistema informático que agilice los procesos de Caja.

En el presente trabajo se realiza un estudio detallado de diferentes sistemas que automatizan los procesos de Caja tanto nacionales como internacionales identificando las principales ventajas y desventajas de los mismos, tomando las mejores soluciones, para adaptarlas al sistema financiero cubano. Además se realiza un análisis de las herramientas, técnicas, metodologías y software actuales que serán usados para construir la solución del problema planteado, además de la fundamentación del uso de las metodologías y técnicas escogidas para realizar el trabajo.

Se realiza una descripción de los principales procesos que se identificaron para automatizar. Se confeccionan los prototipos de interfaz de la aplicación acordes a las necesidades del sistema y se detalla la estructura del módulo, desglosándolo por componentes. Finalmente se aplican una serie de métricas y pruebas al proyecto, obteniendo un producto con la calidad requerida.

Con la implementación de este sistema informático se dará un vuelco en el proceso de Caja, ya que este permite una mejor gestión y control de las cajas de la empresa cubanas, dada la eficacia en la toma de decisiones, la agilización de tiempo, el trabajo óptimo y la fácil interacción del usuario con el sistema.

INTRODUCCIÓN. 1

CAPITULO I: FUNDAMENTACION TEORICA 4

1.1 INTRODUCCIÓN 4
1.2 PROCESOS DE SISTEMAS DE CAJA 4
1.3 SISTEMAS AUTOMATIZADOS QUE POSEEN PROCESOS DE CAJA EN CUBA Y EL MUNDO. 4
1.3.1 Módulo de Finanzas y Caja en el Versat-Sarasola. *4*
1.3.2 Módulo Finanzas en el sistema RODAS XXI *5*
1.3.3 Módulo Finanzas en Assets NS *5*
1.3.4 Módulo Cuentas a pagar y Cuentas cobrar en el Openbravo. *6*
1.4 METODOLOGÍA DE DESARROLLO DE SOFTWARE 6
1.4.1 Proceso Unificado de Desarrollo. *7*
1.4.2 Notación de Modelado de Procesos de Negocio. *9*
1.4.3 Modelo de Desarrollo Orientado a Componentes *10*
1.5 HERRAMIENTAS Y TECNOLOGÍAS 11
1.5.1 Herramientas CASE *11*
1.5.1.1 Visual Paradigm 11
1.5.2 ZendStudio Neon para PHP *12*
1.5.3 Apache *13*
1.5.4 Base de Datos postgreSQL *13*
1.5.5 TortoiseSVN *14*
1.5.6 Mozilla Firefox *15*
1.7.3 XML *15*
1.6 FRAMEWORK 16
1.6.1 Zend Framework *16*
1.6.1.1 Zend_Ext Framework 17
1.6.2 Doctrine Framework *17*
1.6.3 ExtJS *18*
1.7 LENGUAJES DE PROGRAMACIÓN 20
1.7.1 PHP *20*
1.7.2 HTML *21*
1.7.4 JavaScript *21*
1.8 ARQUITECTURA MODELO- VISTA- CONTROLADOR 21
1.9 IOC 22
1.10 CONCLUSIONES DEL CAPÍTULO I. 22

CAPITULO II: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA 24

2.1 INTRODUCCIÓN 24
2.2 VALORACIÓN CRÍTICA DEL DISEÑO PROPUESTO POR EL ANALISTA 24
2.2.1 Resultados de la evaluación de la métrica Tamaño Operacional de la clase (TOC). *26*
2.2.2 Resultados de la evaluación de la métrica Relaciones entre Clases (RC). *29*

2.2.3 Resultados de la evaluación de la métrica Profundidad de Herencia (PH). *32*
2.2.4 Resultados de la evaluación de la métrica Número de Descendientes (ND). *32*
2.2.4 Resultados de la evaluación de la métrica Número de Operaciones Redefinidas (NOR). *34*
2.3 ARQUITECTURA DE DISEÑO DEL MÓDULO 37
2.4 ESTÁNDARES DE CÓDIGOS USADOS EN EL MÓDULO. 40
2.4.1 Nomenclatura de las clases *40*
2.4.1.1 Nomenclatura según el tipo de clases 40
2.4.2 Nomenclatura de las funciones *41*
2.4.3 Nomenclatura de las variables *41*
2.4.4 Prefijos para los tipos de datos *41*
2.4.5 Nomenclatura de las constantes *42*
2.4.6 Nomenclatura de los atributos *42*
2.5 DESCRIPCIÓN DE LA ESTRUCTURA E IMPLEMENTACIÓN POR COMPONENTES 43
2.5.1 Componente Configuración. *44*
2.5.2 Componente Movimiento. *50*
2.5.3 Componente Arqueo. *53*
2.5.4 Componente Comunfinanzas *54*
2.6 ALGORITMOS COMPLEJOS. 54
2.6.1 Algoritmos Complejos del Módulo. *55*
2.6.2 Análisis de Complejidad de Algoritmos *56*
2.6.2.1 Cálculo de la Complejidad Ciclomática 57
2.7 DESCRIPCIÓN DE CLASES Y OPERACIONES DEL MÓDULO. 58
2.7.1 Principales clases del componente Configuración *58*
2.7.1.1 Clases del Modelo 58
2.7.1.2 Clases entidad. 58
2.7.2 Principales clases del componente Movimientos. *62*
2.7.2.1 Clase del modelo 62
2.7.2.2 Clases entidad 63
2.7.3 Principales clases del componente Arqueos. *66*
2.7.3.1 Clases del Modelo 66
2.7.3.2 Clases entidad 67
2.7.4 Principales clases del componente comunfinanzas. *69*
2.7.4.1 Clases del Modelo 69
2.7.4.2 Clases Entidad 70
2.8 CONCLUSIONES DEL CAPÍTULO II 74

CAPITULO III: VALIDACION DE LA SOLUCION PROPUESTA. 75

3.1 INTRODUCCIÓN 75
3.2 VALORACIÓN DE LA SOLUCIÓN 75
3.3 PRUEBAS DE SOFTWARE. 75
3.3.1 Objetivos *76*
3.3.2 Alcance. *77*

3.4 DESCRIPCIÓN DE LAS PRUEBAS DE CAJA BLANCA Y CAJA NEGRA. 77
3.4.2 Prueba de Caja Blanca. *78*
3.4.2.1 Prueba de camino básico. 79
3.4.2.2 Pruebas de cubrimiento. 79
3.4.2.3 Pruebas de condiciones. 79
3.4.2.4 Pruebas de bucles. 79
3.4.3 Pruebas de Caja Negra. *80*
3.4.3.1 Métodos de prueba basados en grafos 80
3.4.3.2 Partición equivalente 81
3.4.3.3 Análisis de valores límite 81
3.5 APLICACIÓN DE PRUEBAS DE CAJA BLANCA. 82
3.6 APLICACIÓN DE PRUEBAS DE CAJA NEGRA 87
3.7 CONCLUSIONES DEL CAPÍTULO III 89
CONCLUSIONES GENERALES. 90
RECOMENDACIONES 91
BIBLIOGRAFÍA 92
ANEXOS 94
GLOSARIO DE TÉRMINOS 128

INTRODUCCIÓN.

Durante años la contabilidad ha sido el método que dirige la economía de los países, pero el siglo XX dio un cambio en la forma en que se almacenaban los datos, hasta ese momento en hoja de papel, haciendo que los trabajadores encargados de contabilizar pasaran gran cantidad de horas en tareas engorrosas por el gran volumen de información. Para ello se fueron creando software de sistemas de planificación de recursos que se encargan de llevar en tiempo y forma todos los procesos empresariales de manera confiable, precisa y oportuna.

Nuestro país no está ajeno a este desarrollo y en las dos últimas décadas a dado un salto cuantitativo y cualitativo en busca de la informatización de la sociedad. Las empresas nacionales han incorporado software de gestión para la planificación de los recursos. Estos software son en su mayoría extranjeros, lo que provoca pérdidas millonarias a las entidades en pagos de licencia, además no son compatibles con las particularidades de la economía cubana. Otros software creados en nuestro país no son multiplataforma o no dan solución a todas las áreas de la entidad.

Una de las áreas de las empresas donde existe mayor necesidad de informatizar los procesos, es en las cajas. Las cajas se encargan de realizar movimientos de entrada y salida de dinero de la entidad, mediante el pago anticipos nacionales o extranjeros, además de realizar depósitos, reembolsos y arqueos. Estos procesos de Caja en su mayoría se realizan de forma manual, provocando pérdida de datos, lentitud en los procesos y haciendo que un mismo documento se opere varias veces. Por tales motivos se decidió realizar un sistema informático multiplataforma que integre los procesos de Caja.

Teniendo en cuenta la **situación problemática** planteada anteriormente, se traza el **problema científico** el cual sería: La carencia de un sistema informático que regule los procesos de Caja evitando la pérdida de datos por labores manuales y lentitud en los procesos.

Para dar solución al problema existente se propone como **objeto de estudio** los procesos contables de Caja. Paralelo al objetivo que persigue la presente investigación, el **campo de acción** sería la implementación de sistemas que gestionen los procesos de Caja en las empresas cubanas.

De manera que el **objetivo general** de la investigación, sería la implementación de sistemas de Caja donde se evite la pérdida de datos por labores manuales y lentitud en los procesos, utilizando una metodología y lenguaje que corresponda con las nuevas concepciones de informatización del país y que permita una realización eficiente de los pasos necesarios para efectuar dicha tarea.

Partiendo de la necesidad que en la actualidad enfrenta el proyecto CEDRUX con la gestión de los procesos de Caja, se trazó como **idea a defender**:

La implementación de un sistema informático de Caja, contribuirá a una rápida y eficiente gestión de los procesos de Caja y automatización de labores.

Para lograr un mejor desarrollo de la investigación y darle cumplimiento al objetivo trazado se plantearon las siguientes **tareas investigativas**:

- Confeccionar el marco teórico-conceptual de la investigación a partir de una búsqueda y revisión bibliográfica.
- Realizar un estudio de los artefactos entregados por el flujo de análisis y diseño del subsistema.
- Implementar el módulo para la automatización de los procesos que aquí se desarrollan.

Como **posible resultado** tendremos que logrando efectos deseados en el objetivo propuesto se espera obtener la implementación del módulo Caja del Sistema Integral de Gestión CEDRUX.

Los **métodos de investigación** empleados fueron los siguientes:

Métodos Empíricos: se realizaron **entrevistas** a trabajadores funcionales, especialistas en Contabilidad que actualmente llevan los procesos de gestión de actividades, para saber como funcionan las actividades relacionadas con el módulo Caja.

Métodos Teóricos se utilizó el método **Analítico-Sintético** pues se hizo una breve investigación y estudio de los sistemas de planificación de recursos para empresas y así identificar el problema concreto existente en el país.

El documento consta de 3 capítulos:

Capitulo 1 "Fundamentación Teórica": Se exponen conceptos necesarios para el entendimiento del software, se toman decisiones importantes para la realización del módulo, tales como la elección de los

lenguajes de programación, en este caso PHP y JavaScript, la base de datos a usar es PostgreSQL, la herramienta CASE Visual Paradigm, el navegador web Mozilla el lenguaje extensible XML. Además el uso del IoC para llamar a clases y métodos de otras clases.

Capitulo 2 "Descripción y Análisis de la solución propuesta": Se concluyó que la obtención del diseño propuesto por el analista fue muy importante desde el punto de vista que ayudo en gran medida a la correcta conformación de los componentes a implementar, así como todo lo relacionado con los requisitos no funcionales.

La arquitectura escogida fue la más apropiada, logrando satisfacer las funcionalidades y requerimientos del sistema. Esta estuvo caracterizada por la creación de un nuevo módulo que brinda funcionalidades comunes al resto de los módulos del subsistema.

Para finalizar este capítulo se realizó un análisis de los estándares de codificación, algoritmos complejos implementados y la importancia de estos para el correcto funcionamiento de la solución.

Capitulo 3 "Validación De La Solución": Se hace un bosquejo de distintos aspectos con gran significado para las pruebas de software, resaltándose la importancia de las mismas en el desarrollo de un proyecto. Se aborda acerca de las pruebas de menor escala tales como pruebas de caja blanca y caja negra y las particularidades de cada una de ellas, además de la ejecución de las mismas analizando los resultados obtenidos.

CAPITULO I: FUNDAMENTACION TEORICA

1.1 Introducción

En este capítulo se realiza un proceso investigativo de aspectos teóricos necesarios para la elaboración y concepción del Trabajo de Diploma. Se estudian y definen los conceptos para comprender el dominio del problema. Además se realiza un análisis del proceso de Caja en los Sistemas de Gestión Integral existente en Cuba y el mundo, se muestra un estudio del lenguaje de modelado gráfico a usar y la metodología de desarrollo escogida para guiar el avance del software. Por último se realiza un análisis de las herramientas y del lenguaje de programación seleccionados para llevar a cabo la elaboración del sistema.

1.2 Procesos de Sistemas 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.

1.3 Sistemas automatizados que poseen procesos de Caja en Cuba y el mundo.

1.3.1 Módulo de Finanzas y Caja en el Versat-Sarasola.

El programa Versat-Sarasola, es un sistema cubano de contabilidad confiable, permite enviar información referente a una empresa eficazmente, de forma inmediata, desde lugares apartados, a la vez que ofrece mayor organización, control y disciplina en cada gestión.

Fue éste el primer sistema de contabilidad cubano certificado, en cuya evaluación participaron el Ministerio de Finanzas y Precios, consultorías internacionales y el organismo encargado de la seguridad informática.

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. Posee un módulo de finanzas y caja que está integrado con algunos de los demás subsistemas del mismo, 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.

1.3.2 Módulo Finanzas en el sistema 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.

1.3.3 Módulo Finanzas en Assets NS

Dentro del plano internacional tenemos el sistema ASSET NS, el cual posee un subsistema de Finanzas que está dividido en módulos tales como Caja Chica y Flujo de Caja. Caja Chica se emplea para realizar gastos menores, como son, gastos de combustibles, dietas, gastos de viajes., además, para guardar los documentos generados en el proceso de cobros, hasta su depósito en banco, almacenar el dinero para el pago de nóminas y otros conceptos. Dentro de sus funcionalidades están que permite el

control de los fondos de efectivos, también realizar arqueos de caja que consiste en el análisis de las transacciones que afectan al fondo en un periodo.

Flujo de Caja permite que la Empresa programe sus necesidades financieras en un período a partir de los pronósticos de ventas, cobros, compras, pagos de compras, pagos de arrendamiento, pagos de salarios, dividendos, otros gastos planificados, presupuestos de ventas, presupuesto de producción. El Presupuesto de caja, suministra cifras que indican el saldo final en caja, pudiendo arrojar un déficit o excedente que le permite tomar decisiones a corto plazo. Permite que la Empresa programe sus necesidades de manera que los flujos que obtenga sean positivos, y que estos puedan ser invertidos, en caso contrario, planifique la forma de obtener financiamiento. (1)

Como desventaja señalar que no se ajusta a las características de la economía cubana, ejemplificado en la dualidad monetaria.

1.3.4 Módulo Cuentas a pagar y Cuentas cobrar en el Openbravo.

Es un sistema de gestión empresarial en software libre español, completamente funcional, integrado y basado en web. Openbravo posee varios subsistemas, dentro de ellos **Gestión económico-financiera** tiene diversos módulos como **cuentas a pagar y cuentas cobrar,** donde los procesos de caja *realizar arqueo* y *multicaja* están presentes en las funcionalidades **diario de caja** y **edición de caja** respectivamente. (2)

Luego de realizado un análisis de varios sistemas de Caja, tanto nacionales como extranjeros se arribó a la conclusión de que pese a existir sistemas muy completos y funcionales que resuelven los procesos; estos son en su gran mayoría software privativos cuyas licencias provocan pérdidas millonarias al país en un año fiscal. También en el estudio realizado se concluyó que los sistemas de Caja extranjeros presentaban problemas de compatibilidad con el sistema financiero cubano.

Entre los sistemas nacionales estudiados se constató que en su mayoría no son multiplataforma y sus gestores de base de datos son privativos. Luego de este análisis de los sistemas de caja estudiados se tomaron las mejores soluciones de ellos, para adaptarlas al sistema financiero cubano.

1.4 Metodología de Desarrollo de Software

Actualmente para desarrollar un proyecto con éxito, debe estar regido por metodología de desarrollo, la cual puede seguir uno o varios modelos de ciclo de vida, o sea, el ciclo de vida indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto.

Una metodología de desarrollo de software es un conjunto de técnicas, herramientas, procedimientos y soporte documental que permite a los desarrolladores definir los elementos necesarios para la construcción de un nuevo producto de software. Es la que durante el proceso de desarrollo del software define "quién esta haciendo qué, cuándo y cómo para alcanzar un determinado objetivo."(3)

Mediante la metodología de desarrollo de software se van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Actualmente existen varias metodologías de desarrollo de software, las cuales se deben estudiar con detenimiento para definir cual es la más adecuada a usar en el software a construir. Dichas metodologías se dividen en tres generaciones: desarrollo convencional, desarrollo estructurado y desarrollo orientado a objetos, esta última produjo una evolución de las metodologías orientadas a objetos para desarrollo de software de tiempo real, la cual se clasifica en ligeras o ágiles y pesadas o robustas.

Las metodologías orientadas a objetos en esencia identifican y organizan los conceptos del dominio de la aplicación y no tanto la representación final en un lenguaje de programación. Se caracterizan por ser iterativos e incrementales, pueden dividir varios sistemas en subsistemas independientes y se fomenta la reutilización de componentes.

1.4.1 Proceso Unificado de Desarrollo.

El Proceso Unificado de Desarrollo (Rational Unified Process, RUP por sus siglas en inglés), es una metodología de desarrollo de software robusta y orientada a objetos, basada en Lenguaje de Modelado Unificado (Unified Model Language, UML por sus siglas en inglés).

En RUP se han agrupado las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. Los 6 primeros son conocidos como flujos de ingeniería y los 3 últimos como de apoyo:

- Modelamiento del negocio
- Requerimientos
- Análisis y diseño
- Implementación
- Prueba (Testeo)
- Instalación
- Administración del proyecto
- Administración de configuración y cambios
- Ambiente

Además RUP se agrupa en 4 fases principales las cuales son:

- Conceptualización (Concepción o Inicio)
- Elaboración
- Construcción
- Transición

El proceso de desarrollo de software (RUP) tiene 3 características esenciales:

- Dirigido por casos de uso
- Centrado en la arquitectura
- Iterativo e Incremental

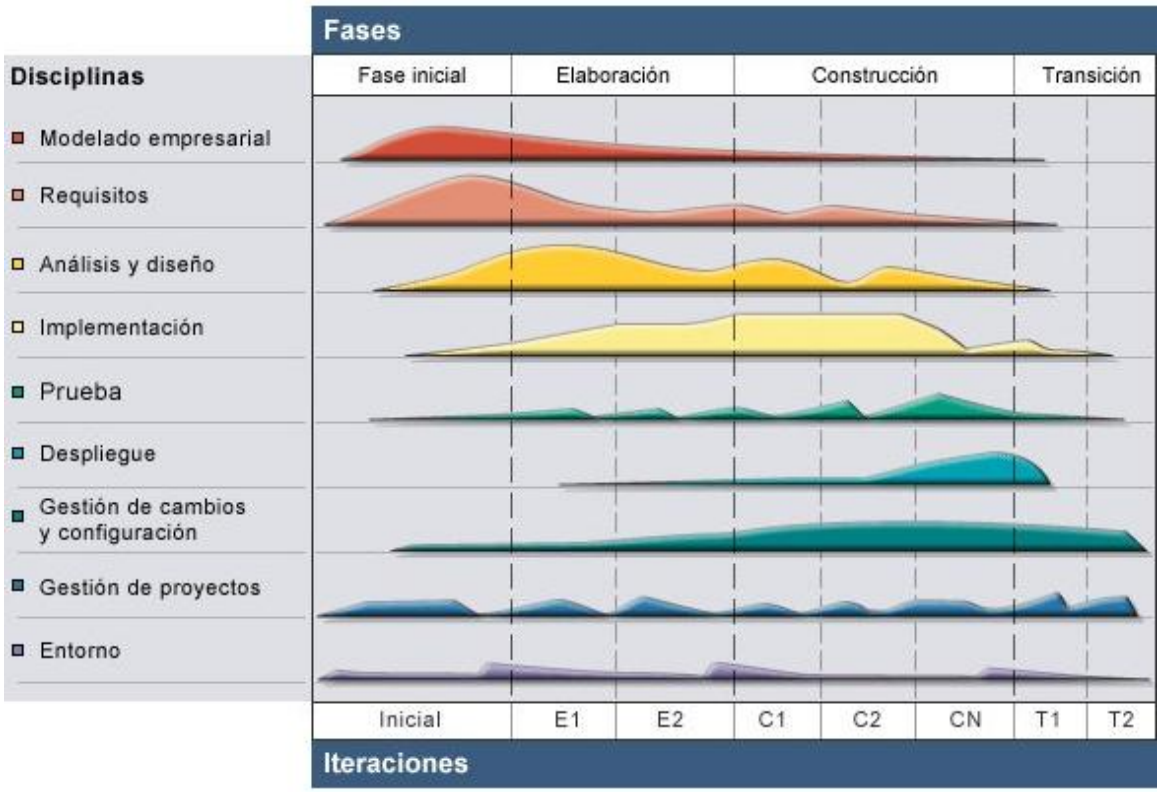


Figura 1.1: Ciclo de vida del software en RUP

1.4.2 Notación de Modelado de Procesos de Negocio.

Con la creciente certeza de que la eficiencia, la optimización y la gestión de los procesos son las claves del éxito, los desarrolladores de software han comenzado a usar la Notación de Modelado de Procesos de Negocio (Business Process Modeling Notation, BPMN por sus siglas en inglés) la cual proporciona un lenguaje común para que las partes involucradas puedan comunicar los procesos de forma clara, completa y eficiente. Siendo esta metodología de tipo ágil y orientada a objeto.

BPMN está diseñado para cubrir varios tipos de modelado y permite la creación, tanto de segmentos de proceso, como procesos de negocio de comienzo a fin, y en diferentes niveles de representatividad. Además se usa para comunicar una amplia variedad de información a diferentes audiencias.

La notación de modelado de procesos de negocio tiene como objetivos primarios:

- Proveer una notación entendible para cualquiera desde el analista del negocio, el desarrollador técnico y hasta la gente propia del negocio.
- Crear un puente estandarizado entre el diseño de procesos de negocio y su implementación.
- Asegurar que los lenguajes para la ejecución de procesos de negocio puedan ser visualizados con una notación común.

Lo que arroja a las ventajas siguientes:

- Define la notación y semántica de un BPD (Business Process Diagram)
- Provee la capacidad de entender los procedimientos internos en una notación gráfica y da a las organizaciones la habilidad de comunicarlos de una manera estándar.
- Mejora las capacidades de las notaciones de proceso de negocio tradicionales para manejar inherentemente los conceptos de procesos de negocio *business to business* (comunicaciones de comercio electrónico, en español).

El ciclo de diseño, análisis, ejecución y gestión de los procesos requiere que diferentes partes interactúen con los procesos, en diferentes momentos y de diferentes formas. BPMN ha sido creado para proporcionar una misma notación que sea comprensible tanto para los analistas como para los profesionales de las tecnologías de la información. Con BPMN los usuarios pueden crear diagramas usando una interfaz intuitiva que gestiona automáticamente muchas de las tareas de dibujo.

1.4.3 Modelo de Desarrollo Orientado a Componentes

La Orientación a Objetos introdujo, durante la década pasada, un cambio radical en el proceso de desarrollo de software. De un proceso caracterizado por su énfasis en la descripción algorítmica de la solución del problema, se pasó a un proceso orientado a la representación y manipulación de los objetos que caracterizan el problema. Este paradigma abrió, también, nuevas posibilidades para desarrollar software basado en la noción de reutilización de componentes. La Orientación a Objetos creó un rumbo diferente en el proceso de reutilización a través de la producción de componentes genéricos, fáciles de integrar, distribuidos e independientes de las plataformas de desarrollo. En este artículo, se discuten los conceptos fundamentales de la reutilización de software, así como los modelos de procesos y los

aspectos metodológicos que caracterizan esta nueva manera de producir software, denominada **Desarrollo de Software Orientado a Componentes**.

Por la complejidad en los procesos de negocio y tamaño de Caja, la visión de que se integre a otros subsistemas en un Sistema de Gestión Integral, y por el cúmulo de información que almacenan y analizan, requiere de una largo período de elaboración, construcción y de coordinación entre los equipos de desarrollo.

Para el desarrollo de un proyecto de esta magnitud es necesario que cada uno de los equipo de desarrollo posean un modelo estandarizado, así como una definición clara y precisa de las responsabilidades de cada uno de los roles que se ven involucrados en el desarrollo de la solución.

1.5 Herramientas y Tecnologías

1.5.1 Herramientas CASE

Las Herramientas de Ingeniería de Software Asistida por Computadoras (Computer Aided Software Engineering, CASE por sus siglas en inglés) son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y dinero. Dentro de sus objetivos están realizar el diseño del proyecto, cálculo de costo, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación y detección de errores.

1.5.1.1 Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite crear diagramas de clases, generar código desde diagramas y documentación.

Visual Paradigm se integra al Entorno de Desarrollo Integrado (Integrated Development Environment, IDE por sus siglas en inglés) de Eclipse. Está diseñado para desarrollar software con Programación Orientada a Objetos, reduce la duración del ciclo de desarrollo brindando ayuda tanto a arquitectos,

analistas, diseñadores como a desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores.

Dentro de sus características fundamentales están:

- Multiplataforma
- Interoperabilidad
- Modelamiento de los Requisitos
- Colaboración de Equipo
- Generación de Documentación
- Editor de Detalles de Casos de Uso
- Ingeniería de Código
- Modelado de Procesos de Negocio
- Integración con Entornos de Desarrollo
- Modelamiento de Bases de Datos

1.5.2 ZendStudio Neon para PHP

Zend Studio Neon es un completo entorno integrado de desarrollo para el lenguaje de programación PHP. Está escrito en Eclipse, y disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Está diseñado para usarse con el lenguaje PHP; sin embargo ofrece soporte básico para otros lenguajes Web, como HTML, Javascript y XML.

Sus principales características son:

- No requiere la instalación previa de PHP ni del entorno de ejecución de Eclipse.
- Soporte para PHP 4 y PHP 5.
- Resaltado de sintaxis, autocompletado de código, ayuda de código y lista de parámetros de funciones y métodos de clase.
- Inserción automática de paréntesis y corchetes de cierre.
- Emparejamiento de paréntesis y corchetes.
- Detección de errores de sintaxis en tiempo real.
- Funciones de depuración.

- Manual de PHP integrado.
- Soporte para navegación en bases de datos y ejecución de consultas SQL.(4)

1.5.3 Apache

Apache, es un servidor de protocolo para la transferencia de hipertextos (Hypertext Transfer Protocol, HTTP por sus siglas en inglés) de software libre para plataformas Unix, Windows, y Macintosh, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo. (5)

Las características que lo definen son las siguientes:

- Multiplataforma
- Es un servidor de web conforme al protocolo HTTP/1.1
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Basado en hebras en la versión 2.0.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor. (6)

1.5.4 Base de Datos postgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos de software libre, dentro de sus características destacan las siguientes:

Alta concurrencia

Mediante un sistema denominado MVCC (Acceso concurrente multiversión, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo

cambios. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases de datos, eliminando la necesidad del uso de bloqueos explícitos.

Amplia variedad de tipos nativos

PostgreSQL suministra nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas)
- Direcciones IP (IPv4 e IPv6).
- Arrays.

Como características adicionales tiene:

Claves ajenas también denominadas Llaves ajenas o **Claves Foráneas**

Disparadores: Un disparador se define en una acción específica basada en algo ocurrente dentro de la base de datos. En PostgreSQL esto significa la ejecución de un procedimiento almacenado basado en una determinada acción sobre una tabla específica.

Todos los disparadores se definen por seis características:

- El nombre del disparador.
- El momento en que el disparador debe arrancar.
- La tabla donde el disparador se activara.
- La frecuencia de la ejecución.
- La función que podría ser llamada.

PostgreSQL posee integración con un gran número de lenguajes tales como PHP, Java, C, C++, etc.

1.5.5 TortoiseSVN

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. TortoiseSVN maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en

un repositorio central. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio. (7)

Las principales características de TortoiseSVN son:

- **Integración con el shell de Windows**: TortoiseSVN se integra perfectamente en el shell de Windows (por ejemplo, el explorador). Esto significa que puede seguir trabajando con las herramientas que ya conoce.
- **Iconos sobreimpresionados:** El estado de cada carpeta y fichero versionado se indica por pequeños iconos sobreimpresionados. De esta forma, puede ver fácilmente el estado en el que se encuentra su copia de trabajo.
- **Fácil acceso a los comandos de Subversion**: Todos los comandos de Subversion están disponibles desde el menú contextual del explorador. TortoiseSVN añade su propio submenú allí.

1.5.6 Mozilla Firefox

Mozilla Firefox es un navegador de Internet libre y de código abierto. Es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva, marcadores dinámicos y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. Además se pueden añadir funciones a través de complementos desarrollados por terceros.

Las características de Mozilla Firefox son las siguientes:

- Multiplataforma.
- Cuenta con una protección antiphishing, antimalware e integración con el antivirus.
- La navegación por pestañas.
- Bloqueador de ventanas emergentes.
- Multiples Extensiones.
- Incluye de serie un buscador integrado en la interfaz que hace búsquedas en Google.
- Posee gestor de descargas.
- Utiliza el sistema SSL para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTPS.

1.7.3 XML

El Lenguaje de Etiquetado Extensible (Extensible Markup Language, XML por sus siglas en inglés) es muy simple, pero estricto, pues juega un papel fundamental en el intercambio de una gran variedad de datos. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Dicha tecnología es un conjunto de módulos que ofrece servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información. Siendo su función principal es describir datos y no mostrarlos como es el caso de HTML. (8)

1.6 Framework

Es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

1.6.1 Zend Framework

Zend Framework se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es de código abierto y trabaja con PHP 5.

Dentro de sus principales características están:

- Trabaja en 3 capas, usando el Modelo Vista Controlador.
- Cuenta con módulos para manejar archivos en formato de documento portátil, canales de sindicación de noticias y servicios web.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar la base de datos.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.
- Clientes para servicios web. (9)

1.6.1.1 Zend_Ext Framework

Es un framework de código abierto, diseñado para PHP 5 o superior. Se deriva de Zend Framework por lo que cumple con todas sus características. Posee un controlador vertical para el control de las acciones realizadas por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor, y se le agregó el IoC para la integración entre los módulos o componentes. Tiene incorporado el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos y el ExtJS Framework para el desarrollo de las vistas.

1.6.2 Doctrine Framework

Framework Doctrine es un potente y completo sistema de mapas de relaciones de objetos (Object Relational Mapper, ORM por sus siglas en ingles) para PHP 5.2 o superior con una base de datos con capas de abstracción incorporada.

Entre muchas otras cosas tienes la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente. Es fácilmente integrado a los principales frameworks de desarrollo utilizados actualmente, por lo que se propone su uso.

A continuación se representan las clases fundamentales a representar en el modelo. Las mismas son generadas automáticamente.

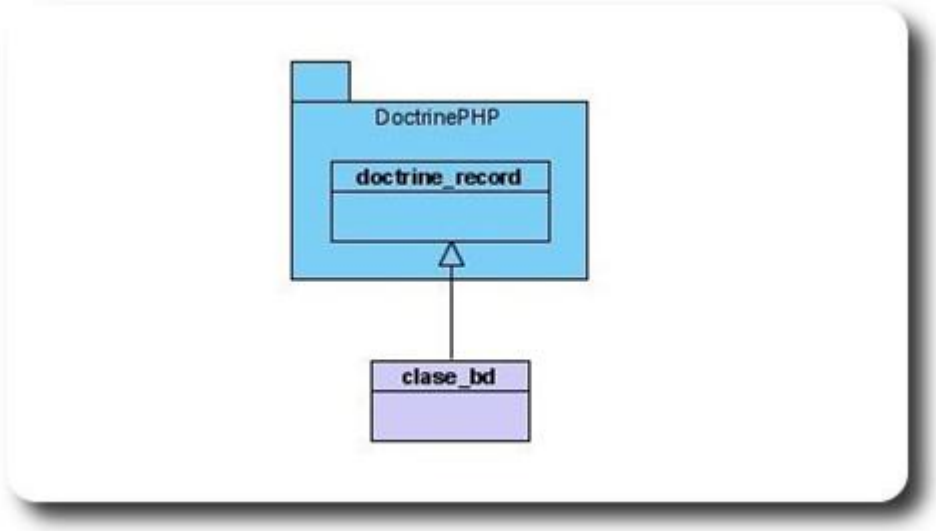


Figura 1.2 Diagrama de Clases Genérico para Doctrine PHP

1.6.3 ExtJS

ExtJS es una librería construida con JavaScript que proporciona una interfaz cuya potencia radica en la rica colección de componentes para el diseño de interfaces del lado del cliente.

Tiene incluidos la mayoría de los controles de los formularios Web incluyendo tablas para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería de componentes incluye componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX. Presenta el uso de JavaScript con una programación orientada a objetos.

Clase	Descripción
ext-base	Encargada del manejo de las solicitudes y respuestas, trabajo con AJAX y manejo de componentes de EXT. Está incluida en el paquete original.
ext-all	Es la encargada de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae EXT JS.
Vista	Representa la vista que se muestra al usuario.
js_vista	Fichero JS con las funciones Java Script asociadas a la vista. Aquí se

	establece la referencia a las clases de EXT.

En el siguiente diagrama de clases se representa el uso de EXT en la vista de la aplicación resaltando las clases fundamentales utilizadas de entre el resto.

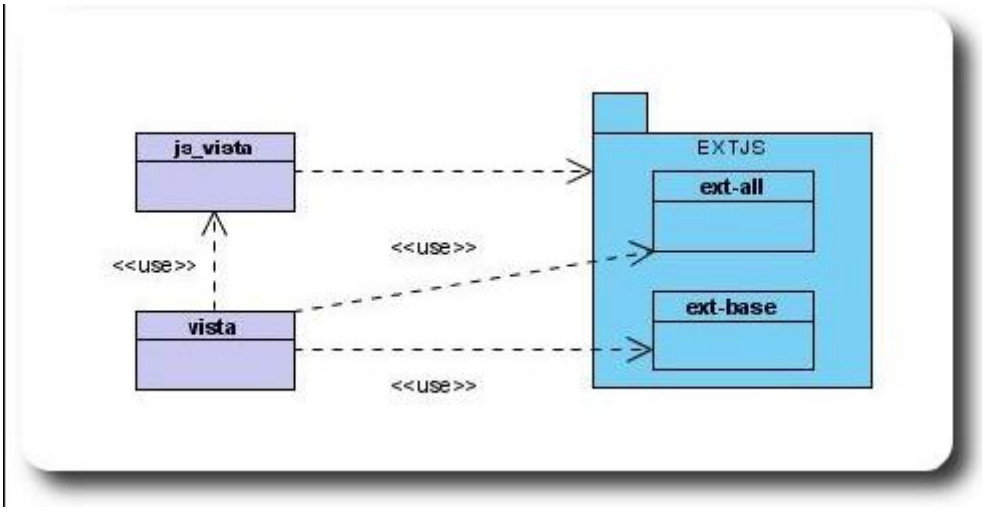


Figura 1.3: Diagrama de Clases Genérico para EXTJS

JavaScript 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, posibilitando realizar cambios sobre una página sin necesidad de recargarla, aumentando de esta forma la interactividad, velocidad y usabilidad de la misma.

Algunas tecnologías por la que está compuesto AJAX son:

- XHTML y CSS, para crear una presentación basada en estándares.
- DOM, para la interacción y manipulación dinámica de la presentación.
- XML, XSLT y JSON, para el intercambio y la manipulación de información.
- XMLHttpRequest, para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

Sus características principales son:

- Mayor rapidez e interactividad, al estilo aplicaciones de escritorio.

- Reduce significativamente la carga de información continua del servidor, actualizando solamente porciones de la página.
- Reduce de manera significativa los tiempos de carga inicial.

1.7 Lenguajes de programación

Un lenguaje de programación es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. (10)

1.7.1 PHP

PHP es un lenguaje de programación interpretado, diseñado originalmente para la creación de páginas web dinámicas. Es usado principalmente en interpretación del lado del servidor pero actualmente puede ser utilizado desde una interfaz de línea de comandos o en la creación de otros tipos de programas incluyendo aplicaciones con interfaz gráfica usando distintas bibliotecas. El Pre-procesador de hipertextos (Hypertext Pre-processor, PHP por sus siglas en inglés) inicialmente se llamó PHP Tools, siendo publicada bajo licencia de software libre.

Sus principales características son:

- Es un lenguaje multiplataforma.
- Soporte para una gran cantidad de bases de datos
- Integración con varias bibliotecas externas, permite generar documentos en PDF.
- Ofrece una solución simple y universal para las paginaciones dinámicas del Web de fácil programación.
- Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.
- Producto de código abierto
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones

1.7.2 HTML

El Lenguaje de Marcas de Hipertexto **(**HyperText Markup Language, HTML por sus siglas en ingles) es el lenguaje de marcado predominante para la construcción de páginas web. Usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

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

1.7.4 JavaScript

Javascript es un lenguaje de programación interpretado que permite a los desarrolladores crear acciones en sus páginas web. Es utilizado para crear pequeños programas que luego son insertados en una página web y en programas más grandes, orientados a objetos pero mucho más complejos.

Dentro de sus principales características están:

- Multiplataforma.
- Orientada a objetos.
- Se ejecuta del lado del cliente.
- Compatible con la mayoría de los navegadores. (11)

1.8 Arquitectura Modelo- Vista- Controlador

El Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la interface de usuario y el código es el que provee de datos dinámicos a la página; el modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio; y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- **Modelo**: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite derivar nuevos datos; por ejemplo, no

permitiendo comprar un número de unidades negativo, calculando si hoy es el cumpleaños del usuario o los totales, impuestos o importes en un carrito de la compra.

- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- **Controlador**: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. (12)

1.9 IoC

Inversión de control (Inversion of Control, IoC por sus siglas en inglés,) es un concepto junto a unas técnicas de programación en las que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa haciendo llamadas a procedimientos o funciones. Tradicionalmente el programador especifica la secuencia de decisiones y procedimientos que pueden darse durante el ciclo de vida de un programa mediante llamadas a funciones. En su lugar, en la inversión de control se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que algún tipo de entidad o arquitectura externa lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir.

El flujo habitual se da cuando es el código del usuario quien invoca a un procedimiento de una librería. La inversión de control sucede cuando es la librería la que invoca el código del usuario.

Típicamente sucede cuando la librería es la que implementa las estructuras de alto nivel y es el código del usuario el que implementa las tareas de bajo nivel. (13)

1.10 Conclusiones del capítulo I.

En este capitulo se exponen conceptos necesarios para el entendimiento del software, se toman decisiones importantes para la realización del módulo, tales como la elección de los lenguajes de programación, en este caso PHP y JavaScript, la base de datos a usar es PostgreSQL, la herramienta CASE Visual Paradigm, el navegador web Mozilla el lenguaje extensible XML. Además el uso del IoC para llamar a clases y métodos de otras clases.

CAPITULO II: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1 Introducción

En el desarrollo de este capitulo se hace una valoración critica del diseño propuesto por el analista, se realiza una descripción de los principales procesos que se identificaron para automatizar. Se confeccionan los prototipos de interfaz de la aplicación acordes a las necesidades del sistema y se detalla la estructura del módulo, desglosándolo por componentes; además se describe uno de los principales algoritmos no triviales a implementar.

2.2 Valoración crítica del diseño propuesto por el analista

A partir del diseño propuesto por los analistas (*ver anexo 1*) para la solución del módulo Caja, se llegó a una mejor comprensión de los requisitos, logrando de esta manera adentrarse en las especificidades de los requisitos no funcionales, las restricciones de los lenguajes de programación, las características de la programación en capas y las ventajas que da la multiplataforma. También se logró buscar un punto congruente para iniciar la implementación de las clases y comprender las tecnologías de interfaz de usuario necesarias.

Durante la evaluación a la solución propuesta también se aplicaron métricas de diseño expuestas por Pressman en su libro “Un enfoque Práctico”:

- Responsabilidad. Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad del diseño. Consiste en la complejidad que posee una estructura de diseño de clases.
- Complejidad de implementación. Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización. Consiste en el grado de reutilización de presente en una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, esta muy ligada a la característica de Reutilización.

- Complejidad del mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.
- Cantidad de pruebas. Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.
- Nivel de Cohesión. Consiste en el grado de especialización de las clases concebidas para modelar un dominio o concepto específico.
- Abstracción del diseño. Consiste en la capacidad de modelar lo más cercano posible a la realidad un concepto o dominio determinado. (17)

Las métricas aplicadas a la solución del módulo fueron las siguientes:

Tamaño operacional de clase (TOC): Está dado por el número de métodos asignados una clase. En la misma se evalúan los siguientes atributos de calidad.

- **Responsabilidad:** El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
- **Complejidad de implementación**: El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
- **Reutilización**: Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.

Relaciones entre clases (RC): Está dado por el número de relaciones de uso de una clase con otras.

- **Acoplamiento:** El aumento del RC provoca un aumento del Acoplamiento de la clase.
- **Complejidad del mantenimiento**: El aumento del RC provoca un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización**: El aumento del RC provoca una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas**: El aumento del RC provoca un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Profundidad de Herencia (PH): Está dado por la profundidad en al herencia de las clases heredades de un nodo padre.

- **Complejidad del mantenimiento**: El aumento del PH provoca una disminución de la complejidad del mantenimiento de la clase.
- **Complejidad del diseño:** El aumento del PH provoca una disminución de la complejidad del diseño de la clase.
- **Reutilización:** El aumento del PH provoca un aumento en el grado de reutilización de la clase.

Número de Descendientes (ND): Está dado por la cantidad de clases que heredan de un padre.

- **Abstracción del diseño:** El aumento del ND puede disminuir la abstracción del diseño de clases.
- **Cantidad de pruebas:** El aumento del ND implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
- **Reutilización**: El aumento del ND implica un aumento en el grado de reutilización de la clase.
- **Nivel de Cohesión**: El aumento del ND implica una disminución en el nivel de cohesión de la clase.

Número de Operaciones Redefinidas para una clase hija (NOR): Está dado por la cantidad de operaciones redefinidas en cada clase hija.

- **Abstracción del diseño:** El aumento del NOR implica que se ha logrado una buena definición de la abstracción del diseño de clases.
- **Cantidad de pruebas:** El aumento del NOR implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.
- **Complejidad del mantenimiento:** El aumento del NOR implica un aumento de la complejidad del mantenimiento de la clase.

Seguidamente se muestran los resultados obtenidos de la aplicación de los instrumentos de evaluación para medir las métricas descritas anteriormente además de una valoración de los mismos.

2.2.1 Resultados de la evaluación de la métrica Tamaño Operacional de la clase (TOC).

Ver instrumentos y tabla de resultados en (Anexo 14 Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

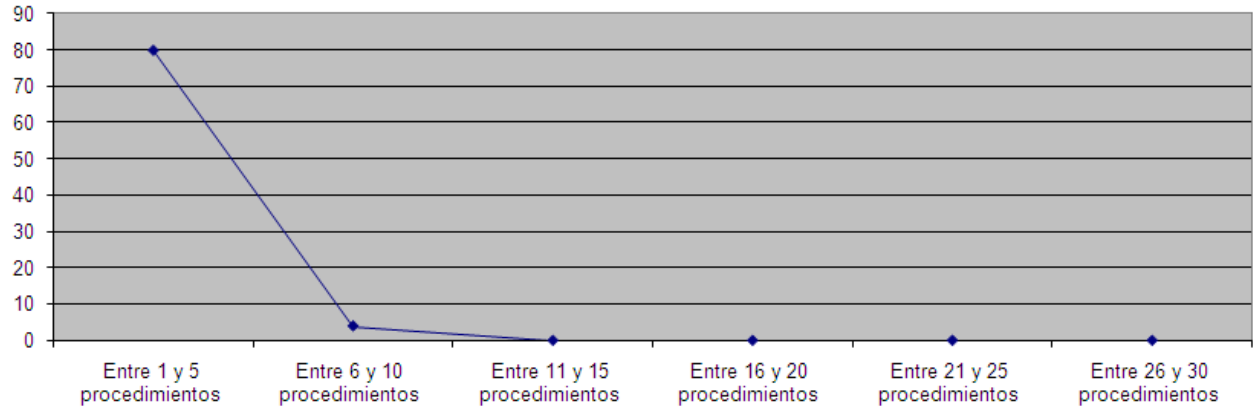


Figura 2.1 Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

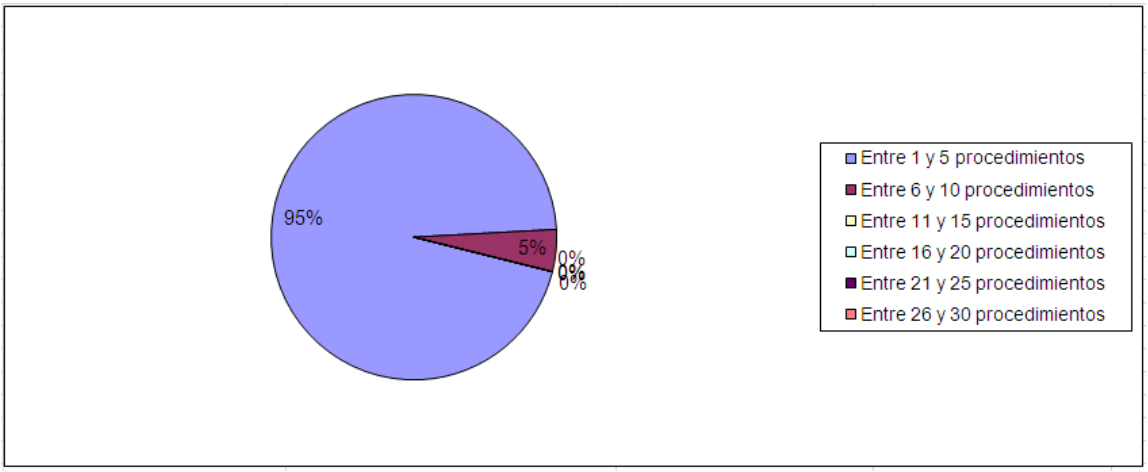


Figura 2.2: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

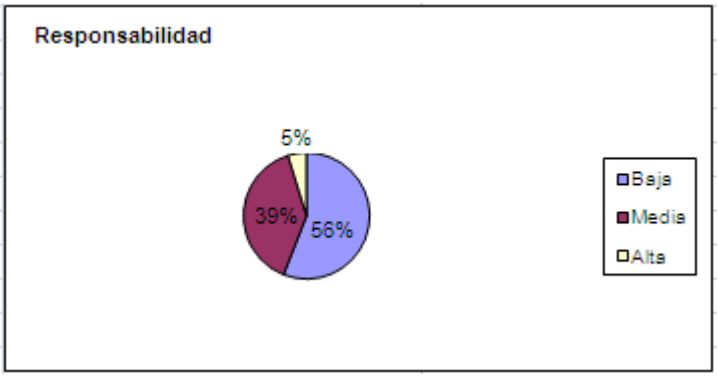


Figura 2.3: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

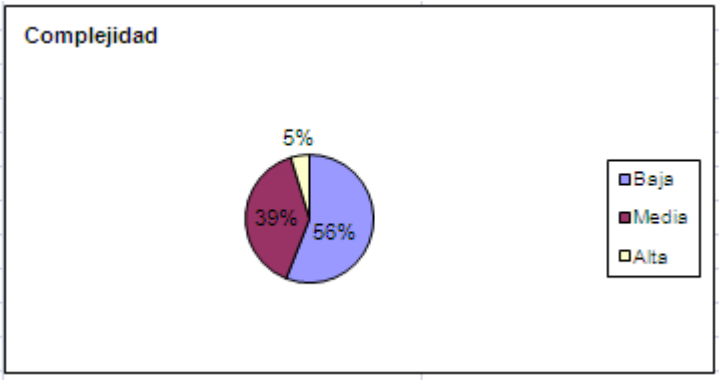


Figura 2.4 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

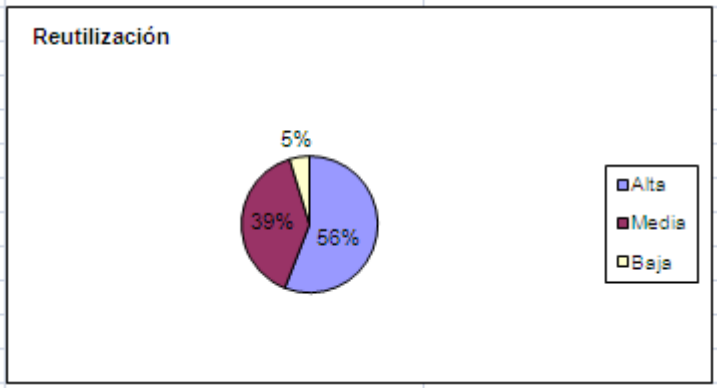


Figura 2.5 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo

Reutilización.

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del módulo Caja tiene una buena calidad pues un 95 % de las clases incluidas en estos componentes posee menos cantidad de operaciones que el doble del promedio registrado en las mediciones. También el 95% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización).

2.2.2 Resultados de la evaluación de la métrica Relaciones entre Clases (RC).

Ver instrumentos y tabla de resultados en (Anexo 15 Instrumento de medición de la métrica Relaciones entre clases (RC)).

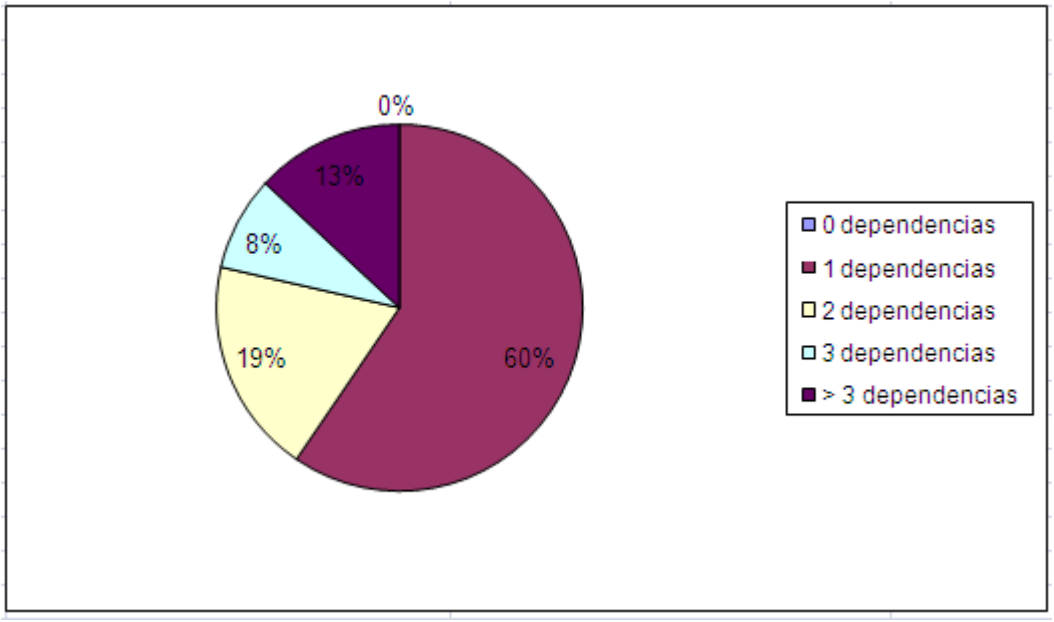


Figura 2.6 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

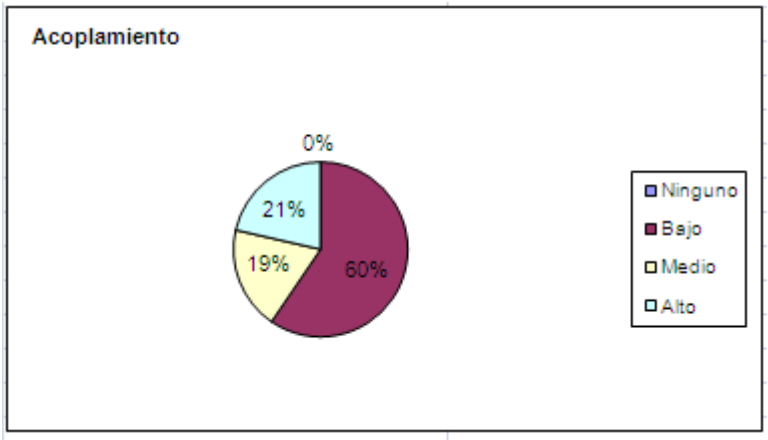


Figura 2.7 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

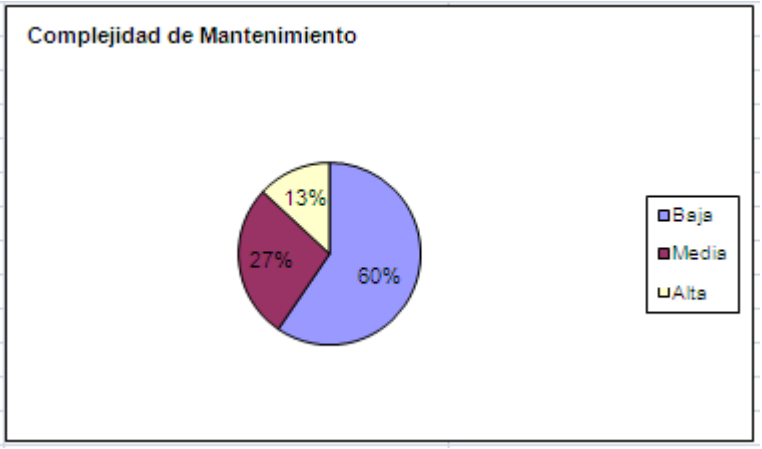


Figura 2.8 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

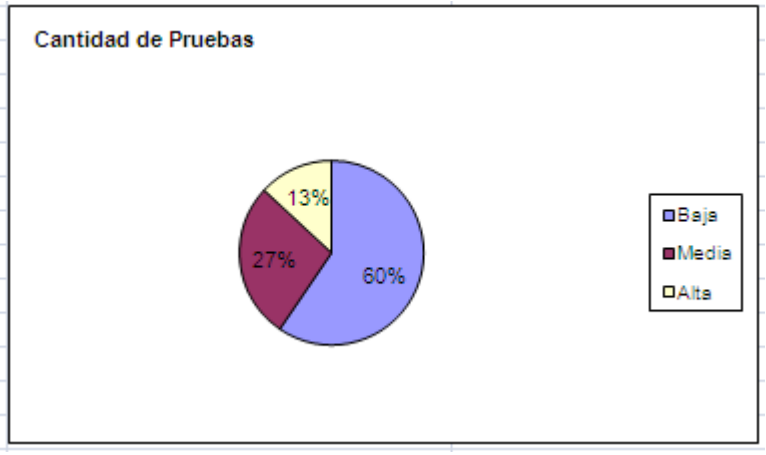


Figura 2.9 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

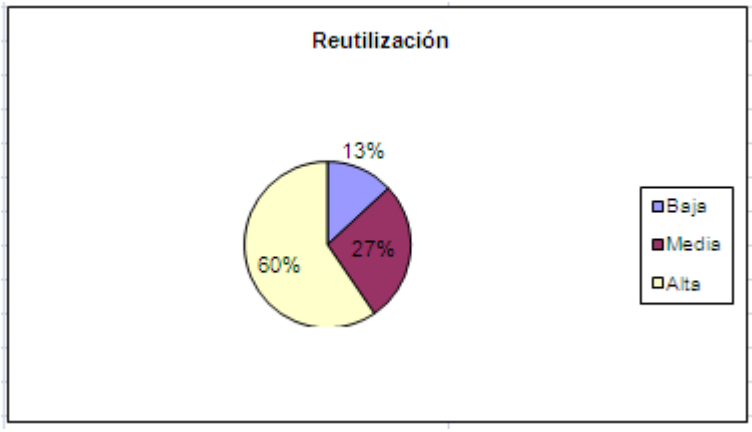


Figura 2.10 Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño del módulo Caja tiene una calidad aceptable para realizar la implementación, teniendo en cuenta que el 79% de las clases incluidas en el módulo poseen menos de 3 dependencias de otras clases. Además el 79% poseen indicadores aceptables referentes al acoplamiento entre clases. Por último los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización poseen niveles aceptables en un 87 % de las clases.

2.2.3 Resultados de la evaluación de la métrica Profundidad de Herencia (PH).

Ver instrumentos y tabla de resultados en (Anexo 16 Instrumento de medición de la métrica Profundidad de Herencia (PH)).

Figura 2.11 Representación en % de los resultados obtenidos en el instrumento agrupados por nivel.

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica PH, se puede concluir que el diseño del módulo Caja tiene una calidad aceptable teniendo en cuenta que la profundidad predominante de la herencia en el módulo desarrollado es entre 1 y 2.

2.2.4 Resultados de la evaluación de la métrica Número de Descendientes (ND).

Ver instrumentos y tabla de resultados en (Anexo 17 Instrumento de medición de la métrica Número de Descendientes (ND)).

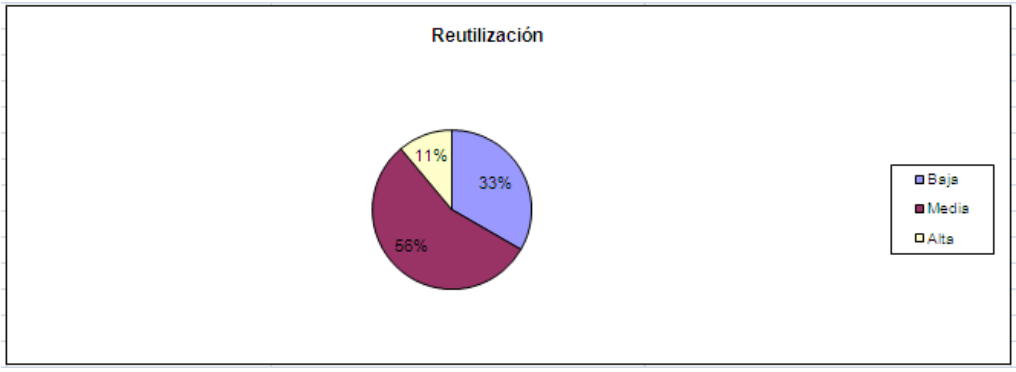


Figura 2.12 Representación de la incidencia de los resultados de la evaluación de la métrica ND en el atributo Reutilización.

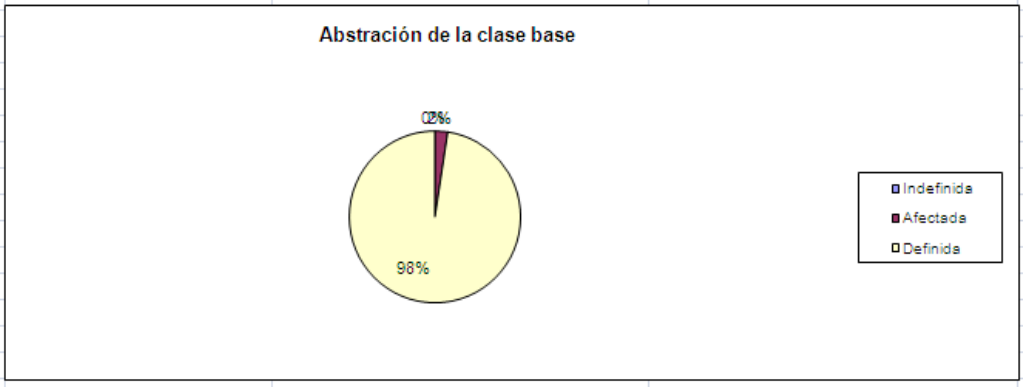


Figura 2.13 Representación de la incidencia de los resultados de la evaluación de la métrica ND en el atributo Abstracción de la clase base.

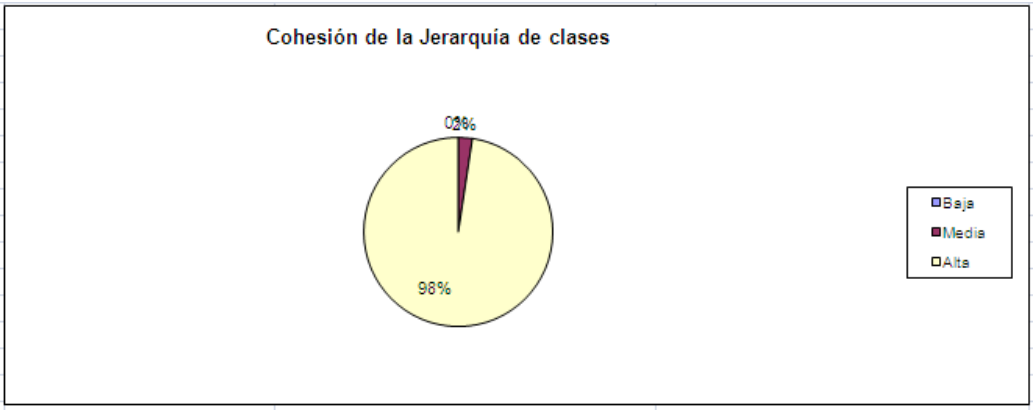


Figura 2.14 Representación de la incidencia de los resultados de la evaluación de la métrica ND en el atributo Cohesión de la Jerarquía de clases.

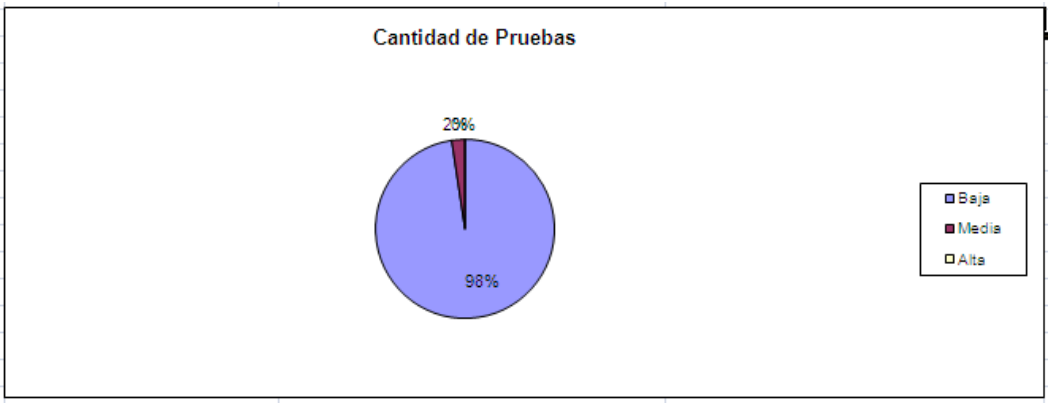


Figura 2.15 Representación de la incidencia de los resultados de la evaluación de la métrica ND en el atributo Cantidad de Pruebas.

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica ND, se puede concluir que el diseño del módulo Caja tiene una calidad aceptable teniendo en cuenta que solo se emplea la herencia en casos bien identificados a partir de las necesidades del negocio o del diseño. Solo 6 clases superan el promedio de descendientes, en el resto de los casos están en los valores indicados. Analizando el atributo de calidad Reutilización se puede decir que por lo antes explicado los índices de Reutilización son medios representando un 67%. Por otro lado el atributo Abstracción de la clase base muestra una tendencia a la conservación de la abstracción. Por ultimo la Cohesión de la jerarquía de clases como para el atributo Cantidad de Pruebas los índices son buenos, favoreciendo así el diseño

.

2.2.4 Resultados de la evaluación de la métrica Número de Operaciones Redefinidas (NOR).

Ver instrumentos y tabla de resultados en (Anexo 18 Instrumento de medición de la métrica Número de Operaciones Redefinidas (NOR)).

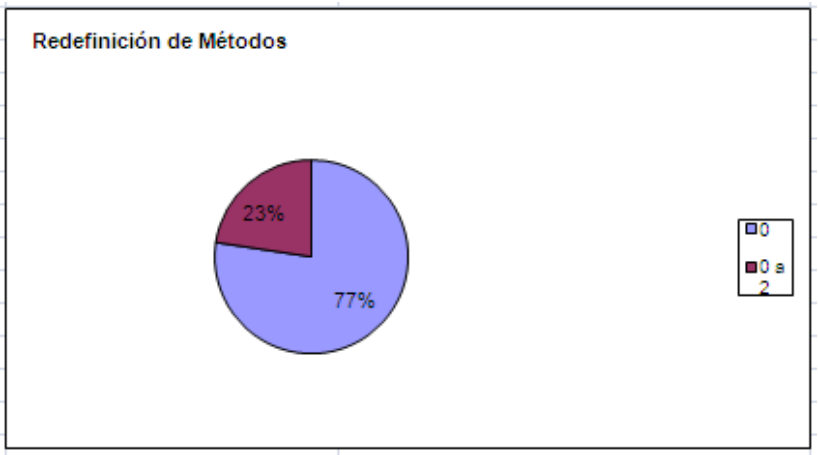


Figura 2.16 Representación en % de los resultados obtenidos en el instrumento agrupados en los valores existentes.

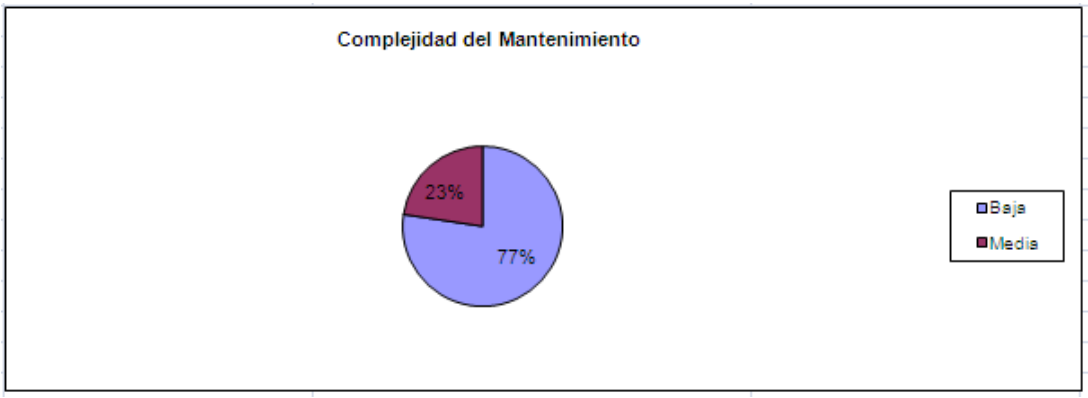


Figura 2.17 Representación de la incidencia de los resultados de la evaluación de la métrica NOR en el atributo Complejidad del Mantenimiento.

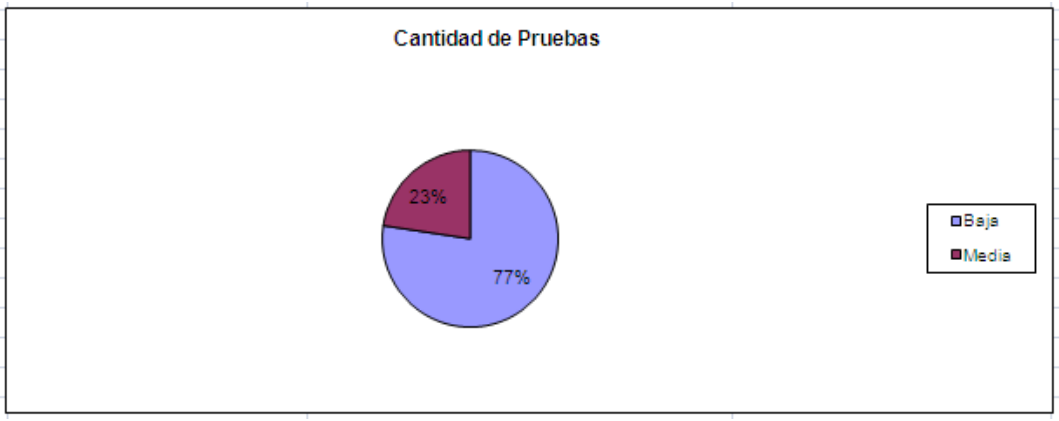


Figura 2.18 Representación de la incidencia de los resultados de la evaluación de la métrica NOR en el atributo Cantidad de Pruebas

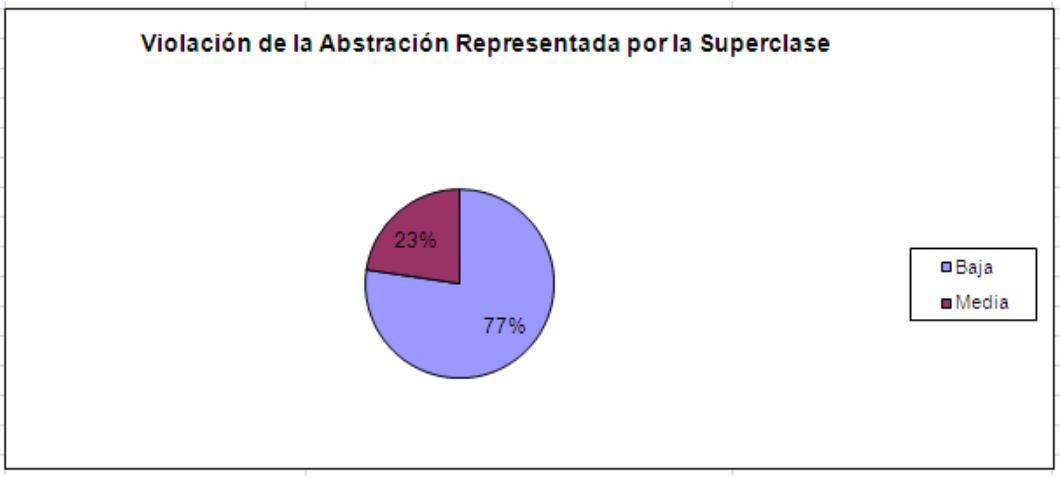


Figura 2.19 Representación de la incidencia de los resultados de la evaluación de la métrica NOR en el atributo Violación de la Abstracción representada por la superclase

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica NOR, se puede concluir que el diseño del módulo Caja tiene una calidad aceptable debido a que solo el 23 % de las clases incluidas en el análisis relacionadas con la herencia redefinen algún método heredado. También los atributos de calidad Complejidad del Mantenimiento, Cantidad de Pruebas y Violación de la Abstracción representada por la superclase se comportan de forma adecuada.

Resumiendo los resultados obtenidos en las métricas aplicadas, se muestra la siguiente tabla:

	TOC	PH	RC	ND	NOR	Total
Complejidad Implementación	0,5	1				0,75
Reutilización	0,5	1	0,5	0		0,5
Acoplamiento			0,5			0,5
Complejidad Mantenimiento		1	0,5		1	0,833333333
Cantidad de Pruebas			0,5	1	1	0,833333333
Abstracción				1	1	1
Cohesión	0,5			1		0,75
Calidad de Diseño					1	1
Total	0,5	1	0,5	0,75	1	

Umbrales		Met	AC
0.1 a 0.3	M	0	0
0.4 a 0.7	R	3	4
0.8 a 1	B	2	4

2.3 Arquitectura de diseño del módulo.

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.

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. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas.

La arquitectura de software define, de manera abstracta, los componentes, sus interfaces y la comunicación ente ellos, siendo el resultado ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema.

El proyecto CEDRUX aplica una arquitectura basada en capas. La arquitectura de n−capas brinda una gran cantidad de ventajas para los software que necesitan soluciones flexibles y fiables que resuelvan problemas complejos que cambien constantemente.

El patrón n-capas es uno de los más generalizados y utilizados en el desarrollo de aplicaciones web a nivel global y sencillo de implementar. Se resume en lograr:

- Organizar la estructura lógica de gran escala de un sistema en capas separadas de responsabilidades distintas y relacionadas, con una separación clara y cohesiva de intereses como que las capas "más bajas" son servicios generales de bajo nivel, y las capas más altas son más específicas de la aplicación.
- La colaboración y el acoplamiento es desde las capas más altas hacia las más bajas; se evita el acoplamiento de las capas más bajas a las más altas.

Sin embargo, muchos de los frameworks que se usan hoy en día están implementados bajo otros estilos arquitectónicos y no en 3 capas.

El proyecto CEDRUX decidió trabajar con el estilo Modelo-Vista-Controlador basado en un patrón de diseño que plantea la separación de diferentes clases en dependencia de la función que realizan de modo tal que sea posible manejar dinámicamente la forma en que se procesan solicitudes y se gestiona la manera en que se muestran resultados al usuario final. En otras palabras separa la presentación del dominio de la aplicación.

A simple vista ya se tienen ventajas; por ejemplo, una modificación en la Vista (Presentación) conlleva poco o ningún cambio tanto en el Controlador (Negocio) o en el Modelo y viceversa. También los métodos del Modelo (Acceso a Datos) pueden ser reutilizados por múltiples componentes al estar diseñado en formato modular.

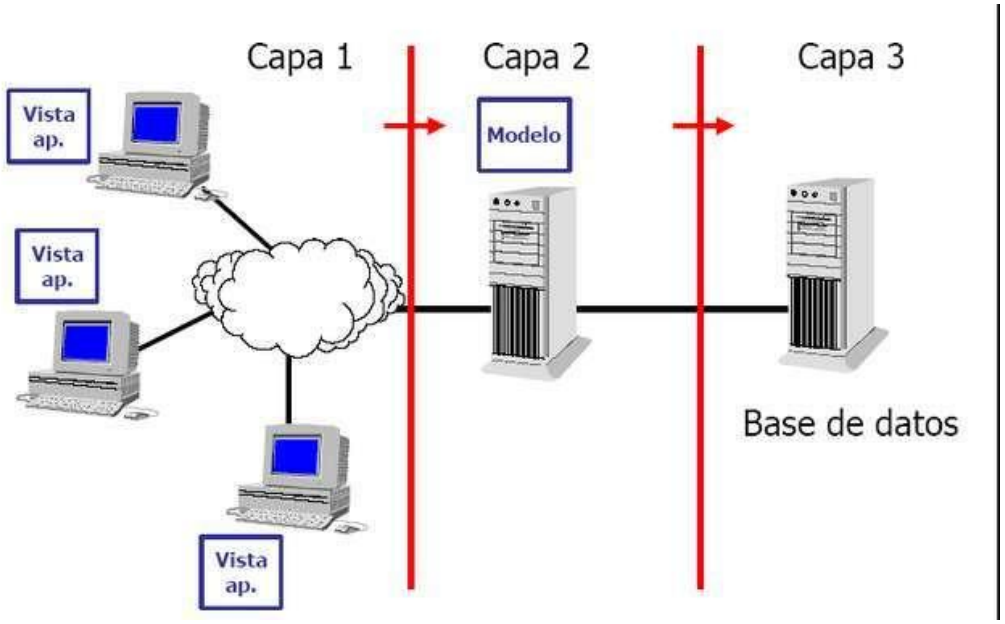


Figura 2.20. Diseño en 3 Capas.

Para dar cumplimiento a las necesidades del proyecto se adoptó para la Vista el ExtJS-Framework para JavaScript, el cual es muy utilizado en el desarrollo de aplicaciones Web con AJAX, además de tener una librería inmensa que permite configurar las interfaces Web de manera semejante a aplicaciones desktop. Otro Framework utilizado fue es ZendExt, el mismo hereda del Zend-Framework que es fácilmente integrable a las aplicaciones debido a su composición y a que contiene diferentes clases de gran utilidad, como por ejemplo en la búsqueda dinámica de ficheros a incluir o utilizar. El Zend-Framework es uno de los framework más utilizados para PHP y utiliza el estilo Modelo- Vista- Controlador como base de su funcionamiento.

Para agilizar el acceso a datos se utilizó Doctrine, que es un potente y completo sistema ORM (Mapeo Objeto Relacional) para PHP 5.2+ con un DBAL incorporado.

Una de las novedades en la implementación de este sistema es el uso del IoC, que se utiliza para acceder de un subsistema a otro o de un módulo a otro, evitando tablas repetidas en diferentes componentes. Por ejemplo, para lograr que el módulo de Caja se incluya en el subsistema Finanzas y se integre a los módulos Banco y Cobros Pagos como un todo, se hace necesario la utilización del IoC, pues el mismo permite que todos accedan a la información común, evitando datos duplicados.

2.4 Estándares de códigos usados en el módulo.

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 mantenibilidad 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.

2.4.1 Nomenclatura de las clases

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

Ejemplo: *GestionarCaja*

2.4.1.1 Nomenclatura según el tipo de clases

- Clases controladoras

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

Ejemplo: *GestionarCajaController*

- 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 de Datos.

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

2.4.2 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*, y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: inserta*rCaja*

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

Ejemplo: *insertarCajaAction*

2.4.3 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*

2.4.4 Prefijos para los tipos de datos

Los prefijos a utilizar en la creación de variables serán los siguientes:

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str

float	flt
Boolean	Boo

2.4.5 Nomenclatura de las constantes

El nombre a emplear para las constantes se escribe con todas las letras en mayúscula.

Ejemplo: *MENUDO*.

2.4.6 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

La utilización de los estándares establecidos en el proyecto tuvo gran importancia, pues facilitó el trabajo a realizar y mejoró el entendimiento entre los integrantes del equipo de desarrollo del módulo. Con las normas y estándares de codificación se logró una mejor comprensión del código y el futuro mantenimiento del mismo a largo plazo.

2.5 Descripción de la Estructura e Implementación por Componentes.

El módulo Caja se desglosa en varios componentes: **Configuración**, **Movimientos**, **Arqueos y Cierre**, estos poseen dentro varias funcionalidades que dan una solución eficiente a los procesos de Caja.

Dentro del subsistema Finanzas existe un componente **comunfinanzas** que integra necesidades de los módulos de Caja, Banco y Cobros Pagos. Este componente se conformó para dar una solución que sirviera a todos los módulos. Por ejemplo en el caso de los procesos de Caja, era necesario un documento que justificara cada movimiento realizado en un fondo. La solución genérica que se logró con el **comunfinanzas** permitirá también que Cobros Pagos y Banco utilicen esos documentos sin tener que repetir los datos en cada módulo.

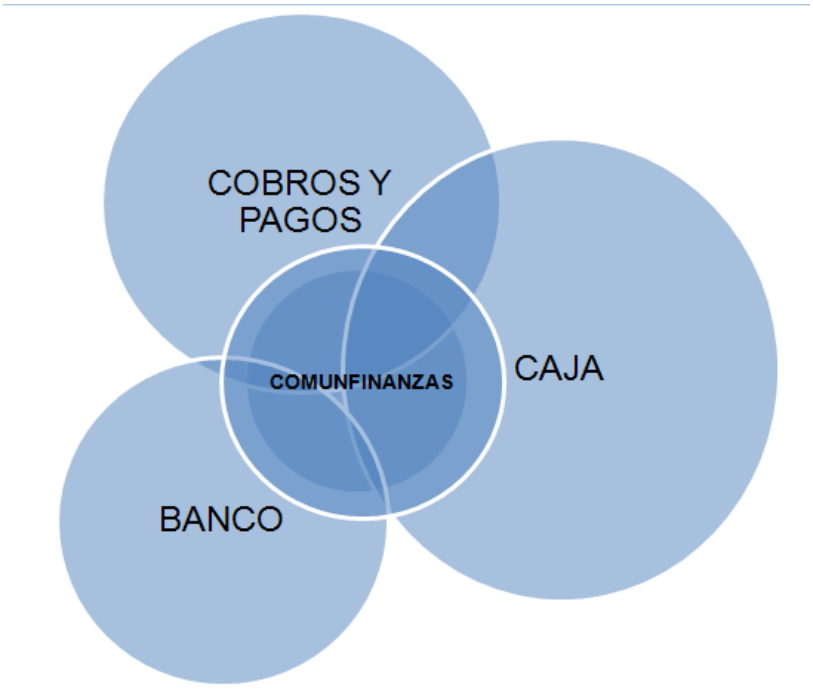


Figura 2.21. Estructura de finanzas

¿Cómo funciona el sistema?

El usuario accede al sistema autenticándose, selecciona una entidad para trabajar y en dependencia del rol, se muestran las funcionalidades a las cuales puede acceder. Dentro del subsistema finanzas se

encuentra el módulo de caja donde aparecen los componentes del mismo, los cuales se describen a continuación.

2.5.1 Componente Configuración.

El componente Configuración tiene varias funcionalidades: **Tipo de Fondo**, **Gestionar Fondos**, **Gestionar Caja** y **Gestionar Tipo de Movimiento**. Este componente se encarga de nomenclar los elementos fundamentales para el funcionamiento del módulo.

Gestionar Tipos de Fondos

Cuando el usuario selecciona el menú *Gestionar Tipo de Fondo,* se muestran en la interfaz los tipos de fondos existentes *(ver anexo 2).* En este momento el usuario tiene la posibilidad de:

- Adicionar un nuevo tipo fondo.

Al seleccionar el botón adicionar, se le despliega una ventana con los campos necesarios a llenar para la creación de un nuevo tipo de fondo.

La implementación de un nuevo tipo de fondo se logra a través del método *insertarTipoFondoAction().* En este método de la clase controladora *GesttipofondoController,* se crea un objeto de la clase entidad *NomTipoFondo*, luego este se envía a la clase modelo *NomTipoFondoModel* para guardar el tipo de fondo en la base de datos.

- Modificar tipo fondo existente.

Es necesario seleccionar un tipo de fondo existente para que se active el botón modificar, al ser seleccionado el mismo se despliega un ventana con las características de dicho tipo de fondo, permitiendo al usuario realizar los cambios que estime necesarios.

La implementación para modificar un fondo, se logra a través del método *modificarFondoAction()* existente en la clase controladora. En este método se obtiene un objeto *NomTipoFondo* existente en la base de datos, luego este se envía a la clase modelo para actualizar los datos del mismo.

- Eliminar un tipo fondo existente.

Es necesario seleccionar un tipo fondo existente para que se active el botón eliminar, y así poder realizar la operación.

La implementación para eliminar un tipo-fondo, se logra a través del método *eliminarFondoAction()* de la clase controladora. En este método se obtiene un objeto *NomTipoFondo* existente en la base de datos, luego este se envía a la clase modelo para eliminar los datos del mismo.

- Buscar un tipo fondo existente.

Existen 2 parámetros para la búsqueda de un tipo-fondo, el usuario puede buscar por código y/o nombre. Luego de realizar la búsqueda se le muestra al usuario el resultado de todas las coincidencias encontradas.

La implementación para buscar un fondo, se logra a través del método *cargarFondoAction()*. En este método se llama al método *buscarTipoFondo(arreglo condiciones)* de la clase modelo para realizar la búsqueda del tipo de fondo con los parámetros previamente entrados, obteniéndose un arreglo de objetos *NomTipoFondo*, los cuales se muestran finalmente al usuario como resultado de la búsqueda.

Gestionar Fondos

Cuando el usuario selecciona el menú *Gestionar Fondo,* se muestran en la interfaz los fondos existentes*(ver anexo 3)*. En este momento el usuario tiene la posibilidad de:

- Adicionar un nuevo fondo.

Al seleccionar el botón adicionar, se le despliega una ventana intuitiva con los campos necesarios a llenar para la creación de un nuevo fondo.

La implementación de un nuevo fondo se logra a través del método *insertarFondoAction()*. En este método de la clase controladora *GestfondoController* se crea un objeto de la clase entidad *NomFondo*, luego este se envía a la clase modelo *NomFondoModel* para guardar los datos del fondo en la base de datos.

- Modificar fondo existente.

Es necesario seleccionar un fondo existente para que se active el botón modificar, al ser seleccionado el mismo se despliega un ventana con las características de dicho fondo, permitiendo al usuario realizar los cambios que estime necesarios, siempre y cuando el fondo no esté asociado a una caja.

La implementación para modificar un fondo, se logra a través del método *modificarFondoAction()* de la clase controladora. En este método se obtiene un objeto *NomFondo* existente en la base de datos, luego este se envía a la clase modelo para actualizar los datos del mismo, comprobándose previamente que el fondo no esté asociado a una caja.

- Eliminar un fondo existente.

Es necesario seleccionar un fondo existente para que se active el botón eliminar, solo se puede realizar la operación si el fondo no está asociado a una caja.

La implementación para eliminar un fondo, se logra a través del método *eliminarFondoAction().*En este método de la clase controladora se obtiene un objeto de *NomFondo* existente en la base de datos, luego este se envía a la clase modelo para eliminar los datos del mismo, comprobándose previamente que el fondo no esté asociado a una caja.

- Buscar un fondo existente.

Existen 3 parámetros para la búsqueda de fondos, el usuario puede buscar por código, nombre, y/o tipo de fondo. Luego de realizar la búsqueda se le muestra al usuario el resultado de todas las coincidencias encontradas.

La implementación para buscar un fondo, se logra a través del método *cargarFondoAction()*.En este método se realiza una búsqueda a partir de los parámetros previamente entrados, filtrándose a través del método de la clase modelo *buscarFondo(arreglo condiciones)*, obteniéndose un arreglo de objetos *NomFondo*, los cuales se muestran finalmente al usuario como resultado de la búsqueda.

Gestionar Caja

Cuando el usuario selecciona el menú *Gestionar Caja*, se muestran en la interfaz las cajas existentes *(ver anexo 4)*. En este momento el usuario tiene la posibilidad de:

- Adicionar nueva caja.

Al seleccionar el botón adicionar, se le despliega una ventana la cual contiene los atributos necesarios para la creación de una nueva caja. El usuario llena los campos de acuerdo a sus necesidades creando la caja deseada.

La implementación de una nueva caja se logra a través del método *insertarCajaAction()* de la clase controladora *GestcajaController.* En este método se crea un objeto de la clase entidad *NomCaja* a partir de los campos llenados por el usuario, luego este se envía a la clase modelo *NomCajaModel* para persistir los datos de la caja.

- Modificar caja existente.

Es necesario seleccionar una caja existente para que se active el botón modificar, al ser seleccionado el mismo se despliega un ventana con las características de dicha caja, permitiendo al usuario realizar los cambios que estime necesarios, siempre y cuando la caja no tenga asociado ningún fondo.

La implementación para modificar una caja, se logra a través del método *modificarCajaAction()* de la clase controladora. En este método se obtiene un objeto *NomCaja* existente en la base de datos, luego este se envía a la clase modelo para actualizar los datos del mismo; previamente se debe haber verificado que la caja no estuviera asociada a ningún fondo.

- Eliminar caja existente.

Es necesario seleccionar una caja existente para que se active el botón eliminar, solo se puede realizar la operación si la caja no está asociado a un fondo.

La implementación para eliminar una caja, se logra a través del método de la clase controladora *eliminarCajaAction()*.En este método se obtiene un objeto *NomCaja* existente en la base de datos, luego se comprueba que la caja no esté asociada a un fondo, para finalmente enviar el objeto *NomCaja* a la clase modelo y eliminarlo.

- Buscar caja existente.

Existen 2 parámetros para la búsqueda de cajas, el usuario puede buscar por código y/o caja. Luego de realizar la búsqueda se le muestra al usuario el resultado de todas las coincidencias encontradas.

La implementación para buscar caja, se logra a través del método *cargarCajaAction().* Este método de la clase controladora realiza una búsqueda a partir de los parámetros previamente entrados, a través del método de la clase modelo *buscarCaja(arreglo condiciones),* obteniéndose un arreglo de objetos *NomCaja*, los cuales se muestran finalmente al usuario como resultado de la búsqueda.

- Asociar / Desasociar fondos a determinada caja.

Es necesario seleccionar una caja existente para que se active el botón Asociar/Desasociar Fondos. Al ser seleccionado el mismo se despliega una ventana con los fondos asociados a dicha caja, además de los fondos sin asociar; permitiendo al usuario asociar y/o desasociar los fondos seleccionados a la caja. Sólo se permitirá al usuario desasociar fondo cuando a este no se le haya realizado una apertura.

La implementación para Asociar / Desasociar fondos a una caja, se logra a través del método *asociarDesasociarAction().* En este método se obtiene un objeto *NomCaja* existente en la base de datos. Para asociar se obtiene además, un objeto *NomFondo*, los mismos se envían a la clase modelo *DatCajafondoModel*, en la cual se persiste la asociación de un fondo a una caja. Para Desasociar solamente se verifica que al fondo asociado a la caja no se le haya realizado ninguna apertura.

- Realizar Apertura a un fondo.

Es necesario seleccionar una caja existente para que se active el botón Apertura Fondo, al ser seleccionado el mismo se despliega una ventana con los fondos asociados a dicha caja. Se selecciona el fondo al cual se le va a realizar la apertura y se muestra una nueva ventana que contiene varios campos, los cuales serán llenados por el usuario y que permitirá realizar la apertura.

La implementación para Realizar Apertura a un fondo, se logra a través del método *crearAperturaFondoAction().* En este método se crea un objeto de la clase entidad *NomAperturaFondo* a partir de los campos llenados por el usuario y los instrumentos con los que se va a crear. Dichos instrumentos se registran en el componente **comunfinanzas**, mediante la integración por el IoC, en el cual se guardan un documento por cada instrumento que se registra para la apertura del fondo.

Gestionar Tipo Movimiento

Cuando el usuario selecciona el menú *Gestionar Fondos*, se muestran en la interfaz los tipos de movimientos existentes *(ver anexo 5)*. En este momento el usuario tiene la posibilidad de:

- Adicionar nuevo tipo de movimiento.

Al seleccionar el botón adicionar, se le despliega una ventana la cual contiene los atributos necesarios para la creación de un nuevo tipo de movimiento. El usuario llena los campos de acuerdo a sus necesidades creando un nuevo tipo de movimiento.

La implementación de un nuevo tipo de movimiento se logra a través del método *insertarTipoMovAction()* de la clase controladora *GesttipomovimientoController.* En este método se crea un objeto de la clase entidad *NomTipomovimientocaja* a partir de los campos llenados por el usuario, luego este se envía a la clase modelo *NomTipomovimientocajaModel* para guardar los datos.

- Modificar tipo de movimiento existente.

Es necesario seleccionar un tipo de movimiento existente para que se active el botón modificar, al ser seleccionado el mismo se despliega un ventana con las características de dicho tipo de movimiento, permitiendo al usuario realizar los cambios que estime necesarios, siempre y cuando el tipo de movimiento no esté asociado a ningún fondo.

La implementación para modificar un tipo de movimiento, se logra a través del método *modificarTipoMovAction()* de la clase controladora. En este método se obtiene un objeto *NomTipomovimientocaja* existente en la base de datos, luego este se envía a la clase *NomTipomovimientocajaModel* para actualizar los datos del mismo; previamente se verifica si está asociado a algún fondo, caso en el que solo se podrán modificar algunos atributos.

- Eliminar tipo de movimiento existente.

Es necesario seleccionar un tipo de movimiento existente para que se active el botón eliminar, solo se puede realizar la operación si el tipo de movimiento no está asociado a un fondo.

La implementación para eliminar un tipo de movimiento, se logra a través del método de la clase controladora *eliminarTipoMovAction()*. En este método se obtiene un objeto *NomTipomovimientocaja* existente en la base de datos, luego se comprueba que la caja no esté asociada a un fondo, para finalmente enviar el objeto *NomTipomovimientocaja* a la clase modelo *NomTipomovimientocajaModel* y eliminarlo de la base de datos.

2.5.2 Componente Movimiento.

El componente Movimiento tiene varias funcionales: **Anticipos**, **Depósito**, **Liquidación de Derechos de Cobros**, **Reembolso** y **Movimientos Genéricos**. Este componente posee la mayor cantidad de funcionalidades dentro del módulo Caja, y en el mismo se da solución a la mayoría de los procesos de Caja de forma óptima.

Anticipos.

Cuando el usuario selecciona el menú *Anticipos* se muestran varias opciones de Anticipo Nacional: Registrar nuevo, Entregar y Liquidar; además de poder Registrar y Liquidar Anticipo Extranjero. En este momento el usuario tiene la posibilidad de:

- Registrar nuevo anticipo nacional.

Al ser seleccionado por el encargado de control de anticipos se muestran en la interfaz varios campos a llenar para confeccionar el anticipo *(ver anexo 6)*. Después de llenos los campos, se crea el anticipo en el estado de pendiente a entregar.

La implementación para registrar anticipo nacional se logra a través del método *registrarAnticipoNacionalAction()* de la clase controladora *CrearanticiponacionalController.* En este método se crea un objeto *DatAnticipoviajenacional* que hereda de *DatAnticipo*, es decir que lo contiene, este se manda a *DatAnticipoviajenacionalModel* para guardar el anticipo, además se salva un objeto *Nomestadoanticipo* en el estado de pendiente a entregar.

- Entregar Anticipo Nacional

Para entregar un Anticipo Nacional se muestran al cajero todos los anticipos nacionales pendientes a entregar *(ver anexo 7)*. El cajero selecciona el fondo al que se le va a realizar el movimiento y solicita al trabajador que va de viaje sus datos personales y le entrega la cantidad de dinero que se le asignó para el viaje, quedando el Anticipo en el estado de pendiente a liquidar.

La implementación para entregar anticipo nacional se logra mediante el método *crearEntregarAnticipoAction()* de la clase *EntregaranticiponacionalController*. En este método se crea un objeto *DatAnticipoviajenacionalentregado*, el cual se salva mediante la clase modelo

DatAnticipoviajenacionalModel. En este momento se modifica el objeto *Nomestadoanticipo* que pasa al estado de pendiente a liquidar.

- Liquidar Anticipo Nacional.

Al seleccionar el cajero el menú *Liquidar Anticipo Nacional* se muestra un listado de los Anticipos Nacionales en el estado de pendientes a liquidar de la entidad *(ver anexo 8)*. A su vez el cajero puede realizar una búsqueda avanzada por el nombre del trabajador, clasificación y/o fecha del anticipo. Luego se registra la liquidación y se genera un Documento.

La implementación de esta liquidación se logra en el método *liquidacionAnticipoNacionalAction()* de la clase controladora *LiquidaranticiponacionalController.* Este método crea un objeto *DatLiquidacionviajenacional* para la liquidación del Anticipo de Viaje Extranjero que se guarda en la clase modelo *DatLiquidacionviajenacionalModel*. Se genera un documento.

- Registrar Anticipo Extranjero.

Al ser seleccionado por el encargado de control de anticipos esta opción, se muestran en la interfaz varios campos a llenar para confeccionar el anticipo de viaje extranjero *(ver anexo 9)*. Después de llenos los campos se crea el anticipo extranjero en el estado de pendiente a entregar.

La implementación para registrar anticipo extranjero se logra en la clase controladora *CrearanticipoextranjeroController* mediante el método *registrarAnticipoExtranjeroAction().* En este método se crea un objeto *DatAnticipoviajeextranjero* que hereda de *DatAnticipo*, es decir que lo contiene, además se crean tantos objetos *DatAnticipoviajeextranjeropais* como países vaya a visitar el trabajador, los mismos se guardan en *DatAnticipoviajenacionalModel,* finalmente el objeto *DatAnticipoviajeextranjero* se manda a *DatAnticipoviajeextranjeroModel* para guardar el anticipo extranjero, además se salva un objeto *Nomestadoanticipo* en el estado de pendiente a entregar.

- Liquidar Anticipo Extranjero.

Al seleccionar el cajero el menú *Liquidar Anticipo Extranjero* se le muestra el listado de los Anticipos Extranjeros en el estado de pendientes a liquidar. A su vez el cajero puede realizar una búsqueda avanzada por el nombre del trabajador y/o fecha del anticipo. Luego se registra la liquidación y se genera un Documento del tipo anticipo y liquidación de gastos de viajes-divisas.

La implementación de esta liquidación se logra en el método *liquidacionAnticipoExtranjeroAction()* de la clase controladora *LiquidaranticipoextranjeroController.* Este método crea un objeto *DatLiquidacionviajeextranjeropais* por cada país que haya visitado el trabajador, y un objeto *DatLiquidacionviajeextranjero* para la liquidación del Anticipo de Viaje Extranjero que se guarda en la clase modelo *DatLiquidacionviajeextranjeroModel*. En caso de no haber consumido el anticipo totalmente se actualiza el importe en caja y se genera el documento.

Depósito.

Cuando el usuario selecciona el menú *Realizar Depósito*, se muestran una ventana para que seleccione el fondo al que realizarle un depósito *(ver anexo 10)*. Luego se le muestra al cajero en la interfaz para llene el modelo de depósito.

Luego de seleccionado el fondo se le muestra al usuario en la interfaz el modelo de depósito, el mismo posee un desglose de dinero en monedas y billetes. El cajero llena los datos y realiza un depósito.

La implementación para realizar un depósito se logra en el método *realizardepositoAction ()*, de la clase controladora *RealizardepositoController.* En este método se crea un objeto *DatDeposito* a partir de los datos entrados por el cajero y los instrumentos seleccionados, los mismos después de realizar el depósito pasan al estado de depositado. Esta operación genera un documento.

Liquidación de Derechos de Cobros.

Cuando el usuario selecciona el menú *Liquidar Derechos de Cobro* se muestran una ventana para seleccionar el fondo. Luego se muestra una interfaz para llenar los datos del instrumento con el cual se va a liquidar los derechos de cobros.

- Liquidación de Derechos

Se muestra la ventana para seleccionar el fondo. Luego se muestra una interfaz para llenar los datos del instrumento con el cual se va a liquidar los derechos de cobros de un cliente. Luego se muestra al cajero una serie de operaciones pendientes a liquidar del cliente en cuestión. Finalmente se liquidan las operaciones seleccionadas con el instrumento. Este movimiento genera un documento.

La implementación para realizar una liquidación de Derecho de Cobro se logra en la clase controladora *LiquidarderechocobroController* en el método *liquidacionDerechoCobro().* En este método se guarda el

objeto instrumento en el **comunfinanzas;** luego se seleccionan las operaciones que se van a liquidar a partir del instrumento. Finalmente se liquidan las operaciones y se emite un documento.

Realizar Reembolso.

Al ser seleccionado el menú *Realizar Reembolso* por el cajero se muestra una ventana que carga los fondos de reembolso de una caja *(ver anexo 11)*. Luego se muestran los documentos del fondo que pueden ser reembolsados. El cajero selecciona los documentos que desea reembolsar y realiza el reembolso.

La implementación de Reembolso se realiza mediante el método *realizarReembolso()* de la clase controladora *ReembolsoController*. En este método se crea un objeto *DatReembolso* que contiene una lista de instrumentos a reembolsar. Se genera un documento.

Movimiento Genérico.

Al ser seleccionado el menú *Realizar Movimiento Genérico* por el cajero se muestra una ventana para seleccionar el fondo al que se le va a realizar el movimiento *(ver anexo 12)*. Luego se le muestran los campos para llenar el movimiento genérico y guardar sus datos.

Después de seleccionado el fondo se le muestra al usuario un modelo general para realizar otros movimientos o movimientos genéricos. El cajero finalmente llena el movimiento especificando si es de entrada o salida.

La implementación de los Movimientos Genéricos se logra a través del método *registrarMovimientoGenerico()* de la clase controladora *MovimientosgenericosController.* En este método se crea un objeto *DatMovimientogenerico* que tiene asociado varios instrumentos. Estos instrumentos se guardan en el comunfinanzas y el objeto *DatMovimientogenerico* se envía a la clase modelo *DatMovimientogenericoModel* donde se persiste en la base de datos.

2.5.3 Componente Arqueo.

Cuando el usuario selecciona el menú *Realizar Arqueo,* se muestran en la interfaz las cajas existentes *(ver anexo 13)*. En este momento el usuario tiene la posibilidad de:

- Realizar Arqueo a los fondos asociados a una caja

Es necesario seleccionar una caja existente para que se active el botón Realizar Arqueo, al ser seleccionado el mismo se despliega un ventana con las características necesarias para realizar un arqueo, tales como seleccionar todos los fondos a los cuales se desea realizar un arqueo; junto con sus denominaciones de monedas. Luego se realizan operaciones tales como efectivo por denominación de un fondo, calcular el total de efectivo, listar los documentos justificantes de un fondo, calcular el importe total de documentos de valor del fondo, calcular el importe total de documentos justificantes seleccionados, determinar la diferencia con el fondo autorizado, generar el documento de arqueo, generar la vista previa del documento de arqueo, imprimir el documento de arqueo y Generar un comprobante de operaciones.

Al cierre de cada mes o por sorpresa, cuantas veces se considere necesario, por el Director de Administración o persona en quien éste delegue. También deben realizarse arqueos al producirse una sustitución del cajero o por ausencias temporales conocidas del mismo. Para realizar este tipo de actividad es necesario que el cajero o custodio esté presente.

Todas las funcionalidades antes expuestas en su conjunto posibilitan que el usuario de manera intuitiva pueda realizar las operaciones con el mayor grado de simplicidad posible.

La implementación para realizar un fondo, se logra a través de métodos tales como cargarFondosAsociadosAUnaCajaAction(enteropos id). En este método se le pasa por parámetros el identificador de una caja y a través de llamadas al IoC busca en el componente Configuración, específicamente en la capa Modelo el método que carga los fondos asociados a la caja seleccionada, de esta manera se le muestran al usuario los fondos para que seleccione a los que desea realizar el arqueo.

2.5.4 Componente Comunfinanzas

El componente **Comunfinanzas** se creó con de propósito de reutilizar las funcionalidades comunes para los módulos de **Caja**, **Banco** y **Cobros Pagos**. El mismo es el encargado de generar los documentos e instrumentos en la medida que se crean las operaciones en las entidades. También **Comunfinanzas** provee talonarios para los módulos **Caja** y **Banco** de acuerdo a las especificidades de cada uno.

2.6 Algoritmos Complejos.

Un algoritmo complejo es aquel donde se resuelve un problema a través de arboles de decisiones. Algunos algoritmos complejos son usados como artefactos de implementación en el módulo Caja durante

su elaboración. Dado el grado de dificultad del módulo de Caja surgió la necesidad de usar el tipo de dato abstracto Árbol para alcanzar una solución eficiente.

2.6.1 Algoritmos Complejos del Módulo.

Unos de algoritmos complejos usados en la solución del módulo caja es Talonario, el mismo está representado por una estructura arbólica que brinda servicios no solo al módulo estudiado, sino también al módulo Banco. En la estructura adoptada se encuentra dentro del componente **Comunfinanzas.**

Las empresas tradicionalmente a cada documento le asignan un número de talonario; el mismo se debe haber fijado a un fondo que tiene implícito un bloque de talonarios. Al crear los talonarios las empresas definen los talonarios que serán empleados durante un periodo determinado. Luego este talonario inicial o padre se divide en otros más pequeños asignándoselo a un fondo de una caja, este proceso se puede repetir recursivamente tantas veces lo estime necesario la empresa.

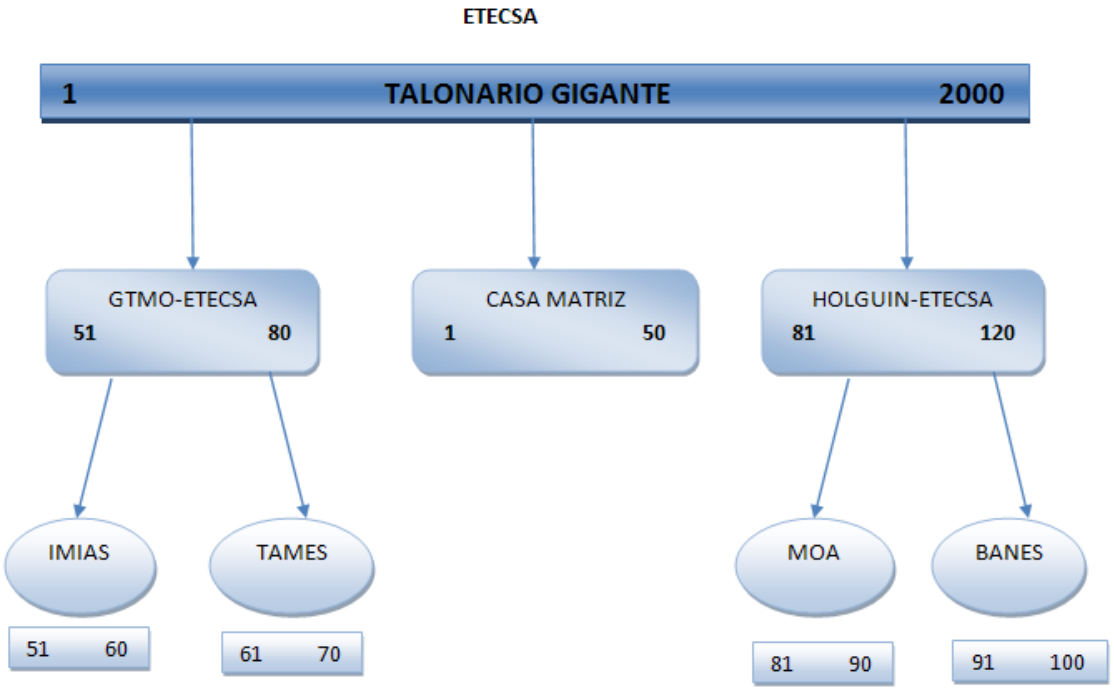


Figura 2.22. Árbol de talonario.

2.6.2 Análisis de Complejidad de Algoritmos

La Complejidad Ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. Es una de las métricas de software mas ampliamente aceptada, ya que ha sido concebida para ser independiente del lenguaje.

Esta métrica, se basa en el diagrama de flujo determinado por las estructuras de control de un determinado código. De dicho análisis se puede obtener una medida cuantitativa de la dificultad de de crear pruebas automáticas del código y también es una medición orientativa de la fiabilidad del mismo.

El resultado obtenido en el cálculo de la complejidad ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

La medida resultante puede ser utilizada en el desarrollo, mantenimiento y reingeniería para estimar el riesgo, costo y estabilidad. Algunos estudios experimentales indican la existencia de distintas relaciones entre la métrica de Thomas McCabe y el número de errores existentes en el código fuente, así como el tiempo requerido para encontrar y corregir esos errores.

Una vez calculada la complejidad ciclomática de un fragmento de código, se puede determinar el riesgo que supone utilizando los rangos definidos en la siguiente tabla:

Complejidad Ciclomática	Evaluación del Riesgo
1-10	Programa Simple, sin mucho riesgo
11-20	Más complejo, riesgo moderado
21-50	Complejo, Programa de alto riesgo
50	Programa no testeable, Muy alto riesgo

A partir del análisis de muchos proyectos McCabe encontró que un valor 10 es un límite superior práctico para el tamaño de un módulo. Cuando la complejidad supera dicho valor se hace muy difícil probarlo, entenderlo y modificarlo. La limitación deliberada de la complejidad en todas las fases del desarrollo ayuda a evitar los problemas asociados a proyectos de alta complejidad. El límite propuesto por McCabe sin embargo es fuente de controversias. Algunas organizaciones han utilizado el valor 15 con bastante éxito.

2.6.2.1 Cálculo de la Complejidad Ciclomática

Para poder dar una definición acabada de la complejidad ciclomática, es necesario primero introducir una sencilla notación para la representación del flujo de control, denominada Grafos de Flujo de Control de un programa.

M = Complejidad ciclomática.

E = Número de aristas del grafo. Una arista conecta dos vértices si una sentencia puede ser ejecutada inmediatamente después de la primera.

N = Número de nodos del grafo correspondientes a sentencias del programa.

P = Número de componentes conexos correspondientes a las diferentes subrutinas, funciones o métodos. (16)

Definidos estos conceptos, la Complejidad Ciclomática puede calcularse de la siguiente manera:

$$M = E - N + P$$

Una versión simplificada para el cálculo de la Complejidad Ciclomática es la siguiente:

M = Número de condiciones + 1

Más adelante en el **CAPÍTULO III** se realizará el cálculo a un algoritmo implementado en la solución del módulo.

2.7 Descripción de clases y operaciones del módulo.

2.7.1 Principales clases del componente Configuración.

2.7.1.1 Clases del Modelo

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.

2.7.1.2 Clases entidad.

Clase Entidad 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.

Clase Entidad 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.

existeTipoInstrumento($idfondo,$idinst)	Encargado de verificar si existe un tipo de intrumento.
buscar(arreglo $condiciones)	Encargado buscar todos los tipos de fondos

Clase Entidad 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
dameFondoAsociados($idcaja)	Encargado de devolver los fondos asociados a una caja.
dameFondoNoAsociados()	Encargado de devolver los fondos no asociados a un caja
movimientosFondo ($idfondo)	Encargado de devolver todos los movimientos realizados a un fondo.

Clase Entidad 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

Clase Entidad Nomenclador de tipos de fondos tipos 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.

Clase Entidad Apertura de Fondos

Nombre: DatAperturafondo	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**

Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
existeApertura($idfondo)	Encargado de verifica si el fondo tiene apertura realizadas.

Clase Entidad 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.

2.7.2 Principales clases del componente Movimientos.

2.7.2.1 Clase del modelo

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.

2.7.2.2 Clases entidad

Clase Entidad 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

Clase Entidad Anticipo de Viaje Nacional.

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.

Clase Entidad liquidación viaje nacional.

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.

Clase Entidad Liquidación de Viaje al 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.

Clase Entidad Detalles de Movimientos.

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.

Clase Entidad 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.

Clase Entidad 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.

2.7.3 Principales clases del componente Arqueos.

2.7.3.1 Clases del Modelo

Clase Modelo Arqueos 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.

2.7.3.2 Clases entidad

Clase Entidad 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

Clase Entidad 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.

2.7.4 Principales clases del componente comunfinanzas.

2.7.4.1 Clases del Modelo

Clase modelo Talonarios de comunfinanzas

Nombre: NomTalonariofnzModel	
Tipo de clase: *Modelo***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
NonTalonariofnzModel ()	Constructor de la clase.
insertarTalonario(& $objtalonario)	Encargado de insertar los talonarios
darNumeroTalonario($idtal)	Encargado de dar un numero de talonario, a partir de un talonario existente.
talonariosNoAsociados($identi,$idpadr,$idsub, $cajabanco, $elem,$arregloTD)	Encargado de devolver todos los talonarios no asociados.
asociarTalonario($arreglo)	Encargado de asociar los talonarios

El resto de las clases modelos del componente comunfinanzas son genéricas, por lo que quedarían de la siguiente forma:

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.
NombreClaseModel ()	Constructor de la clase.

2.7.4.2 Clases Entidad

Clase Entidad Talonarios de Cajas.

Nombre: NomTalonariocajafnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($idtalonario)	Encargado de obtener un objeto talonario caja.

Clase Entidad Talonarios de Banco.

Nombre: NomTalonariobancofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
Obtener($indtalonario)	Encargado de obtener un objeto talonario banco.

Clase Entidad talonarios de comunfinanzas.

Nombre: NomTalonariofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtenerArbolTalonarios($idestructura, $idpadre,$subsistema)	Encargado de obtener un árbol de talonarios.
obtenerNumeroTalonario($idtalonario)	Encargado de obtener un número de talonarios.
obtenerTalonariosAsignados($cajabanco)	Encargado de obtener los talonarios asignados a caja o banco
talonariosAsociar($idestructura, $idpadre, $caja, $idsubsistema, $elemAsoc, $idtipodoc)	Encargado de devolver los talonarios que se van a asociar.

Clase Entidad documento de comunfinanzas.

Nombre: DatDocumentofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($iddoc)	Encargado de obtener un objeto documento por id.

cargarTodoDoc()	Encargado de obtener todos los documentos
cargarDocPorEstado($idestado)	Encargado cargar los documentos por estado

Clase Entidad Contratos de comunfinanzas

Nombre: DatContratofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($idcontrato)	Encargado de obtener un objeto contrato por id.

Clase Entidad Modelo de comunfinanzas.

Nombre: DatModelofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($idmodelo)	Encargado de obtener un objeto modelo por id.

Clase Entidad Instrumentos de comunfinanzas.

Nombre: DatInstrumentofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($idinstrumento)	Encargado de obtener un objeto instrumento por id.
cargarTodoInstrumento()	Encargado de obtener todos los instrumentos.

Clase Entidad Letras de comunfinanzas.

Nombre: DatLetrafnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**
Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($idletra)	Encargado de obtener un objeto letra por id.
cargarTodaLetra ()	Encargado de obtener todas los letras.

Clase Entidad Anticipos de comunfinanzas

Nombre: DatAnticipofnz	
Tipo de clase: *Entidad***.**	
Atributo	**Tipo**

Para cada responsabilidad	
Nombre.	**Descripción.**
setUp ()	Constructor de la clase.
obtener($idanticipo)	Encargado de obtener todos los anticipos por id.
cargarTodoAnticipo ()	Encargado de obtener todos los anticipos.
cargarAnticipoPorIdDoc(iddoc)	Encargado de cargar los anticipos por id de documento

2.8 CONCLUSIONES DEL CAPÍTULO II

En este capítulo se concluyó que la obtención del diseño propuesto por el analista fue muy importante desde el punto de vista que ayudo en gran medida a la correcta conformación de los componentes a implementar, así como todo lo relacionado con los requisitos no funcionales.

La arquitectura escogida fue la más apropiada, logrando satisfacer las funcionalidades y requerimientos del sistema. Esta estuvo caracterizada por la creación de un nuevo módulo que brinda funcionalidades comunes al resto de los módulos del subsistema.

Para finalizar este capítulo se realizó un análisis de los estándares de codificación, algoritmos complejos implementados y la importancia de estos para el correcto funcionamiento de la solución.

CAPITULO III: VALIDACION DE LA SOLUCION PROPUESTA.

3.1 Introducción

En el desarrollo de este capitulo se hace un análisis de los temas tratados en el transcurso del Trabajo de Diploma y se valida mediante pruebas el software realizado. Las pruebas de software son un elemento crítico para la garantía de la calidad de software y representan una revisión final de las especificaciones del diseño y de la codificación. También se hace una valoración de la solución y por último se realizan los distintos tipos de prueba tales como caja blanca y caja negra, la primera de ella se encarga de descubrir errores de codificación y rendimiento, y la segunda de la fortaleza de las interfaces.

3.2 Valoración de la solución

Luego de la implementación de la solución se hizo un análisis donde se evidenció que se logró cumplir con las especificidades propuestas por los analistas, ya que el sistema contribuirá a tener un mayor control de las operaciones realizadas en las cajas, erradicando los documentos repetidos, facilitando el trabajo y evitando posibles errores en el proceso.

Se logrará con este sistema una mayor integración en el subsistema finanzas, pues todos sus módulos podrán usar el componente **comunfinanzas** con el mismo fin, adaptándolo en cada caso a las especificidades del mismo.

Se utilizaron varios patrones de arquitectura y diseño, tales como el MVC y el patrón Proxy respectivamente. Conllevando a obtener un sistema robusto, en una solución simple, limpia, elegante y con mejores oportunidades de reutilización.

3.3 Pruebas de software.

Las pruebas de software son elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

La creciente percepción del software como un elemento del sistema y la importancia de los costos asociados a un fallo del propio sistema, están motivando la creación de pruebas minuciosas y bien planificadas. No es raro que una organización de desarrollo de software emplee entre el 30 y 40 por ciento del esfuerzo total del proyecto en las pruebas.

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. (17)

3.3.1 Objetivos

El objetivo de las pruebas, expresado de forma sencilla, es encontrar el mayor número posible de errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista. Se debe ejecutar el programa antes de que llegue al cliente, con la intención especifica de descubrir todos los errores, de manera que el cliente no experimente la frustración asociada con un producto de baja calidad. Con el propósito de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente.

Otro de sus objetivos son que:

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.
- Reducir costos de mantenimiento.
- Obtener información concreta acerca de fallas, que pueda usarse como apoyo en la mejora de procesos, y en la de los desarrolladores.

Los objetivos anteriores suponen un cambio dramático de punto de vista. Nos quitan la idea que, normalmente, tenemos de que una prueba tiene éxito si no descubre errores. Nuestro objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y de esfuerzo. Si la prueba se lleva a cabo con éxito (de acuerdo con el objetivo anteriormente establecido), descubrirá errores en el software. Como ventaja secundaria, la prueba demuestra hasta que punto las funciones del software parecen funcionar de acuerdo con las especificaciones y parecen alcanzarse los requisitos de rendimiento. (18)

3.3.2 Alcance.

La prueba de software tiene limitantes, tanto teóricos como prácticos. Desde el punto de vista teórico, la prueba es un problema que llamamos no-decidible; esto implica, de algún modo, que no podemos escribir un programa que pruebe los programas sin intervención humana. Sin embargo, como mencionábamos anteriormente, la prueba sí es automatizable en muchos aspectos. Desde el punto de vista práctico, la cantidad de posibilidades para probar exhaustivamente un sistema es sencillamente inmanejable; es necesario entonces utilizar técnicas adecuadas para maximizar la cantidad de fallas importantes encontradas con los recursos asignados. Cada método que se utilice para detectar defectos deja un residuo de defectos más sutiles contra los cuales ese método es ineficaz (la llamada “Paradoja del Pesticida”).

La prueba de software implica pues, la aplicación de técnicas y herramientas apropiadas en el marco de un proceso bien definido, determinado por el tipo de proyectos de desarrollo de software que se abordan.

3.4 Descripción de las pruebas de caja blanca y caja negra.

Los test de unidad validan pequeñas subrutinas y funciones. Se suelen delegar en los programadores individuales, mediante la utilización de herramientas de apoyo a las actividades de test, dejando de lado herramientas más sofisticadas para el desarrollo de pruebas basadas en criterios formales. Un test de unidad es una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

Para que un test de unidad sea efectivo debe cumplir los siguientes requisitos:

- **Automatizable**: no debería requerirse una intervención manual. Esto es especialmente útil para integración continua.
- **Completas**: deben cubrir la mayor cantidad de código.
- **Repetibles o Reutilizables**: no se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para integración continua.
- **Independientes**: la ejecución de una prueba no debe afectar a la ejecución de otra.

- **Profesionales**: las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc. (19)

Otra de las características de los test de unidades es que son orientados en su mayoría a las pruebas de “caja blanca” aunque para realizar uno de estos test es necesario probar el flujo de datos desde la interfaz del componente. Si los datos no se comportan correctamente al ser introducidos en la aplicación, todas las demás pruebas carecen de fiabilidad.

Por último los test de unidades se dividen en 2 tipos:

- **Técnicas estructurales o de caja blanca**: Son orientadas a la complejidad subyacente del software, teniéndose conocimiento de la implementación y el testeo de funciones.
- **Técnicas funcionales o de caja negra**: Son orientadas a las entradas o salidas de un componente, desconocen de la implementación y son útiles para testear el comportamiento observable e interfaces.

3.4.2 Prueba de Caja Blanca.

La prueba de caja blanca, denominada a veces prueba de caja de cristal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, se puede obtener casos de prueba que:

- Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo.
- Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa.
- Ejecuten todos los bucles en sus límites y con sus límites operacionales.
- Ejerciten las estructuras internas de datos para asegurar su validez.

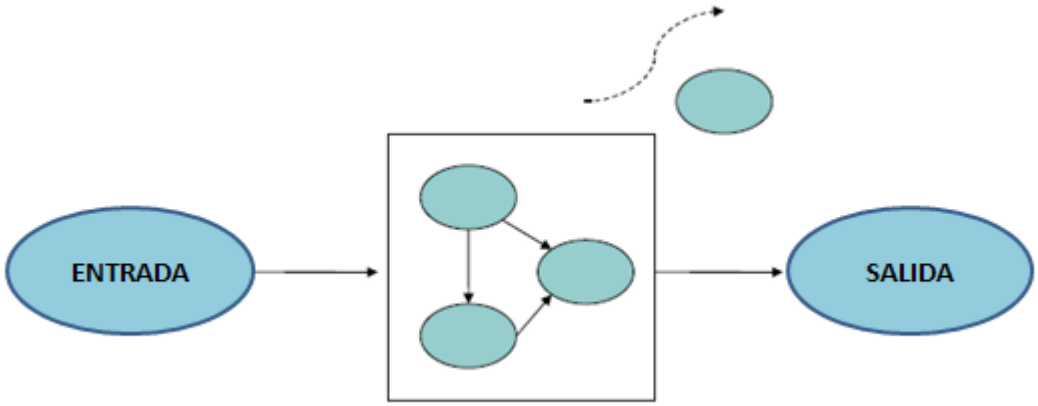


Figura 3.1: prueba de caja blanca

3.4.2.1 Prueba de camino básico.

La prueba del camino básico es una técnica de prueba de caja blanca. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

3.4.2.2 Pruebas de cubrimiento.

Las pruebas de cubrimiento necesitan varios casos de prueba para determinar posibles caminos independientes, además de que cada condición debe cumplirse en un caso y para otro no, en general se necesitan tantos casos como condiciones, más un número ciclomático. (20)

3.4.2.3 Pruebas de condiciones.

Las pruebas de condiciones cumplen o no cada parte de la condición. Se necesitan varios casos de prueba para determinar expresiones simples en las condiciones.

3.4.2.4 Pruebas de bucles.

Las pruebas de bucles se especializan en la construcción de bucles. Necesitan de un número de repeticiones especiales, y se pueden dividir en simples o anidados.

3.4.3 Pruebas de Caja Negra.

Las pruebas de caja negra se centran en lo que se espera de un módulo, es decir, intentan encontrar casos en que el módulo no se atiene a su especificación. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro.

Las pruebas de caja negra están especialmente indicadas en aquellos módulos que van a ser interfaz con el usuario (en sentido general: teclado, pantalla, ficheros, canales de comunicaciones, etc.)

Las pruebas de caja negra se apoyan en la especificación de requisitos del módulo. De hecho, se habla de "cobertura de especificación" para dar una medida del número de requisitos que se han probado. Es fácil obtener coberturas del 100% en módulos internos, aunque puede ser más laborioso en módulos con interfaz al exterior. En cualquier caso, es muy recomendable conseguir una alta cobertura en esta línea. (21)

Figura 3.2: Prueba de caja negra

Algunos métodos empleados en las pruebas de caja negra son:

3.4.3.1 Métodos de prueba basados en grafos

En este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas.

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

3.4.3.2 Partición equivalente

Es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Típicamente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

El objetivo de partición equivalente es reducir el posible conjunto de casos de prueba en uno más pequeño, un conjunto manejable que evalúe bien el software. Se toma un riesgo porque se escoge no probar todo. Así que se necesita tener mucho cuidado al escoger las clases. (22)

3.4.3.3 Análisis de valores límite

Los errores tienden a darse más en los límites del campo de entrada que en el centro. Por ello, se ha desarrollado el análisis de valores límites como técnica de prueba. El análisis de valores límite lleva a una elección de casos de prueba que ejerciten los valores límite.

El análisis de valores límite es una técnica de diseño de casos de prueba que completa a la partición equivalente. En lugar de seleccionar cualquier elemento de una clase de equivalencia, el análisis de valores límite lleva a la elección de casos de prueba en los extremos de la clase. En lugar de centrarse

solamente en las condiciones de entrada, el análisis de valores límite obtiene casos de prueba también para el campo de salida.

Condiciones sublímite. Las condiciones limite normales son las más obvias de descubrir. Estas son definidas en la especificación o son evidentes al momento de utilizar el software. Algunos límites, sin embargo, son internos al software, no son necesariamente aparentes al usuario final pero aún así deben ser probadas por el probador. Estas son conocidas como condiciones sublímites o condiciones límite internas. Una condición sublímite común es la tabla de caracteres ASCII, por ejemplo, si se está evaluando una caja de texto que acepta solamente los caracteres AZ y az, se debe incluir los valores en la partición inválida justo «debajo de» y «encima de» esos caracteres de la tabla ASCII

3.5 Aplicación de pruebas de caja blanca.

Luego de realizar un análisis de las pruebas de caja blanca descritas en el epígrafe 3.3 y los procesos del capítulo 2, epígrafe 2.5.2 “**Análisis de complejidad del algoritmo”** para calcular los valores de la complejidad ciclomática del procedimiento al cual se le va a aplicar la prueba, se enumeran a continuación las sentencias de código del procedimiento.

```
public function asosiarDesasociarAction()
{
    $asociados    = json_decode(stripslashes($this->_request->getPost('asociados'))); //1
    $sinasociados = json_decode(stripslashes($this->_request->getPost('sinasociar'))); //1
    $idcaja     = $this->_request->getPost('idcaja');    //1
    if(count($asociados)>0) //2
    {
        foreach($asociados as $a)    //3
        {
            $objAsoc = new NomCajaModel();  //4
            $objAsoc->Desasociar($a->idfondo);  //4
        }   //5
    }   //6
    if(count($sinasociados)>0)  //7
    {
        foreach($sinasociados as $s)     //8
        {
            $objmodel = new DatCajafondoModel();     //9
            $objCF = new DatCajafondo();     //9
            $objCF->idfondo = $s->idfondo;  //9
            $objCF->idcaja  = $idcaja;  //9
            $objmodel->Insertar($objCF);     //9
        }   //10
    }   //11
}   //12
```

Figura 3.3: Representación del algoritmo asociarDesasociarAction().

Posteriormente se construye el grafo de flujo asociado al código anterior.

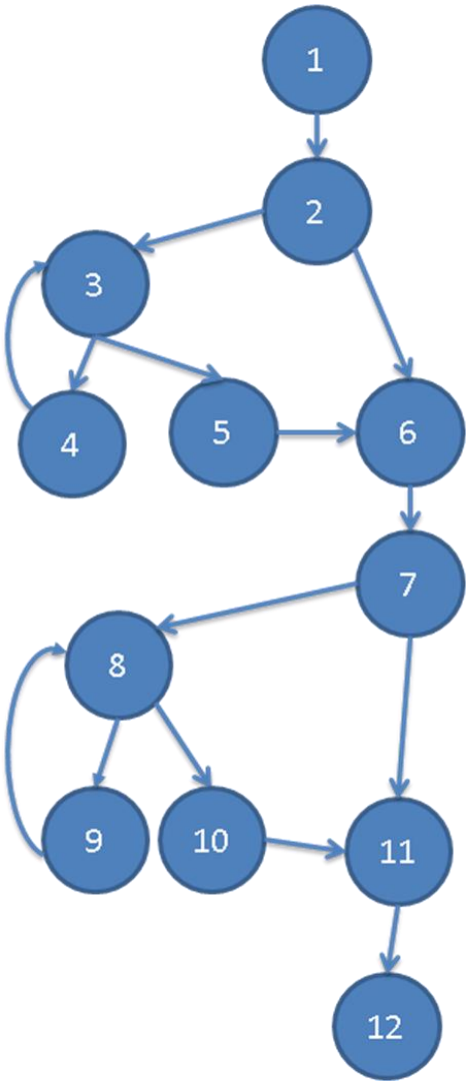


Figura 3.4: Grafo de flujo asociado al algoritmo asociarDesasociarAction().

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas descritas en el capítulo 2, epígrafe 2.5.2 **"Análisis de complejidad del algoritmo",** las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Fórmulas para calcular complejidad ciclomática.

1. **V (G) = (A – N) + 2.**
2. **V (G) = P + 1.**

3. V (G) = R.

Aplicando estas fórmulas al grafo de flujo de la figura 3.4 se obtienen los siguientes resultados:

Calculando mediante la fórmula 1:

V (G) = (15 - 12) + 2

V (G) = 5.

Calculando mediante la fórmula 2:

V (G) = 4 + 1

V (G) = 5.

Calculando mediante la fórmula 3:

V (G) = 5.

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 5, lo que significa que existen cinco posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo. En estas representaciones se subrayan los elementos de cada camino que los hacen independientes a los demás.

Camino básico #1:

1 – 2 – 6 – 7 – 11 - 12

Camino básico #2:

1 - 2 – 3 – 5 – 6 -7 – 11 - 12

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.

Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:

Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.

Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
Entrada: Se muestran los parámetros que entran al procedimiento
Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico # 1.

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El arreglo de objetos *asociados* posee los fondos que se desean desasociar y el arreglo de objetos *sinasociar* posee los fondos que se quieren asociar a la caja seleccionada, además del *id* de la caja seleccionada.
Condición de ejecución: Que la cantidad de elementos del arreglo *asociados* sea mayor que cero, o que el arreglo *sinasociar* tenga una cantidad de elementos mayor que cero.
Entrada:
asociados[{"idfondo":"8080000041","codigo":"123","descripcion":"123","nombre":"qwe","nombretipofondo":"Pepe","idtipofondo":"8080000056"}]
idcaja8080000028
sinasociar [ ]
Resultados esperados: Se espera que los fondos asociados se muestren en la tabla del panel a que pertenecen, si se realiza la operación asociar/ desasociar, se actualiza el panel mostrándose los datos actualizados.
Resultados: No hay información en la BD que coincida con esos parámetros de entrada, los datos no se muestran en la tabla del panel.

Caso de prueba para el camino básico # 2.

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El objeto *sinasociados* posee el parámetro idfondo de tipo entero.
Condición de ejecución: El id del fondo será igual 800000010, el id del fondo hijo será igual 800000010.
Entrada: $idfondo = 800000010.
Resultados esperados: Se espera que los fondos sin asociar se muestren en la tabla del panel a que pertenecen, si se realiza la operación asociar, se actualiza el panel mostrándose los datos actualizados.

Resultados: No hay información en la BD que coincida con esos parámetros de entrada, los datos no se muestran en la tabla del panel.

3.6 Aplicación de pruebas de caja negra

Para el desarrollo de este tipo de prueba se utilizará el requisito Adicionar Caja. El objetivo del mismo es persistir en la base de datos la información referente a las cajas que se han creado en la entidad dando la posibilidad de modificarla, asociarla o desasociarla y eliminarla.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar Caja.	El sistema permite adicionar cajas, persistiendo la información en la BD y buscarla en caso necesario.	EP 1.1: Adicionar una Caja introduciendo los datos correctamente al presionar el botón **Aceptar.**	– Se presiona el botón **Adicionar.** – Se introducen los datos correctamente. – Se presiona el botón **Aceptar.** – Se muestra la notificación "Se ha insertado satisfactoriamente la Caja.". – Se presiona el botón **Aceptar** de la notificación – Se muestran varios cuadros de textos a llenar.

			– Se muestran 2 opciones para introducir la fecha. – Se presiona el botón **Aceptar.** – Se muestra información satisfactoria.
		EP 1.2: Adicionar una Caja introduciendo datos inválidos al presionar el botón **Aceptar.**	– Se presiona el botón **Adicionar.** – Se introducen los datos correspondientes insertando algunos datos inválidos. – Se presiona el botón **Aceptar.**
		EP 1.3: Adicionar una Caja dejando campos requeridos en blanco al presionar el botón **Aceptar.**	– Se presiona el botón **Adicionar**. – Se introducen los datos correspondientes en el formulario y deje al menos un campo requerido en blanco. Se presiona el

		botón **Aceptar.**
	EP 1.4: Cancelar.	– Se presiona el botón **Adicionar**. – Se introducen o no los datos en los campos. Se presiona el botón **Cancelar**.

3.7 CONCLUSIONES DEL CAPÍTULO III

En este capítulo se hace un bosquejo de distintos aspectos con gran significado para las pruebas de software, resaltándose la importancia de las mismas en el desarrollo de un proyecto. Se aborda acerca de las pruebas de menor escala tales como pruebas de caja blanca y caja negra y las particularidades de cada una de ellas, además de la ejecución de las mismas analizando los resultados obtenidos.

CONCLUSIONES GENERALES.

Con el presente trabajo se concluye que el mismo a contribuido a fomentar los conocimientos sobre el funcionamiento de las caja de las empresas cubanas, además se evaluaron varios sistemas nacionales e internacionales con el objetivo de sacar de cada uno de estos las funcionalidades que más se adecuaban al sistema financiero nacional. Se usaron metodologías, herramientas y lenguajes acordes a las necesidades del país de migrar a Software Libre.

Durante el presente trabajo se realizaron valiosos aportes en la solución no solo a los procesos de caja, sino también al resto de los módulos del subsistema finanzas, pues se partió de un problema específico y se llegó a una solución genérica que abarca las necesidades del subsistema. Señalar que fue muy importante el diseño propuesto por los analistas, pues ahorró tiempo al equipo de desarrollo, avalado por un conjunto de métricas de diseño que arrojaron un resultado satisfactorio.

Finalmente se realizan un conjunto de pruebas al software en busca de posibles vulnerabilidades del mismo, con el objetivo de minimizar los posibles errores luego de la implantación del sistema en las entidades nacionales.

En resumen, se cumplió el objetivo propuesto de realizar la implementación del módulo, basado en los requisitos funcionales, obteniendo como resultado la implementación de un sistema capaz de gestionar, confiable y eficientemente, los procesos de Caja en las empresas cubanas a raíz de su incorporación al Sistema Integral de Gestión CEDRUX.

RECOMENDACIONES

Se recomienda:

- Continuar haciendo pruebas al módulo para aumentar su confiabilidad.
- Añadir una mayor cantidad de funcionalidades que son tratadas de igual forma en los módulos del subsistema finanzas, al componente **comunfinanzas,** con el objetivo de lograr una mayor integración en el subsistema.
- Realizar pruebas de conceptos al código del módulo Caja para así obtener un mejor rendimiento del mismo.

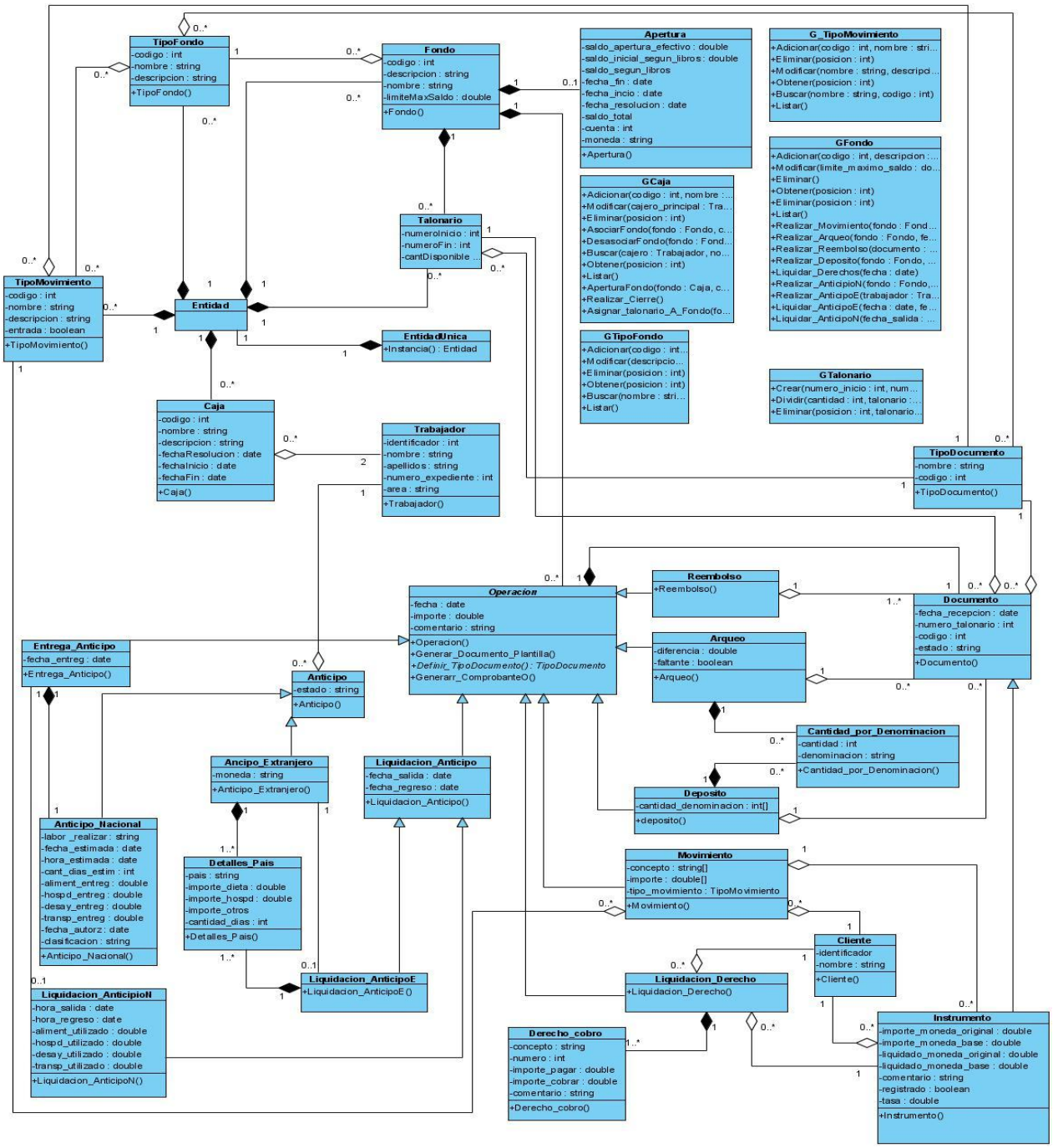
BIBLIOGRAFÍA

1. Assets S.A.Sistema de Gestión Integral, 2009. [Disponible en: http://www.assets.co.cu/
2. Openbravo ERP. Sistema de Gestión Integral Openbravo ERP, 2006, [Disponible en: http://www.openbravo.com/es/product/erp/]
3. Jacobson, I., G. Booch, and J. Rumbaugh, El Proceso Unificado de Desarrollo de Software. Madrid, Addison-Wesley, 1999.
4. Zend. Zend Framework, 2008. [Disponible en: http://framework.zend.com/]
5. Quer System Informática. Tecnología, 2008. [Disponible en: http://www.quersystem.com/index.php?option=com_content&view=article&id=20&Itemid=3]
6. Características de apache, 2004. [Disponible en: http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/x31.html]
7. Küng, S. Onken, L. Large, S. TortoiseSVN, 2007. [Disponible en: https://forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf]
8. Word Wide Web Consortium Guía Breve de Tecnologías XML, 2008. [Disponible en: http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML]
9. Thomas Weidner. Zend Framework, 2007. [Disponible en: http://www.slideshare.net/thomasw/phpbootcamp-zend-framework?src=related_normal&rel=184203]
10. Cifuentes, F. Elementos físicos y lógicos del computador, 2008. [Disponible en: http://www.slideshare.net/ares_egeo/elementos-fsicos-y-lgicos-del-computador-presentation]
11. Maestros del Web. ¿Qué es Javascript?, 2007. [Disponible en: http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/]
12. Martín, S. López, E. Boletín Nº8 Rama de Estudiantes de IEEE-UNED, 2007. [Disponible en: http://www.ieec.uned.es/ieee/investigacion/ieee_dieec/sb/boletin/boletin_8_Octubre_2007.pdf]
13. Java Injection Framework, 2008. [Disponible en: http://deveeel.files.wordpress.com/2008/09/google_guice.ppt]
14. Dante, M. Patrones de Diseño, 2008. [Disponible en: http://www.megaunder.com.ar/java/6948-patron-de-diseno-ebook.html]
15. PATRONES GRASP, 2006. [Disponible en: http://sophia.javeriana.edu.co/~lcdiaz/ADOO2006-3/grasp_cpaternostro-lvargas-jviafara.pdf]

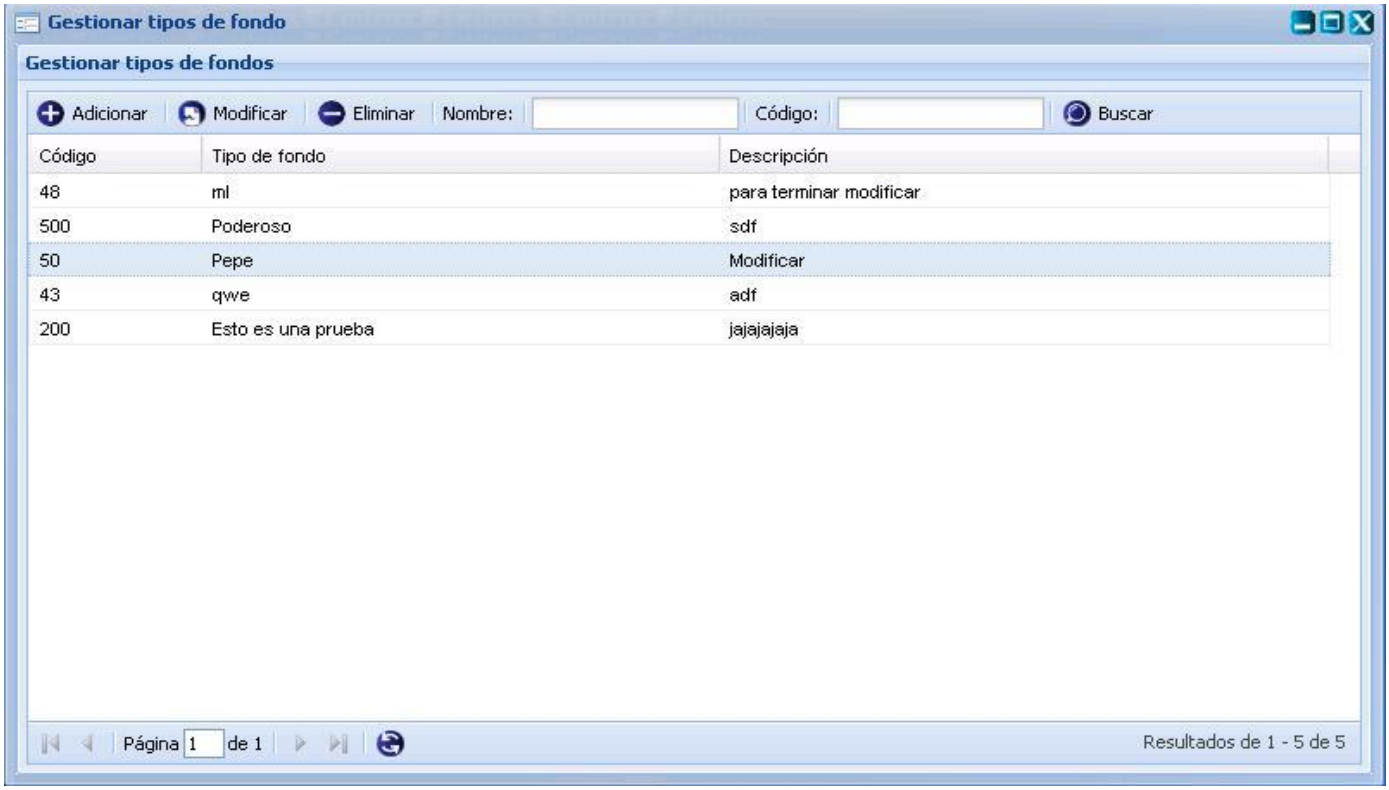
16. Javier. Complejidad Ciclomática, 2008. [Disponible en: http://javier.callon.org/complejidad-ciclomatica]
17. Roger S. Pressman. Un enfoque práctico. Ciudad Madrid, editorial Mc Graw Hill, 2002. 640 páginas
18. Etapa: Pruebas, 2008. [Disponible en: http://lsi.ugr.es/~arroyo/inndoc/doc/pruebas/pruebas_d.php]
19. Carvajal, A. Fase De Pruebas, 2008. [Disponible en: http://www.slideshare.net/angel.carvajal/fase-de-pruebas-angel-chucho-presentation]
20. Collado, M. Pruebas de software, 2003. [Disponible en: http://lml.ls.fi.upm.es/ftp/ed2/0203/Apuntes/pruebas.ppt]
21. Mañas, J. Prueba de Programas, 2004. [Disponible en: http://www.lab.dit.upm.es/~lprg/material/apuntes/pruebas/testing.htm#s22]
22. Rojas, J. Barrios, E. Métodos de prueba de caja negra, 2007. [Disponible en: http://www.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML%20-%20Pruebas%20de%20software/node28.html]

ANEXOS

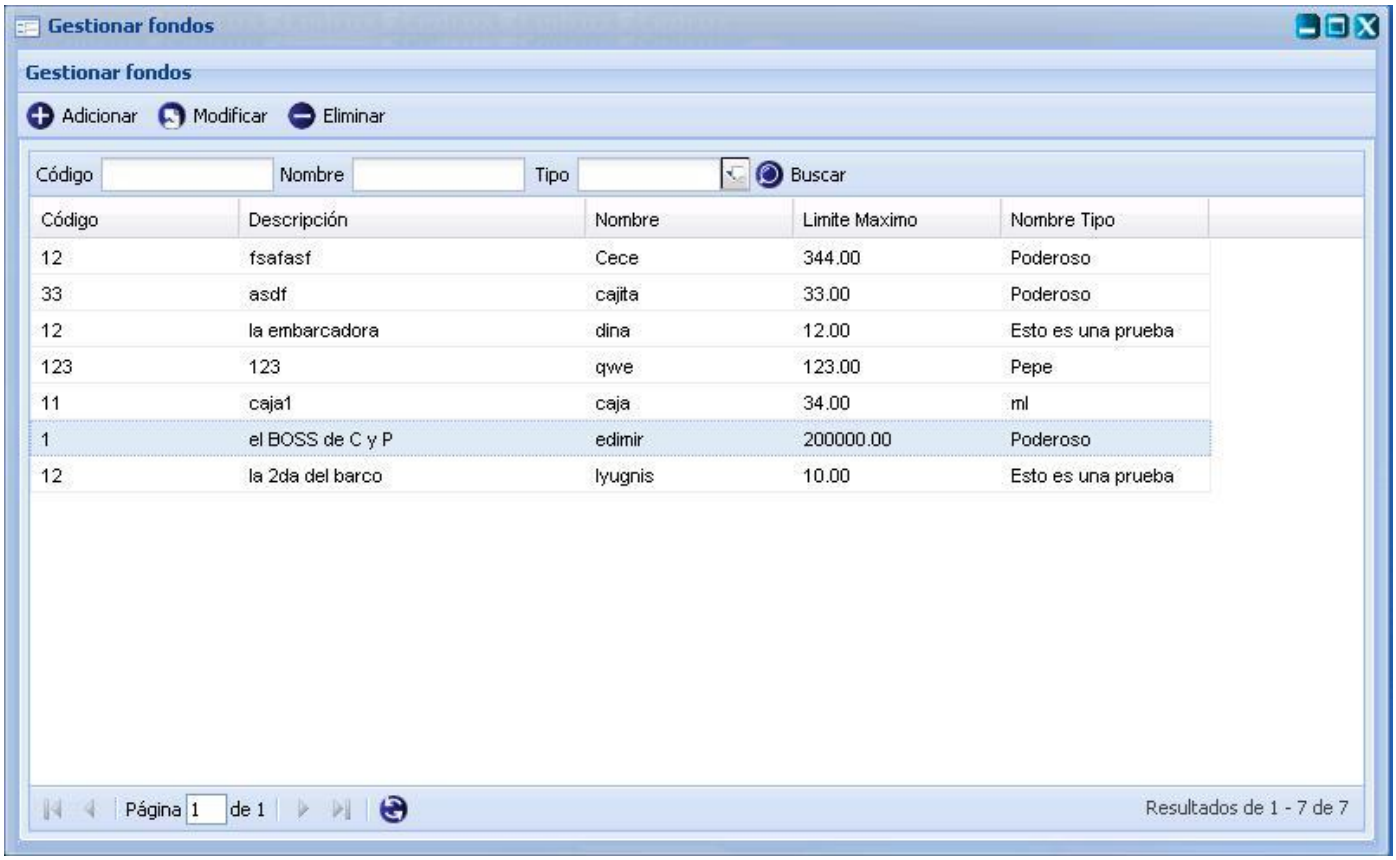
Anexo 1. Diagrama de clases del diseño



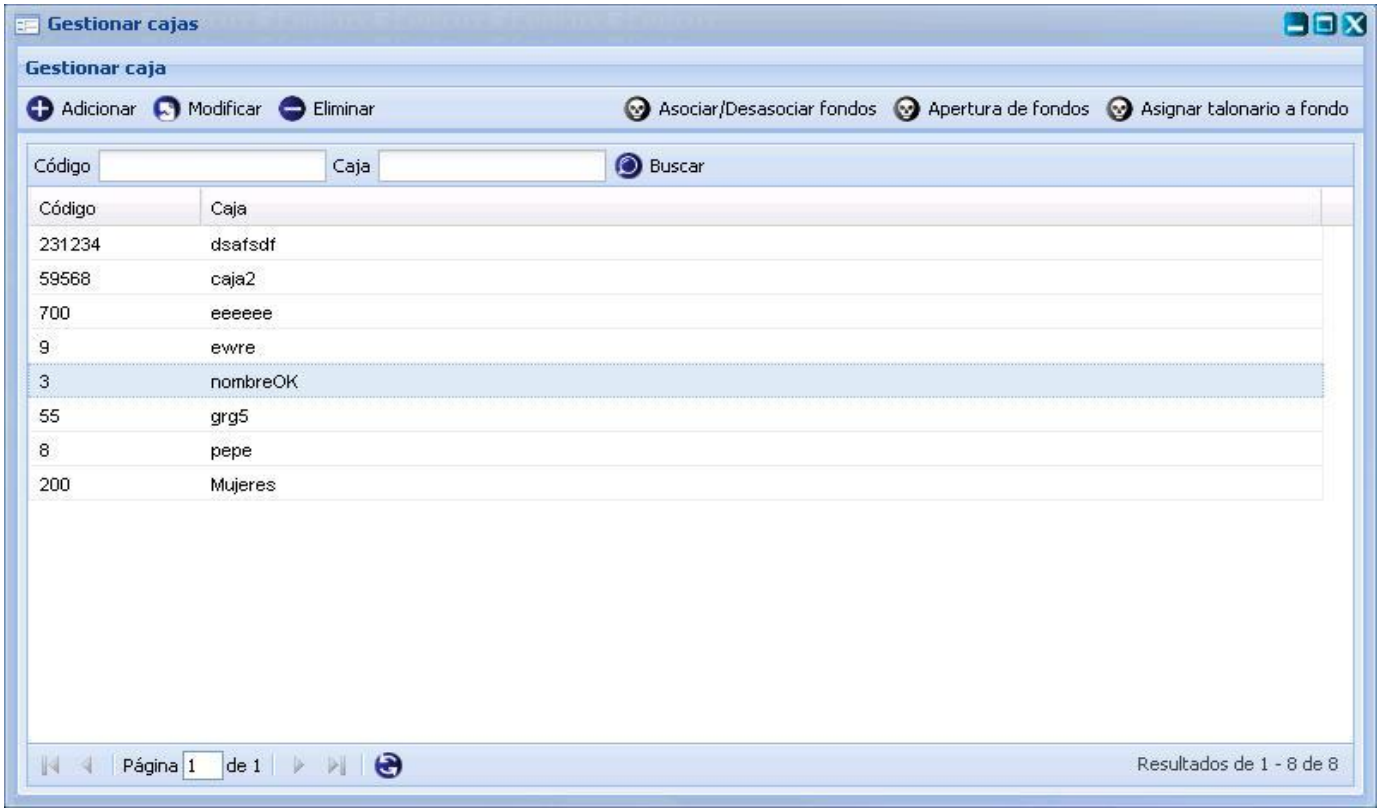
Anexo 2. Gestionar Tipo de Fondo



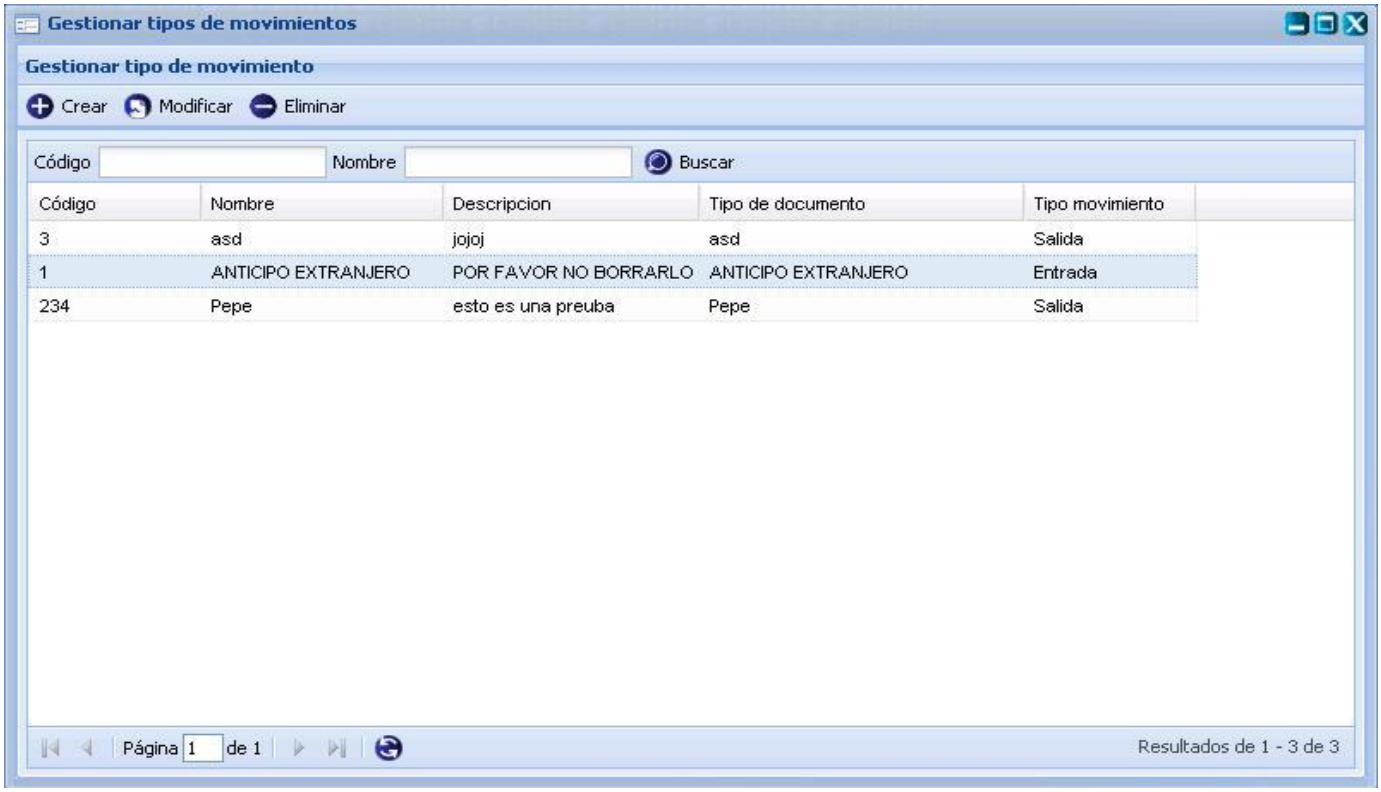
Anexo 3. Gestionar Fondo



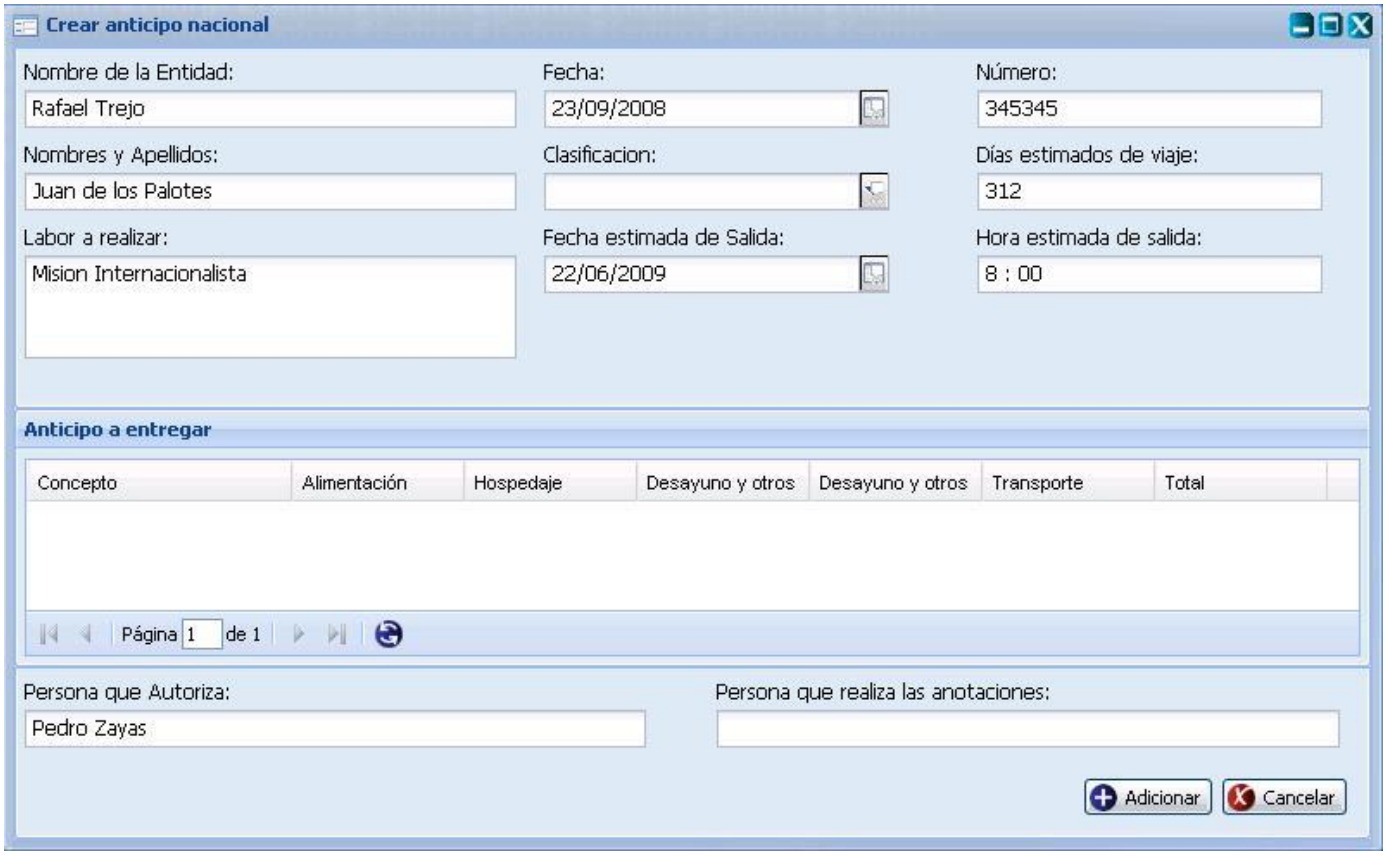
Anexo 4. Gestionar Caja



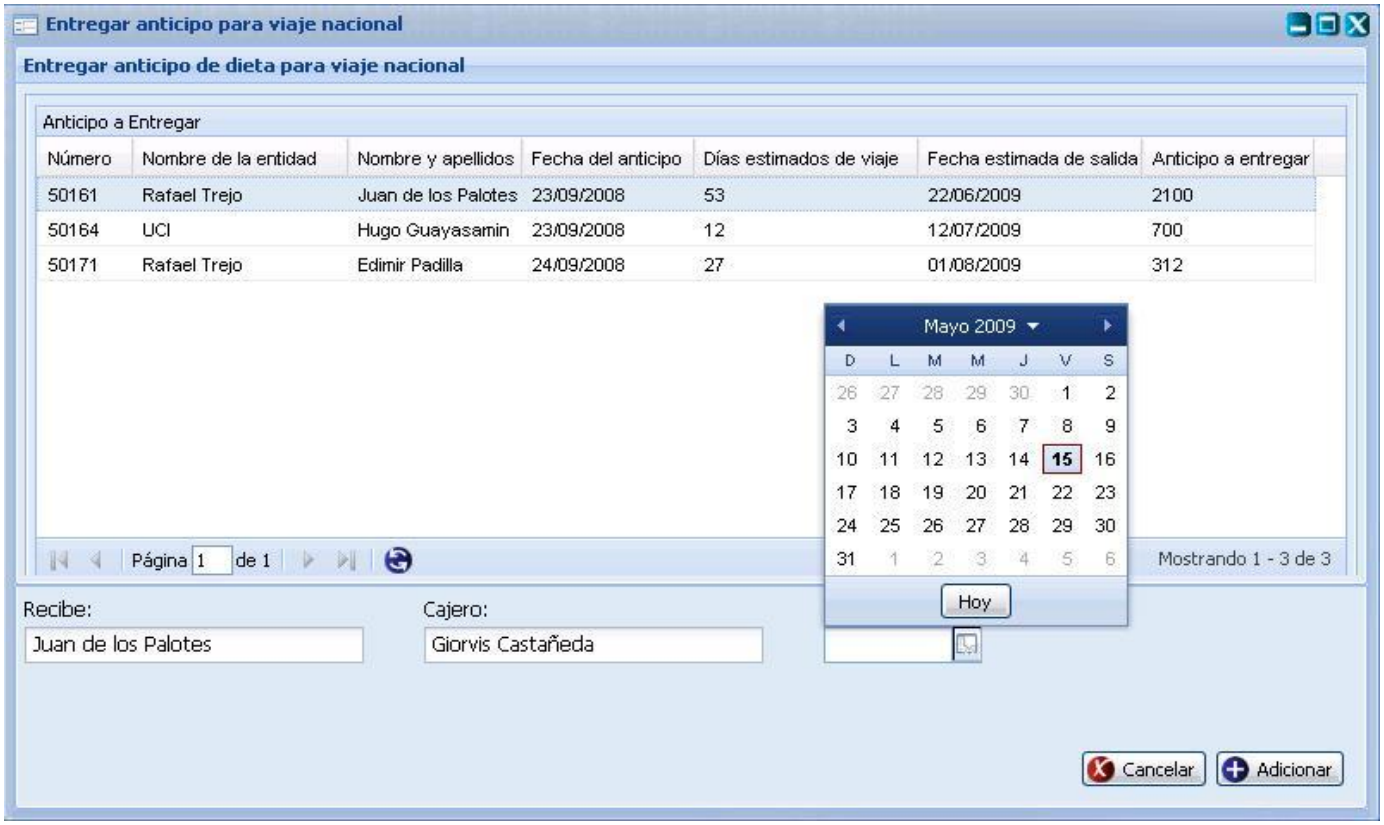
Anexo 5. Gestionar Tipo de Movimiento



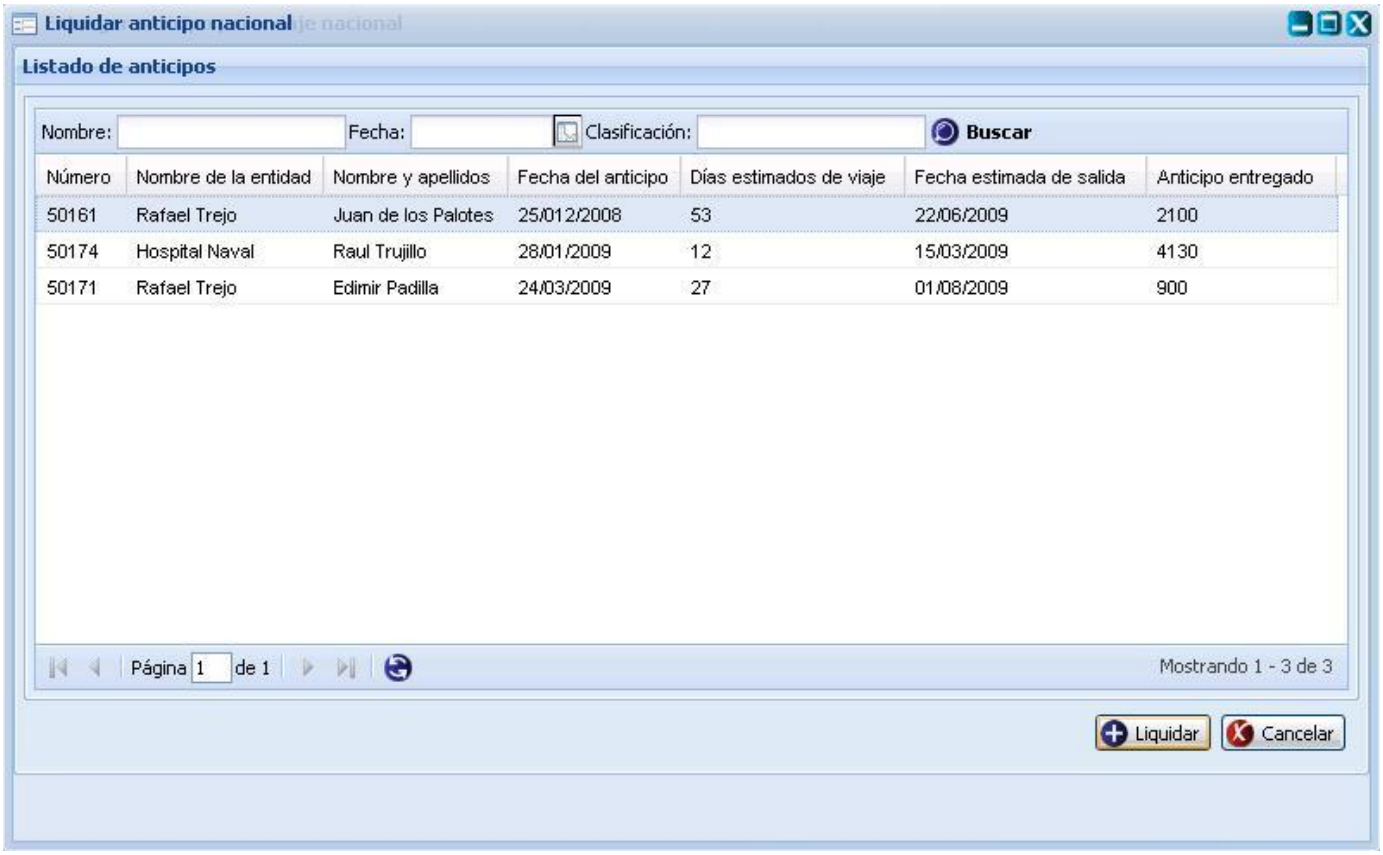
Anexo 6. Registrar Anticipo Nacional



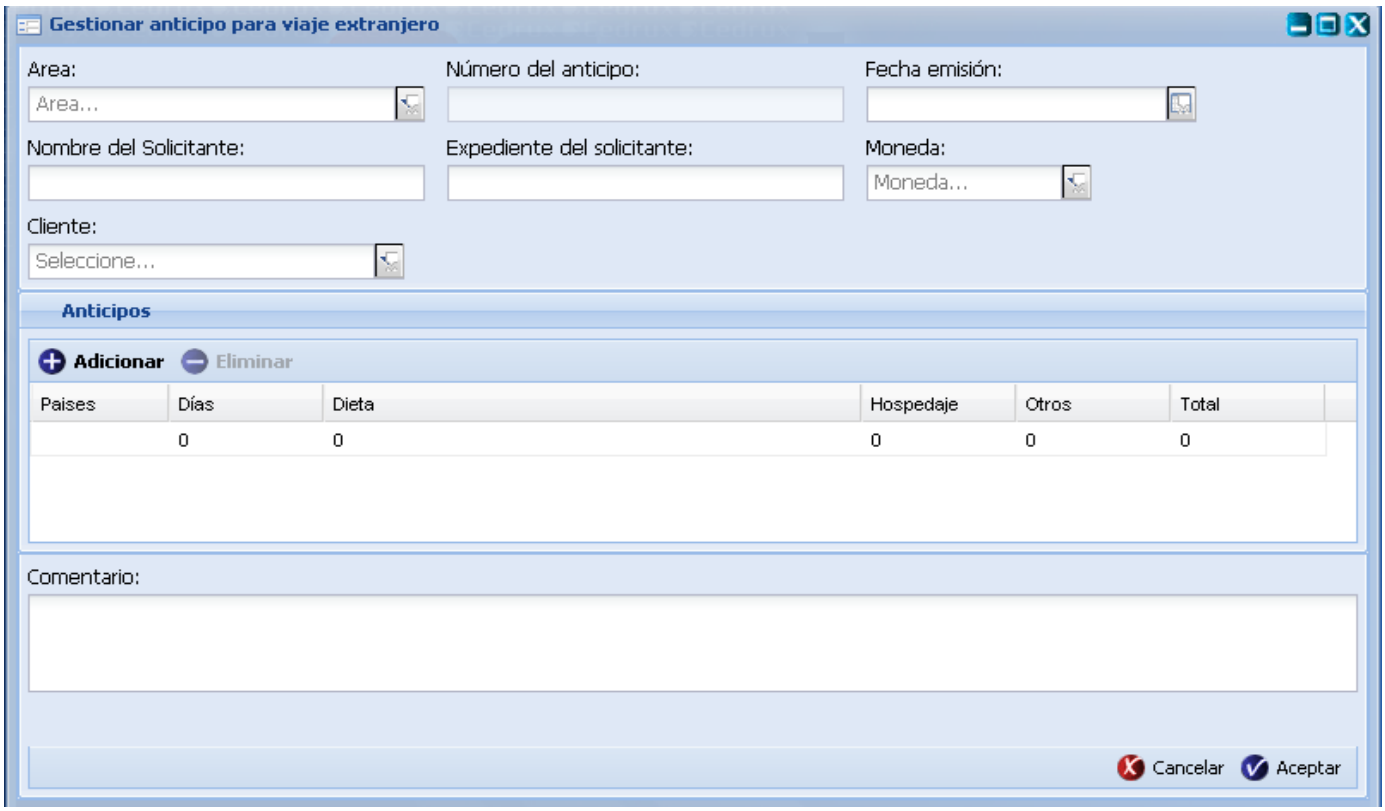
Anexo 7. Entregar Anticipo Nacional



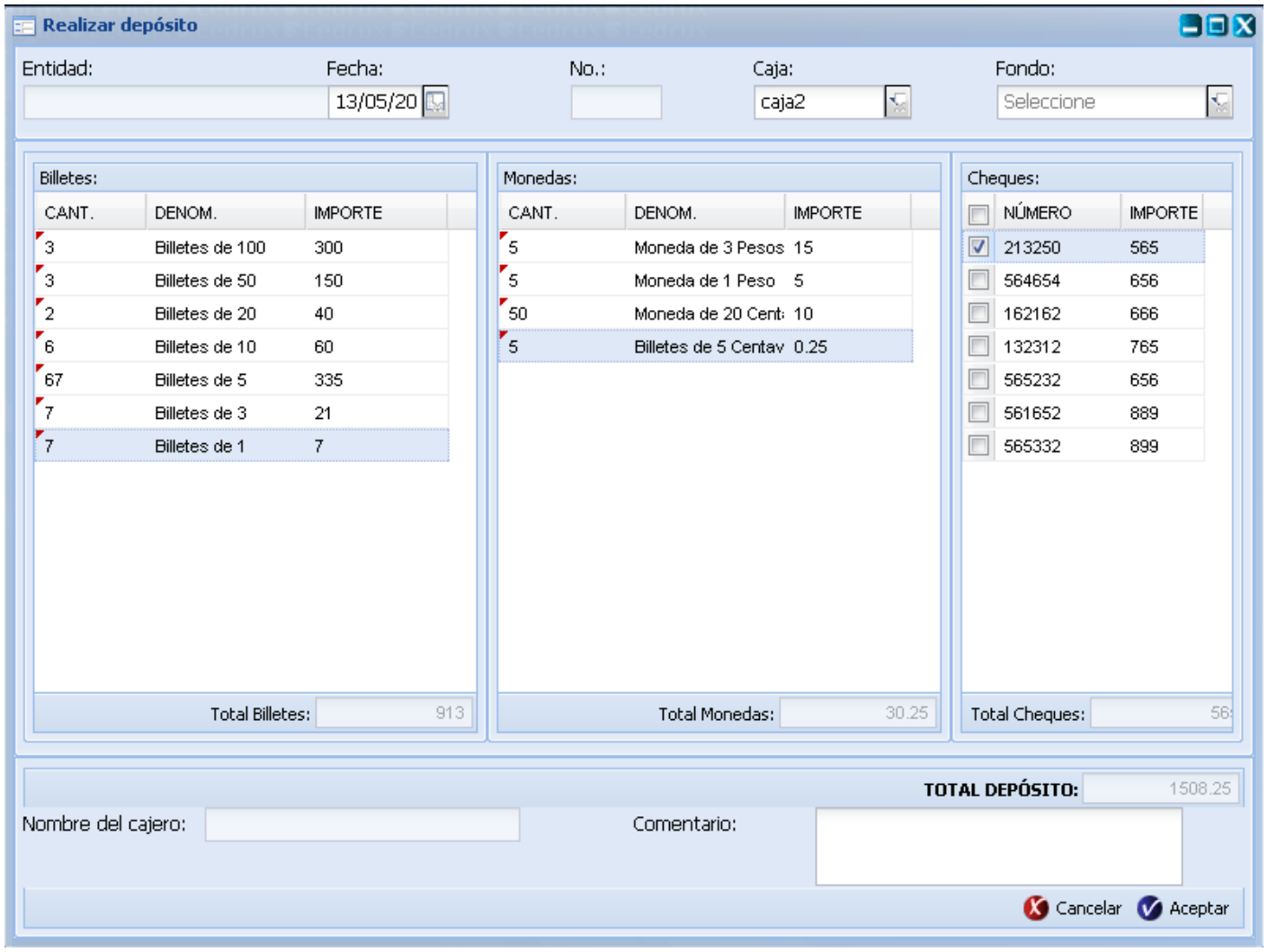
Anexo 8. Liquidar anticipo nacional.



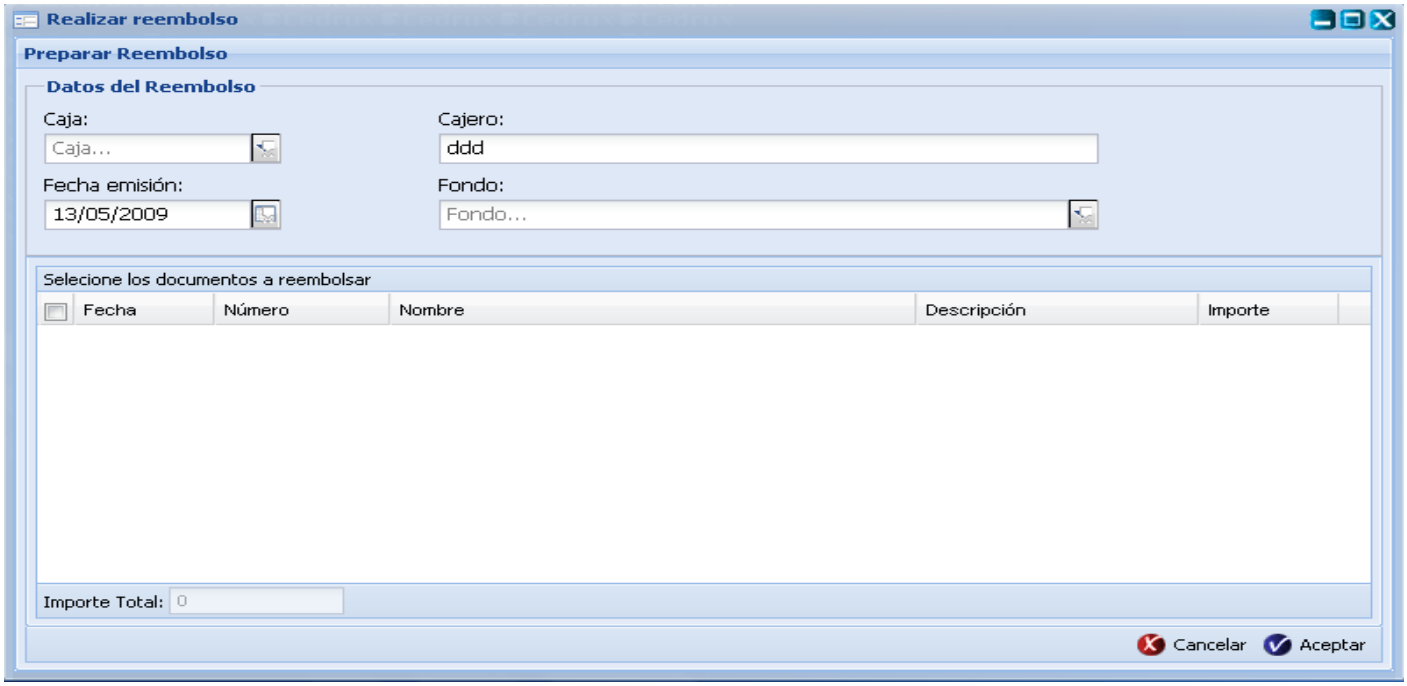
Anexo 9. Registrar anticipo para viaje extranjero



Anexo 10. Realizar depósito

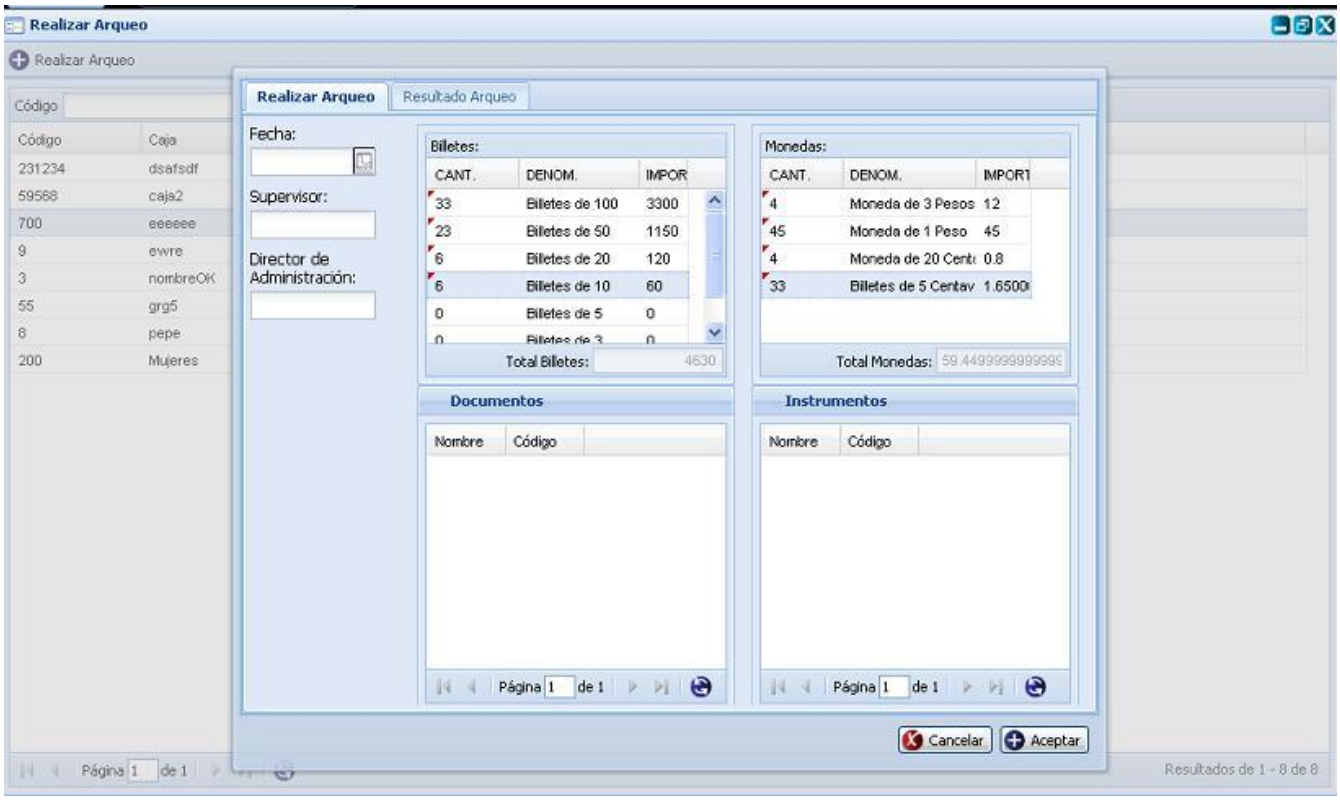


Anexo 11. Realizar reembolso



Anexo 12. Realizar movimiento genérico.

Anexo 13. Realizar Arqueo



Anexo 14. Instrumento de medición de la métrica Tamaño operacional de clase (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.

Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

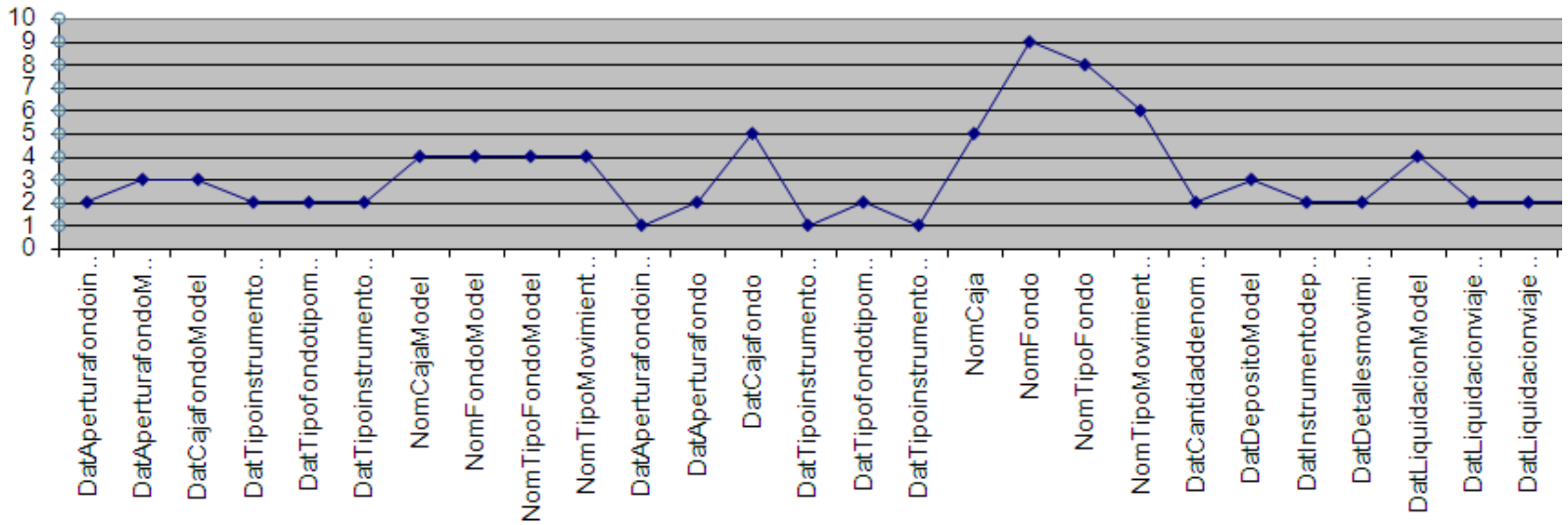
Subsistema	**Clase**	**Cantidad de Procedimientos**	**Responsabilidad**	**Complejidad**	**Reutilización**
Configuraciones	DatAperturafondointrumentoModel	2	Baja	Baja	Alta
Configuraciones	DatAperturafondoModel	3	Media	Media	Media
Configuraciones	DatCajafondoModel	3	Media	Media	Media
Configuraciones	DatTipoinstrumentotipomovimientoModel	2	Baja	Baja	Alta
Configuraciones	DatTipofondotipomovimientoModel	2	Baja	Baja	Alta
Configuraciones	DatTipoinstrumentotipofondoModel	2	Baja	Baja	Alta
Configuraciones	NomCajaModel	4	Media	Media	Media

Configuraciones	NomFondoModel	4	Media	Media	Media
Configuraciones	NomTipoFondoModel	4	Media	Media	Media
Configuraciones	NomTipoMovimientocajaModel	4	Media	Media	Media
Configuraciones	DatAperturafondointrumento	1	Baja	Baja	Alta
Configuraciones	DatAperturafondo	2	Baja	Baja	Alta
Configuraciones	DatCajafondo	5	Media	Media	Media
Configuraciones	DatTipoinstrumentotipomovimiento	1	Baja	Baja	Alta
Configuraciones	DatTipofondotipomovimiento	2	Baja	Baja	Alta
Configuraciones	DatTipoinstrumentotipofondo	1	Baja	Baja	Alta
Configuraciones	NomCaja	5	Media	Media	Media
Configuraciones	NomFondo	9	Alta	Alta	Baja
Configuraciones	NomTipoFondo	8	Alta	Alta	Baja
Configuraciones	NomTipoMovimientocaja	6	Alta	Alta	Baja
Movimientos	DatCantidaddenominacionModel	2	Baja	Baja	Alta
Movimientos	DatDepositoModel	3	Media	Media	Media
Movimientos	DatInstrumentodepositoModel	2	Baja	Baja	Alta
Movimientos	DatDetallesmovimientoModel	2	Baja	Baja	Alta
Movimientos	DatLiquidacionModel	4	Media	Media	Media
Movimientos	DatLiquidacionviajeextranjeroModel	2	Baja	Baja	Alta
Movimientos	DatLiquidacionviajeextranjeropaisModel	2	Baja	Baja	Alta
Movimientos	DatLiquidacionviajenacionalModel	2	Baja	Baja	Alta
Movimientos	DatMovimientoModel	3	Media	Media	Media
Movimientos	DatMovimientogenericoModel	4	Media	Media	Media
Movimientos	DatMovimientoinstrumentoModel	2	Baja	Baja	Alta
Movimientos	DatReembolsoModel	3	Media	Media	Media
Movimientos	NomClasificacionModel	2	Baja	Baja	Alta
Movimientos	DatAnticipoModel	3	Media	Media	Media
Movimientos	DatAnticipoviajeextranjeroModel	2	Baja	Baja	Alta
Movimientos	DatAnticipoviajeextranjeropaisModel	2	Baja	Baja	Alta
Movimientos	DatAnticipoviajenacionalModel	2	Baja	Baja	Alta
Movimientos	DatDocumentoreembolsoModel	2	Baja	Baja	Alta

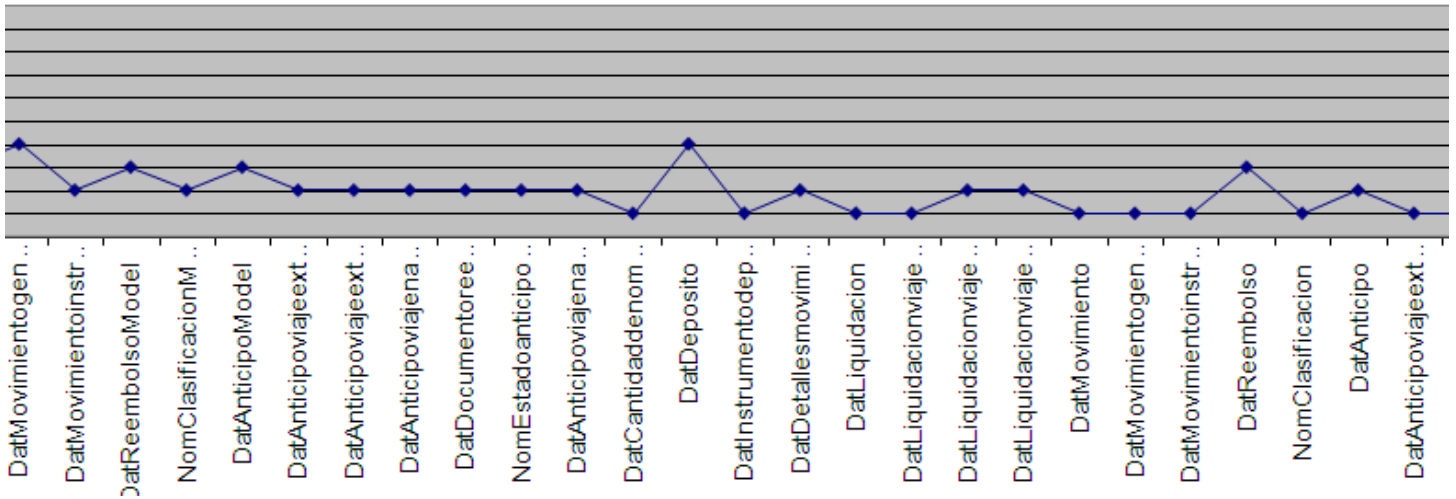
Movimientos	NomEstadoanticipoModel	2	Baja	Baja	Alta
Movimientos	DatAnticipoviajenacionalentregadoModel	2	Baja	Baja	Alta
Movimientos	DatCantidaddenominacion	1	Baja	Baja	Alta
Movimientos	DatDeposito	4	Media	Media	Media
Movimientos	DatInstrumentodeposito	1	Baja	Baja	Alta
Movimientos	DatDetallesmovimiento	2	Baja	Baja	Alta
Movimientos	DatLiquidacion	1	Baja	Baja	Alta
Movimientos	DatLiquidacionviajeextranjero	1	Baja	Baja	Alta
Movimientos	DatLiquidacionviajeextranjeropais	2	Baja	Baja	Alta
Movimientos	DatLiquidacionviajenacional	2	Baja	Baja	Alta
Movimientos	DatMovimiento	1	Baja	Baja	Alta
Movimientos	DatMovimientogenerico	1	Baja	Baja	Alta
Movimientos	DatMovimientoinstrumento	1	Baja	Baja	Alta
Movimientos	DatReembolso	3	Media	Media	Media
Movimientos	NomClasificacion	1	Baja	Baja	Alta
Movimientos	DatAnticipo	2	Baja	Baja	Alta
Movimientos	DatAnticipoviajeextranjero	1	Baja	Baja	Alta
Movimientos	DatAnticipoviajeextranjeropais	1	Baja	Baja	Alta
Movimientos	DatAnticipoviajenacional	2	Baja	Baja	Alta
Movimientos	DatDocumentoreembolso	1	Baja	Baja	Alta
Movimientos	NomEstadoanticipo	1	Baja	Baja	Alta
Movimientos	DatAnticipoviajenacionalentregado	1	Baja	Baja	Alta
Arqueos	ArqueosModel	3	Media	Media	Media
Arqueos	DatArqueodocumentoModel	2	Baja	Baja	Alta
Arqueos	DatArqueo	3	Media	Media	Media
Arqueos	DatArqueodocumento	4	Media	Media	Media
Comunfinanzas	DatContratofnzModel	3	Media	Media	Media
Comunfinanzas	DatDocumentofnzModel	4	Media	Media	Media
Comunfinanzas	DatModelofnzModel	3	Media	Media	Media
Comunfinanzas	NomEstadodocfnzModel	3	Media	Media	Media
Comunfinanzas	DatContratofnz	2	Baja	Baja	Alta

Comunfinanzas	DatDocumentofnz	4	Media	Media	Media
Comunfinanzas	DatModelofnz	2	Baja	Baja	Alta
Comunfinanzas	NomEstadodocfnz	1	Baja	Baja	Alta
Comunfinanzas	DatAnticipofnzModel	3	Media	Media	Media
Comunfinanzas	DatInstrumentofnzModel	4	Media	Media	Media
Comunfinanzas	DatLetrafnzModel	4	Media	Media	Media
Comunfinanzas	DatLetrafnz	3	Media	Media	Media
Comunfinanzas	DatInstrumentofnz	3	Media	Media	Media
Comunfinanzas	DatAnticipofnz	4	Media	Media	Media
Comunfinanzas	NomTalonariocajafnzModel	4	Media	Media	Media
Comunfinanzas	NomTalonariofnzModel	5	Media	Media	Media
Comunfinanzas	NomTalonariobancofnzModel	4	Media	Media	Media
Comunfinanzas	NomTalonariocajafnz	2	Baja	Baja	Alta
Comunfinanzas	NomTalonariofnz	6	Alta	Alta	Baja
Comunfinanzas	NomTalonariobancofnz	2	Baja	Baja	Alta

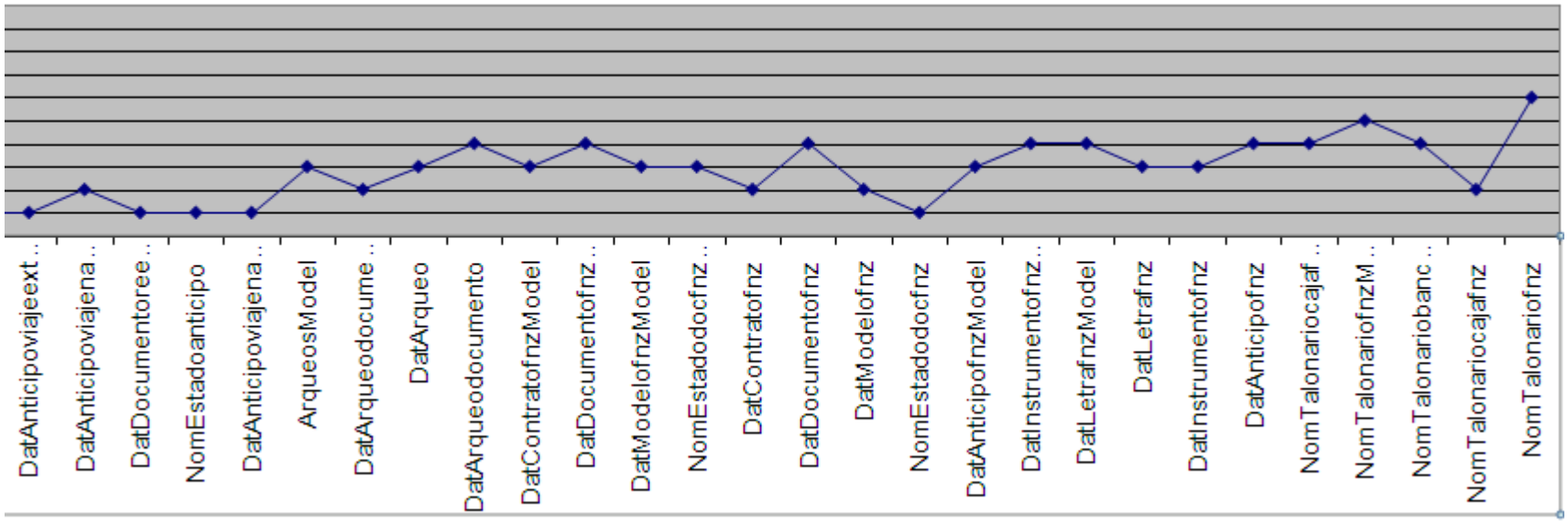
Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización)



Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización), parte 1



Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización), parte 2



Gráfica de los resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización), parte 3

Anexo 15. Instrumento de medición de la métrica Relaciones entre clases (RC)

	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
	Categoria	Criterio

Complejidad Mant.	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.
	Categoria	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.
	Categoria	Criterio
Cantidad de Pruebas	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Rango de valores de para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC.

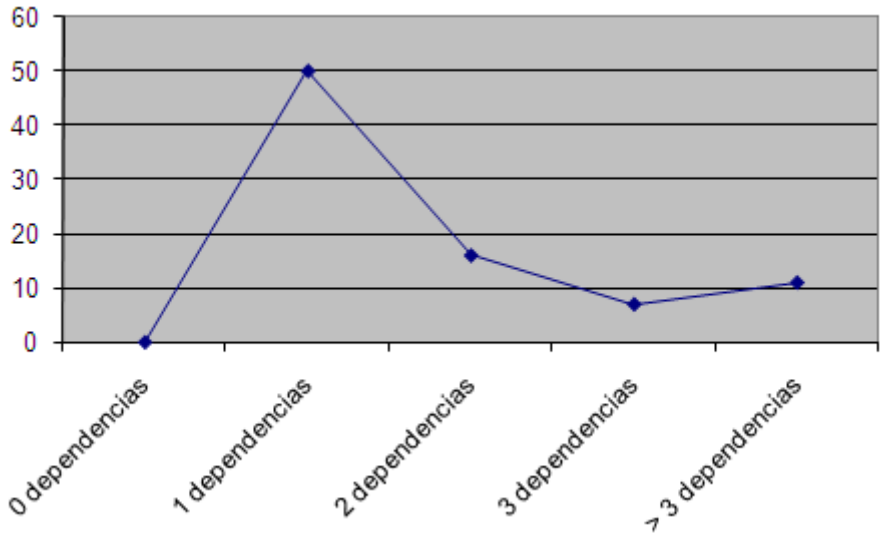
Subsistema	**Clase**	**Cantidad de Relaciones de Uso**	**Acoplamiento**	**Complejidad Mant.**	**Reutilización**	**Cantidad de Pruebas**
Configuraciones	DatAperturafondointrumentoModel	1	Bajo	Baja	Alta	Baja
Configuraciones	DatAperturafondoModel	2	Medio	Media	Media	Media
Configuraciones	DatCajafondoModel	2	Medio	Media	Media	Media
Configuraciones	DatTipoinstrumentotipomovimientoModel	1	Bajo	Baja	Alta	Baja
Configuraciones	DatTipofondotipomovimientoModel	1	Bajo	Baja	Alta	Baja

Configuraciones	DatTipoinstrumentotipofondoModel	1	Bajo	Baja	Alta	Baja
Configuraciones	NomCajaModel	3	Alto	Media	Media	Media
Configuraciones	NomFondoModel	4	Alto	Alta	Baja	Alta
Configuraciones	NomTipoFondoModel	4	Alto	Alta	Baja	Alta
Configuraciones	NomTipoMovimientocajaModel	2	Medio	Media	Media	Media
Configuraciones	DatAperturafondointrumento	1	Bajo	Baja	Alta	Baja
Configuraciones	DatAperturafondo	3	Alto	Media	Media	Media
Configuraciones	DatCajafondo	4	Alto	Alta	Baja	Alta
Configuraciones	DatTipoinstrumentotipomovimiento	1	Bajo	Baja	Alta	Baja
Configuraciones	DatTipofondotipomovimiento	1	Bajo	Baja	Alta	Baja
Configuraciones	DatTipoinstrumentotipofondo	1	Bajo	Baja	Alta	Baja
Configuraciones	NomCaja	3	Alto	Media	Media	Media
Configuraciones	NomFondo	5	Alto	Alta	Baja	Alta
Configuraciones	NomTipoFondo	4	Alto	Alta	Baja	Alta
Configuraciones	NomTipoMovimientocaja	4	Alto	Alta	Baja	Alta
Movimientos	DatCantidaddenominacionModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatDepositoModel	3	Alto	Media	Media	Media
Movimientos	DatInstrumentodepositoModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatDetallesmovimientoModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacionModel	2	Medio	Media	Media	Media
Movimientos	DatLiquidacionviajeextranjeroModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacionviajeextranjeropaisModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacionviajenacionalModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatMovimientoModel	2	Medio	Media	Media	Media
Movimientos	DatMovimientogenericoModel	2	Medio	Media	Media	Media
Movimientos	DatMovimientoinstrumentoModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatReembolsoModel	2	Medio	Media	Media	Media
Movimientos	NomClasificacionModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipoModel	2	Medio	Media	Media	Media
Movimientos	DatAnticipoviajeextranjeroModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipoviajeextranjeropaisModel	1	Bajo	Baja	Alta	Baja

Movimientos	DatAnticipoviajenacionalModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatDocumentoreembolsoModel	1	Bajo	Baja	Alta	Baja
Movimientos	NomEstadoanticipoModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipoviajenacionalentregadoModel	1	Bajo	Baja	Alta	Baja
Movimientos	DatCantidaddenominacion	1	Bajo	Baja	Alta	Baja
Movimientos	DatDeposito	4	Alto	Alta	Baja	Alta
Movimientos	DatInstrumentodeposito	1	Bajo	Baja	Alta	Baja
Movimientos	DatDetallesmovimiento	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacion	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacionviajeextranjero	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacionviajeextranjeropais	1	Bajo	Baja	Alta	Baja
Movimientos	DatLiquidacionviajenacional	1	Bajo	Baja	Alta	Baja
Movimientos	DatMovimiento	1	Bajo	Baja	Alta	Baja
Movimientos	DatMovimientogenerico	1	Bajo	Baja	Alta	Baja
Movimientos	DatMovimientoinstrumento	1	Bajo	Baja	Alta	Baja
Movimientos	DatReembolso	2	Medio	Media	Media	Media
Movimientos	NomClasificacion	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipo	4	Alto	Alta	Baja	Alta
Movimientos	DatAnticipoviajeextranjero	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipoviajeextranjeropais	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipoviajenacional	1	Bajo	Baja	Alta	Baja
Movimientos	DatDocumentoreembolso	1	Bajo	Baja	Alta	Baja
Movimientos	NomEstadoanticipo	1	Bajo	Baja	Alta	Baja
Movimientos	DatAnticipoviajenacionalentregado	1	Bajo	Baja	Alta	Baja
Arqueos	ArqueosModel	2	Medio	Media	Media	Media
Arqueos	DatArqueodocumentoModel	1	Bajo	Baja	Alta	Baja
Arqueos	DatArqueo	5	Alto	Alta	Baja	Alta
Arqueos	DatArqueodocumento	1	Bajo	Baja	Alta	Baja
Comunfinanzas	DatContratofnzModel	1	Bajo	Baja	Alta	Baja
Comunfinanzas	DatDocumentofnzModel	3	Alto	Media	Media	Media
Comunfinanzas	DatModelofnzModel	1	Bajo	Baja	Alta	Baja

Comunfinanzas	NomEstadodocfnzModel	1	Bajo	Baja	Alta	Baja
Comunfinanzas	DatContratofnz	1	Bajo	Baja	Alta	Baja
Comunfinanzas	DatDocumentofnz	3	Alto	Media	Media	Media
Comunfinanzas	DatModelofnz	1	Bajo	Baja	Alta	Baja
Comunfinanzas	NomEstadodocfnz	1	Bajo	Baja	Alta	Baja
Comunfinanzas	DatAnticipofnzModel	2	Medio	Media	Media	Media
Comunfinanzas	DatInstrumentofnzModel	3	Alto	Media	Media	Media
Comunfinanzas	DatLetrafnzModel	2	Medio	Media	Media	Media
Comunfinanzas	DatLetrafnz	1	Bajo	Baja	Alta	Baja
Comunfinanzas	DatInstrumentofnz	2	Medio	Media	Media	Media
Comunfinanzas	DatAnticipofnz	2	Medio	Media	Media	Media
Comunfinanzas	NomTalonariocajafnzModel	2	Medio	Media	Media	Media
Comunfinanzas	NomTalonariofnzModel	4	Alto	Alta	Baja	Alta
Comunfinanzas	NomTalonariobancofnzModel	2	Medio	Media	Media	Media
Comunfinanzas	NomTalonariocajafnz	1	Bajo	Baja	Alta	Baja
Comunfinanzas	NomTalonariofnz	5	Alto	Alta	Baja	Alta
Comunfinanzas	NomTalonariobancofnz	1	Bajo	Baja	Alta	Baja

Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas)



Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los valores.

Anexo 16 Instrumento de medición de la métrica Profundidad de Herencia (PH)

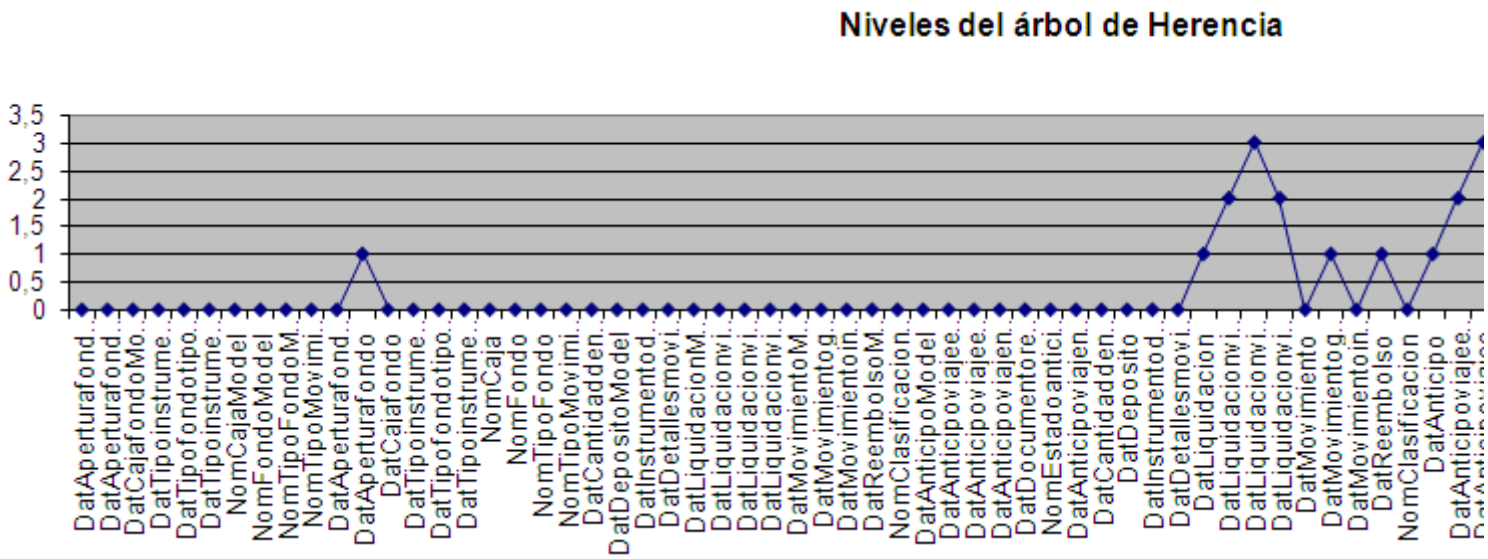
Subsistema	Clase	Clase padre	Niveles del arbol de herencia
Configuraciones	DatAperturafondointrumentoModel		0
Configuraciones	DatAperturafondoModel		0
Configuraciones	DatCajafondoModel		0
Configuraciones	DatTipoinstrumentotipomovimientoModel		0
Configuraciones	DatTipofondotipomovimientoModel		0
Configuraciones	DatTipoinstrumentotipofondoModel		0
Configuraciones	NomCajaModel		0
Configuraciones	NomFondoModel		0

Configuraciones	NomTipoFondoModel		0
Configuraciones	NomTipoMovimientocajaModel		0
Configuraciones	DatAperturafondointrumento		0
Configuraciones	DatAperturafondo	DatCajafondo	1
Configuraciones	DatCajafondo		0
Configuraciones	DatTipoinstrumentotipomovimiento		0
Configuraciones	DatTipofondotipomovimiento		0
Configuraciones	DatTipoinstrumentotipofondo		0
Configuraciones	NomCaja		0
Configuraciones	NomFondo		0
Configuraciones	NomTipoFondo		0
Configuraciones	NomTipoMovimientocaja		0
Movimientos	DatCantidaddenominacionModel		0
Movimientos	DatDepositoModel		0
Movimientos	DatInstrumentodepositoModel		0
Movimientos	DatDetallesmovimientoModel		0
Movimientos	DatLiquidacionModel		0
Movimientos	DatLiquidacionviajeextranjeroModel		0
Movimientos	DatLiquidacionviajeextranjeropaisModel		0
Movimientos	DatLiquidacionviajenacionalModel		0
Movimientos	DatMovimientoModel		0
Movimientos	DatMovimientogenericoModel		0
Movimientos	DatMovimientoinstrumentoModel		0
Movimientos	DatReembolsoModel		0
Movimientos	NomClasificacionModel		0
Movimientos	DatAnticipoModel		0
Movimientos	DatAnticipoviajeextranjeroModel		0
Movimientos	DatAnticipoviajeextranjeropaisModel		0
Movimientos	DatAnticipoviajenacionalModel		0
Movimientos	DatDocumentoreembolsoModel		0
Movimientos	NomEstadoanticipoModel		0

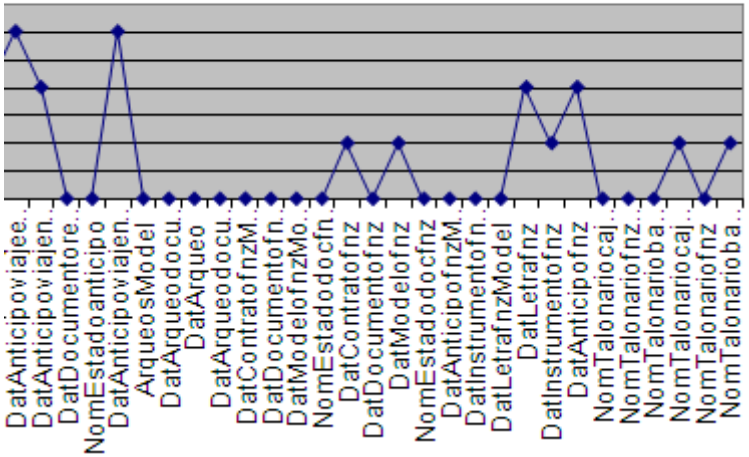
Movimientos	DatAnticipoviajenacionalentregadoModel		0
Movimientos	DatCantidaddenominacion		0
Movimientos	DatDeposito		0
Movimientos	DatInstrumentodeposito		0
Movimientos	DatDetallesmovimiento		0
Movimientos	DatLiquidacion	DatMovimiento	1
Movimientos	DatLiquidacionviajeextranjero	DatLiquidacion	2
Movimientos	DatLiquidacionviajeextranjeropais	DatLiquidacionviajeextranjero	3
Movimientos	DatLiquidacionviajenacional	DatLiquidacion	2
Movimientos	DatMovimiento		0
Movimientos	DatMovimientogenerico	DatMovimiento	1
Movimientos	DatMovimientoinstrumento		0
Movimientos	DatReembolso	DatMovimiento	1
Movimientos	NomClasificacion		0
Movimientos	DatAnticipo	DatMovimiento	1
Movimientos	DatAnticipoviajeextranjero	DatAnticipo	2
Movimientos	DatAnticipoviajeextranjeropais	DatAnticipoviajeextranjero	3
Movimientos	DatAnticipoviajenacional	DatAnticipo	2
Movimientos	DatDocumentoreembolso		0
Movimientos	NomEstadoanticipo		0
Movimientos	DatAnticipoviajenacionalentregado	DatAnticipoviajenacional	3
Arqueos	ArqueosModel		0
Arqueos	DatArqueodocumentoModel		0
Arqueos	DatArqueo		0
Arqueos	DatArqueodocumento		0
Comunfinanzas	DatContratofnzModel		0
Comunfinanzas	DatDocumentofnzModel		0
Comunfinanzas	DatModelofnzModel		0
Comunfinanzas	NomEstadodocfnzModel		0
Comunfinanzas	DatContratofnz	DatDocumentofnz	1
Comunfinanzas	DatDocumentofnz		0

Comunfinanzas	DatModelofnz	DatDocumentofnz	1
Comunfinanzas	NomEstadodocfnz		0
Comunfinanzas	DatAnticipofnzModel		0
Comunfinanzas	DatInstrumentofnzModel		0
Comunfinanzas	DatLetrafnzModel		0
Comunfinanzas	DatLetrafnz	DatInstrumentofnz	2
Comunfinanzas	DatInstrumentofnz	DatDocumentofnz	1
Comunfinanzas	DatAnticipofnz	DatInstrumentofnz	2
Comunfinanzas	NomTalonariocajafnzModel		0
Comunfinanzas	NomTalonariofnzModel		0
Comunfinanzas	NomTalonariobancofnzModel		0
Comunfinanzas	NomTalonariocajafnz	NomTalonariofnz	1
Comunfinanzas	NomTalonariofnz		0
Comunfinanzas	NomTalonariobancofnz	NomTalonariofnz	1

Resultados de la evaluación de la métrica PH.



Gráfica de los resultados de la evaluación de la métrica PH. Parte 1



Gráfica de los resultados de la evaluación de la métrica PH. Parte 2

Anexo 17 Instrumento de medición de la métrica Número de Descendientes (ND)

	Categoria	Criterio
Reutilización	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoria	Criterio
Abstracción	Indefinida	> 5
	Afectada	Entre 2 y 5
	Definida	<= 2

	Categoria	Criterio
Cohesión	Baja	> 5
	Media	Entre 2 y 5
	Alta	<= 2

	Categoria	Criterio
Cantidad de Pruebas	Baja	<= 2
	Media	Entre 2 y 5
	Alta	> 5

Rango de valores de para la evaluación técnica de los atributos de calidad (Reutilización, Abstracción del diseño, Nivel de Cohesión y Cantidad de Pruebas) relacionados con la métrica ND.

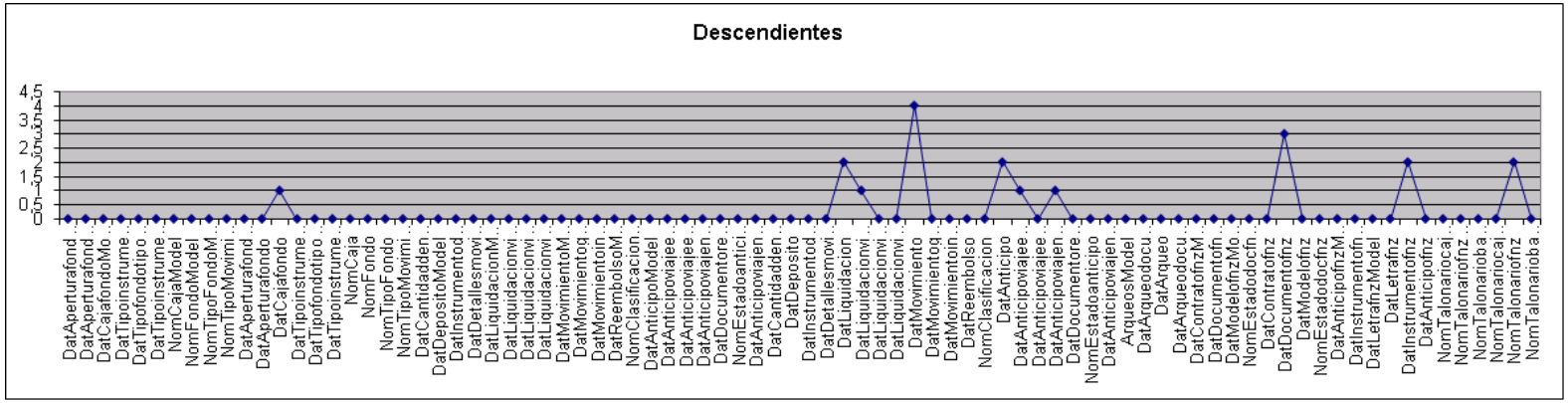
Subsistema	Clase	Clase padre	Número de descendientes	Reutización	Abstracción de la clase base	Cohesión de la JC	Cantidad de Pruebas
Configuraciones	DatAperturafondointrumentoModel		0	Baja	Definida	Alta	Baja
Configuraciones	DatAperturafondoModel		0	Baja	Definida	Alta	Baja
Configuraciones	DatCajafondoModel		0	Baja	Definida	Alta	Baja
Configuraciones	DatTipoinstrumentotipomovimientoModel		0	Baja	Definida	Alta	Baja
Configuraciones	DatTipofondotipomovimientoModel		0	Baja	Definida	Alta	Baja
Configuraciones	DatTipoinstrumentotipofondoModel		0	Baja	Definida	Alta	Baja
Configuraciones	NomCajaModel		0	Baja	Definida	Alta	Baja
Configuraciones	NomFondoModel		0	Baja	Definida	Alta	Baja
Configuraciones	NomTipoFondoModel		0	Baja	Definida	Alta	Baja
Configuraciones	NomTipoMovimientocajaModel		0	Baja	Definida	Alta	Baja
Configuraciones	DatAperturafondointrumento		0	Baja	Definida	Alta	Baja
Configuraciones	DatAperturafondo	DatCajafondo	0	Baja	Definida	Alta	Baja
Configuraciones	DatCajafondo		1	Media	Definida	Alta	Baja
Configuraciones	DatTipoinstrumentotipomovimiento		0	Baja	Definida	Alta	Baja
Configuraciones	DatTipofondotipomovimiento		0	Baja	Definida	Alta	Baja

Configuraciones	DatTipoinstrumentotipofondo		0	Baja	Definida	Alta	Baja
Configuraciones	NomCaja		0	Baja	Definida	Alta	Baja
Configuraciones	NomFondo		0	Baja	Definida	Alta	Baja
Configuraciones	NomTipoFondo		0	Baja	Definida	Alta	Baja
Configuraciones	NomTipoMovimientocaja		0	Baja	Definida	Alta	Baja
Movimientos	DatCantidaddenominacionModel		0	Baja	Definida	Alta	Baja
Movimientos	DatDepositoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatInstrumentodepositoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatDetallesmovimientoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatLiquidacionModel		0	Baja	Definida	Alta	Baja
Movimientos	DatLiquidacionviajeextranjeroModel		0	Baja	Definida	Alta	Baja
Movimientos	DatLiquidacionviajeextranjeropaisModel		0	Baja	Definida	Alta	Baja
Movimientos	DatLiquidacionviajenacionalModel		0	Baja	Definida	Alta	Baja
Movimientos	DatMovimientoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatMovimientogenericoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatMovimientoinstrumentoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatReembolsoModel		0	Baja	Definida	Alta	Baja
Movimientos	NomClasificacionModel		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoviajeextranjeroModel		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoviajeextranjeropaisModel		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoviajenacionalModel		0	Baja	Definida	Alta	Baja
Movimientos	DatDocumentoreembolsoModel		0	Baja	Definida	Alta	Baja
Movimientos	NomEstadoanticipoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoviajenacionalentregadoModel		0	Baja	Definida	Alta	Baja
Movimientos	DatCantidaddenominacion		0	Baja	Definida	Alta	Baja
Movimientos	DatDeposito		0	Baja	Definida	Alta	Baja
Movimientos	DatInstrumentodeposito		0	Baja	Definida	Alta	Baja
Movimientos	DatDetallesmovimiento		0	Baja	Definida	Alta	Baja
Movimientos	DatLiquidacion	DatMovimiento	2	Alta	Definida	Alta	Baja
Movimientos	DatLiquidacionviajeextranjero	DatLiquidacion	1	Media	Definida	Alta	Baja

Movimientos	DatLiquidacionviajeextranjeropais	DatLiquidacion viajeextranjero	0	Baja	Definida	Alta	Baja
Movimientos	DatLiquidacionviajenacional	DatLiquidacion	0	Baja	Definida	Alta	Baja
Movimientos	DatMovimiento		4	Alta	Afectada	Media	Media
Movimientos	DatMovimientogenerico	DatMovimiento	0	Baja	Definida	Alta	Baja
Movimientos	DatMovimientoinstrumento		0	Baja	Definida	Alta	Baja
Movimientos	DatReembolso	DatMovimiento	0	Baja	Definida	Alta	Baja
Movimientos	NomClasificacion		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipo	DatMovimiento	2	Alta	Definida	Alta	Baja
Movimientos	DatAnticipoviajeextranjero	DatAnticipo	1	Media	Definida	Alta	Baja
Movimientos	DatAnticipoviajeextranjeropais	DatAnticipoviaje extranjero	0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoviajenacional	DatAnticipo	1	Media	Definida	Alta	Baja
Movimientos	DatDocumentoreembolso		0	Baja	Definida	Alta	Baja
Movimientos	NomEstadoanticipo		0	Baja	Definida	Alta	Baja
Movimientos	DatAnticipoviajenacionalentregado	DatAnticipoviaje nacional	0	Baja	Definida	Alta	Baja
Arqueos	ArqueosModel		0	Baja	Definida	Alta	Baja
Arqueos	DatArqueodocumentoModel		0	Baja	Definida	Alta	Baja
Arqueos	DatArqueo		0	Baja	Definida	Alta	Baja
Arqueos	DatArqueodocumento		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatContratofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatDocumentofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatModelofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	NomEstadodocfnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatContratofnz	DatDocumentofnz	0	Baja	Definida	Alta	Baja
Comunfinanzas	DatDocumentofnz		3	Alta	Afectada	Media	Media
Comunfinanzas	DatModelofnz	DatDocumentofnz	0	Baja	Definida	Alta	Baja
Comunfinanzas	NomEstadodocfnz		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatAnticipofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatInstrumentofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatLetrafnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	DatLetrafnz	DatInstrumentofnz	0	Baja	Definida	Alta	Baja

Comunfinanzas	DatInstrumentofnz	DatDocumentofnz	2	Alta	Definida	Alta	Baja
Comunfinanzas	DatAnticipofnz	DatInstrumentofnz	0	Baja	Definida	Alta	Baja
Comunfinanzas	NomTalonariocajafnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	NomTalonariofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	NomTalonariobancofnzModel		0	Baja	Definida	Alta	Baja
Comunfinanzas	NomTalonariocajafnz	NomTalonariofnz	0	Baja	Definida	Alta	Baja
Comunfinanzas	NomTalonariofnz		2	Alta	Definida	Alta	Baja
Comunfinanzas	NomTalonariobancofnz	NomTalonariofnz	0	Baja	Definida	Alta	Baja

Resultados de la evaluación de la métrica ND y su influencia en los atributos de calidad (Reutilización, Abstracción del diseño, Nivel de Cohesión y Cantidad de Pruebas).



Representación de los resultados obtenidos al aplicar los instrumentos que evalúan la métrica ND.

Anexo 18 Instrumento de medición de la métrica Número de Operaciones Redefinidas (NOR).

	Categoria	Criterio
Complejidad de Mantenimiento	Baja	0
	Media	Entre 0 y 2

	Categoria	Criterio
Cantidad de Pruebas	Baja	0
	Media	Entre 0 y 2

	Categoria	Criterio
Violación de la Abstracción	Baja	0
representada por la superclase	Media	Entre 0 y 2

Rango de valores de para la evaluación técnica de los atributos de calidad (Abstracción del diseño, Cantidad de pruebas, Complejidad de Mantenimiento) relacionados con la métrica NOR.

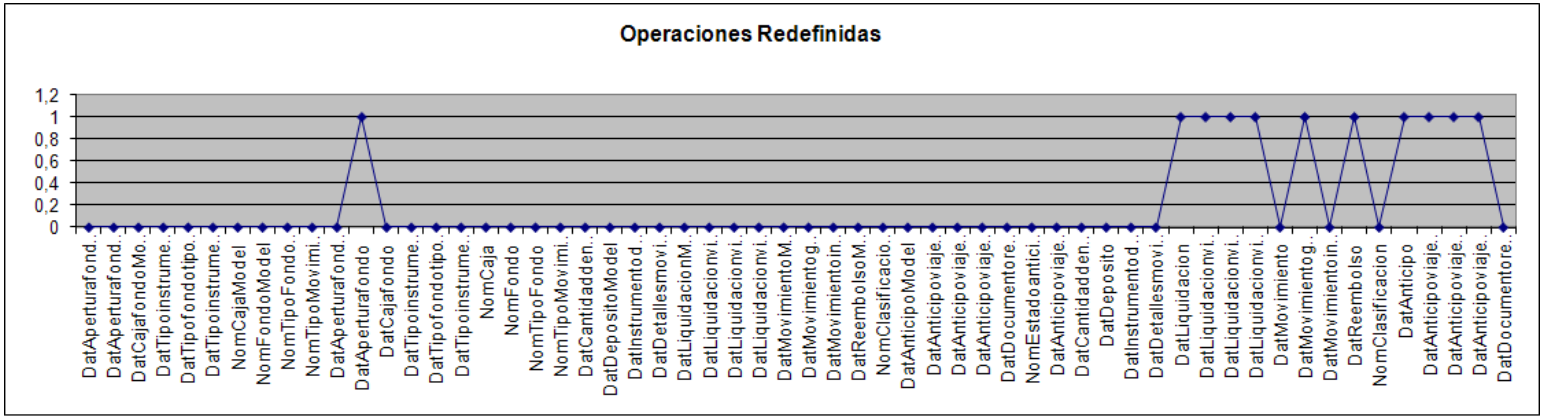
No	Subsistema	Clase	Clase padre	Numero de Operaciones redefinidas	Calidad del Diseño	Complejidad de Mant.	Cantidad de Pruebas	Violación de ARS
1	Configuraciones	DatAperturafondointrumentoModel		0	Buena	Baja	Baja	Baja
2	Configuraciones	DatAperturafondoModel		0	Buena	Baja	Baja	Baja
3	Configuraciones	DatCajafondoModel		0	Buena	Baja	Baja	Baja
4	Configuraciones	DatTipoinstrumentotipomovimientoModel		0	Buena	Baja	Baja	Baja
5	Configuraciones	DatTipofondotipomovimientoModel		0	Buena	Baja	Baja	Baja
6	Configuraciones	DatTipoinstrumentotipofondoModel		0	Buena	Baja	Baja	Baja
7	Configuraciones	NomCajaModel		0	Buena	Baja	Baja	Baja
8	Configuraciones	NomFondoModel		0	Buena	Baja	Baja	Baja
9	Configuraciones	NomTipoFondoModel		0	Buena	Baja	Baja	Baja
10	Configuraciones	NomTipoMovimientocajaModel		0	Buena	Baja	Baja	Baja
11	Configuraciones	DatAperturafondointrumento		0	Buena	Baja	Baja	Baja
12	Configuraciones	DatAperturafondo	DatCajafondo	1	Aceptable	Media	Media	Media

13	Configuraciones	DatCajafondo		0	Buena	Baja	Baja	Baja
14	Configuraciones	DatTipoinstrumentotipomovimiento		0	Buena	Baja	Baja	Baja
15	Configuraciones	DatTipofondotipomovimiento		0	Buena	Baja	Baja	Baja
16	Configuraciones	DatTipoinstrumentotipofondo		0	Buena	Baja	Baja	Baja
17	Configuraciones	NomCaja		0	Buena	Baja	Baja	Baja
18	Configuraciones	NomFondo		0	Buena	Baja	Baja	Baja
19	Configuraciones	NomTipoFondo		0	Buena	Baja	Baja	Baja
20	Configuraciones	NomTipoMovimientocaja		0	Buena	Baja	Baja	Baja
21	Movimientos	DatCantidaddenominacionModel		0	Buena	Baja	Baja	Baja
22	Movimientos	DatDepositoModel		0	Buena	Baja	Baja	Baja
23	Movimientos	DatInstrumentodepositoModel		0	Buena	Baja	Baja	Baja
24	Movimientos	DatDetallesmovimientoModel		0	Buena	Baja	Baja	Baja
25	Movimientos	DatLiquidacionModel		0	Buena	Baja	Baja	Baja
26	Movimientos	DatLiquidacionviajeextranjeroModel		0	Buena	Baja	Baja	Baja
27	Movimientos	DatLiquidacionviajeextranjeropaisModel		0	Buena	Baja	Baja	Baja
28	Movimientos	DatLiquidacionviajenacionalModel		0	Buena	Baja	Baja	Baja
29	Movimientos	DatMovimientoModel		0	Buena	Baja	Baja	Baja
30	Movimientos	DatMovimientogenericoModel		0	Buena	Baja	Baja	Baja
31	Movimientos	DatMovimientoinstrumentoModel		0	Buena	Baja	Baja	Baja
32	Movimientos	DatReembolsoModel		0	Buena	Baja	Baja	Baja
33	Movimientos	NomClasificacionModel		0	Buena	Baja	Baja	Baja
34	Movimientos	DatAnticipoModel		0	Buena	Baja	Baja	Baja
35	Movimientos	DatAnticipoviajeextranjeroModel		0	Buena	Baja	Baja	Baja
36	Movimientos	DatAnticipoviajeextranjeropaisModel		0	Buena	Baja	Baja	Baja
37	Movimientos	DatAnticipoviajenacionalModel		0	Buena	Baja	Baja	Baja
38	Movimientos	DatDocumentoreembolsoModel		0	Buena	Baja	Baja	Baja
39	Movimientos	NomEstadoanticipoModel		0	Buena	Baja	Baja	Baja
40	Movimientos	DatAnticipoviajenacionalentregadoModel		0	Buena	Baja	Baja	Baja
41	Movimientos	DatCantidaddenominacion		0	Buena	Baja	Baja	Baja
42	Movimientos	DatDeposito		0	Buena	Baja	Baja	Baja
43	Movimientos	DatInstrumentodeposito		0	Buena	Baja	Baja	Baja

44	Movimientos	DatDetallesmovimiento		0	Buena	Baja	Baja	Baja
45	Movimientos	DatLiquidacion	DatMovimiento	1	Aceptable	Media	Media	Media
46	Movimientos	DatLiquidacionviajeextranjero	DatLiquidacion	1	Aceptable	Media	Media	Media
47	Movimientos	DatLiquidacionviajeextranjeropais	DatLiquidacion viajeextranjero	1	Aceptable	Media	Media	Media
48	Movimientos	DatLiquidacionviajenacional	DatLiquidacion	1	Aceptable	Media	Media	Media
49	Movimientos	DatMovimiento		0	Buena	Baja	Baja	Baja
50	Movimientos	DatMovimientogenerico	DatMovimiento	1	Aceptable	Media	Media	Media
51	Movimientos	DatMovimientoinstrumento		0	Buena	Baja	Baja	Baja
52	Movimientos	DatReembolso	DatMovimiento	1	Aceptable	Media	Media	Media
53	Movimientos	NomClasificacion		0	Buena	Baja	Baja	Baja
54	Movimientos	DatAnticipo	DatMovimiento	1	Aceptable	Media	Media	Media
55	Movimientos	DatAnticipoviajeextranjero	DatAnticipo	1	Aceptable	Media	Media	Media
56	Movimientos	DatAnticipoviajeextranjeropais	DatAnticipo viajeextranjero	1	Aceptable	Media	Media	Media
57	Movimientos	DatAnticipoviajenacional	DatAnticipo	1	Aceptable	Media	Media	Media
58	Movimientos	DatDocumentoreembolso		0	Buena	Baja	Baja	Baja
59	Movimientos	NomEstadoanticipo		0	Buena	Baja	Baja	Baja
60	Movimientos	DatAnticipoviajenacionalentregado	DatAnticipo viajenacional	1	Aceptable	Media	Media	Media
61	Arqueos	ArqueosModel		0	Buena	Baja	Baja	Baja
62	Arqueos	DatArqueodocumentoModel		0	Buena	Baja	Baja	Baja
63	Arqueos	DatArqueo		0	Buena	Baja	Baja	Baja
64	Arqueos	DatArqueodocumento		0	Buena	Baja	Baja	Baja
65	Comunfinanzas	DatContratofnzModel		0	Buena	Baja	Baja	Baja
66	Comunfinanzas	DatDocumentofnzModel		0	Buena	Baja	Baja	Baja
67	Comunfinanzas	DatModelofnzModel		0	Buena	Baja	Baja	Baja
68	Comunfinanzas	NomEstadodocfnzModel		0	Buena	Baja	Baja	Baja
69	Comunfinanzas	DatContratofnz	DatDocumentofnz	1	Aceptable	Media	Media	Media
70	Comunfinanzas	DatDocumentofnz		0	Buena	Baja	Baja	Baja
71	Comunfinanzas	DatModelofnz	DatDocumentofnz	1	Aceptable	Media	Media	Media
72	Comunfinanzas	NomEstadodocfnz		0	Buena	Baja	Baja	Baja

73	Comunfinanzas	DatAnticipofnzModel		0	Buena	Baja	Baja	Baja
74	Comunfinanzas	DatInstrumentofnzModel		0	Buena	Baja	Baja	Baja
75	Comunfinanzas	DatLetrafnzModel		0	Buena	Baja	Baja	Baja
76	Comunfinanzas	DatLetrafnz	DatInstrumentofnz	1	Aceptable	Media	Media	Media
77	Comunfinanzas	DatInstrumentofnz	DatDocumentofnz	1	Aceptable	Media	Media	Media
78	Comunfinanzas	DatAnticipofnz	DatInstrumentofnz	1	Aceptable	Media	Media	Media
79	Comunfinanzas	NomTalonariocajafnzModel		0	Buena	Baja	Baja	Baja
80	Comunfinanzas	NomTalonariofnzModel		0	Buena	Baja	Baja	Baja
81	Comunfinanzas	NomTalonariobancofnzModel		0	Buena	Baja	Baja	Baja
82	Comunfinanzas	NomTalonariocajafnz	NomTalonariofnz	1	Aceptable	Media	Media	Media
83	Comunfinanzas	NomTalonariofnz		0	Buena	Baja	Baja	Baja
84	Comunfinanzas	NomTalonariobancofnz	NomTalonariofnz	1	Aceptable	Media	Media	Media

Resultados de la evaluación de la métrica NOR y su influencia en los atributos de calidad (Abstracción del diseño, Cantidad de pruebas, Complejidad de Mantenimiento).



Representación de los resultados obtenidos al aplicar los instrumentos que evalúan la métrica NOR.

GLOSARIO DE TÉRMINOS

1. **ERP:** Sistemas de planificación de recursos empresariales (Enterprise Resource Planning, ERP por sus siglas en inglés).
2. **SSL:** Protocolo de comunicación de datos desarrollado por Netscape para transmitir documentos privados a través del Internet.
3. **ECLIPSE:** Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores.
4. **NCSA HTTPd:** Servidor web desarrollado originalmente en el National Center for Supercomputing Applications por Robert McCool y una lista de colaboradores.
5. **IP:** Número que identifica a cada dispositivo dentro de una red con protocolo de internet.
6. **C:** lenguaje de programación creado en 1972 por Ken Thompson y Dennis M. Ritchie en los Laboratorios Bell.
7. **C++:** Es un lenguaje C avanzado, basado en objetos.
8. **JAVA:** Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.
9. **HTTPS:** Protocolo de Web para intercambio de información segura.
10. **JSON**: acrónimo de "JavaScript Object Notation", es un formato ligero para el intercambio de datos.
11. **XHTML:** Reformulación del lenguaje HTML utilizado para crear paginas Web.
12. **CCS:** es un lenguaje de especificación.
13. **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.
14. **XSLT:** Es un estándar de la organización W3C que presenta una forma de transformar documentos XML en otros.
15. **XMLHttpRequest:** es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB.

16. **PDF:** es un formato de almacenamiento de documentos.
17. **ASCII:**
18. Código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales.
19. **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.
20. **DBAL:** Es una capa de abstracción de bases de datos, es decir, una interfaz de programación de aplicaciones que unifica la comunicación entre una aplicación informática y bases de datos tales como PostgreSQL