Universidad de las Ciencias Informáticas

Facultad 15

Implementación de los Componentes Depreciación y Submayor del Subsistema Activo Fijo Tangible del Sistema integral de Gestión Cedrux.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Elier Cruz Hernández.

Tutor: Tte. Ing. Pedro Julio Rodríguez Paredes.

Ing. Damián Viltres Ramírez.

Ciudad de la Habana. Junio de 2009

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Departamento de Tecnología del Centro de Informatización de entidades de la Facultad 15 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Elier Cruz Hernández

_____

Firma del Autor

Tte. Ing. Pedro Julio Rodríguez Paredes

_____

Firmar del Tutor

Ing. Damián Viltres Ramírez

_____

Firma del Tutor

Pensamiento

“El revolucionario verdadero está guiado por grandes sentimientos de amor”

ERNESTO CHE GUEVARA

Datos de contacto

Tutor: Pedro Julio Rodríguez Paredes.

Categoría Científica: Ingeniero.

Categoría Docente:

Correo electrónico: pjrodriguez@uci.cu

Síntesis del Tutor:

Co-Tutor: Damián Viltres Ramírez.

Categoría Científica: Ingeniero.

Categoría Docente:

Correo electrónico: dviltres@uci.cu

Síntesis del Tutor:

Agradecimientos

A mis padres Zaida y Elier por todo el amor, apoyo, comprensión y confianza que me han brindado, por ser los verdaderos artífices de esta obra, por ser mi fuerza, mi inspiración, mi orgullo, por ser todo lo que necesito y necesitaré siempre. Gracias por hacer de mi lo que hoy soy.

A mis abuelos Beatriz y Evaristo por depositar en mí toda la confianza y el amor que un ser humano puede llevar consigo.

A mi hermanita del alma Edalis por todo el amor que me has dado en la vida, gracias por ese apoyo incondicional que siempre me has brindado.

A mis tíos Vary, Kiki, Sara, Oneida y Enrique por todo el cariño, amor y apoyo que me brindaron en los momentos más importantes de mi vida.

A mis primos Beatriz, Susana, José Gabriel, Enrique Alejandro y Ernestico, espero que sigan mis pasos y puedan lograr sus sueños, recuerden que todo en la vida es posible.

A mis amigos Eduardo, Maikel, Riki, Yudenia, Bato, Jorgito, Vicky, Alfredo, Manuel, Street y Leandro, por mencionar unos pocos, a quienes siempre voy a respetar, recordar y querer, por ayudarme y acompañarme en los momentos más difíciles, por hacerme encontrar la dicha en cada momento que paso con ellos.

A mis tutores Pedro y Damián, por todo el trabajo que realizaron y la ayuda que me brindaron.

A mis compañeros de proyecto, Damaris, Osnier, Carlos, Obel y a todos los demás que siempre me brindaron su ayuda cuando lo necesité.

A la revolución y a la Universidad de las Ciencias Informáticas donde me forjé como profesional.

A todos los que de una forma u otra influyeron en mi formación como ser humano y profesional.

A todos ustedes, muchas gracias.

Dedicatoria

A mi mamita linda por brindarme su confianza, cariño de madre, por ser amiga, compañera y consejera indisoluble de mi vida.

A mi papá por haberme sabido guiar y trasmitirme su experiencia a lo largo de estos años.

A mis abuelos Beatriz y Evaristo, espero que se sientan orgullosos de mí.

A mi hermana, a quien quiero más cada día que pasa.

A mi familia por ser el sustento fundamental de mi vida, por construirme el rinconcito más feliz y cálido del mundo.

Resumen

En el presente trabajo de diploma se expone la implementación de los Componentes Depreciación y Submayor del Subsistema Activo Fijo Tangible (AFT) del Sistema Integral de Gestión Cedrux. Actualmente la situación general de los procesos de gestión de las entidades presupuestadas y empresariales a escala nacional está afectada por la existencia de sistemas informáticos que no cumplen con las expectativas de las nuevas tecnologías y la misión de nuestro país en el desarrollo de sus empresas. Proteger los recursos de las empresas así como llegar a la exactitud y confiabilidad de la información económica y contable, ha llevado a la necesidad de mejorar los procesos de gestión de las áreas que las conforman, utilizando plataformas confiables y eficientes. Con el desarrollo de estos componentes se logró optimizar el proceso de gestión de los AFT, disminuir los costos totales de operación, compartir información entre todos los componentes de la empresa y disminuir el margen de contaminación y repetición de la información. Una vez finalizados los componentes se ratifica la solución propuesta a partir de la utilización de métricas para validar el diseño y pruebas de caja blanca para verificar que el código funcione de manera correcta.

PALABRAS CLAVES

Depreciación, Submayor.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Cuba, aún siendo una nación pequeña que por más de cinco décadas ha sido bloqueada por el Gobierno de los Estados Unidos ha tomado de la mano los avances tecnológicos de estos tiempos. El país se encuentra inmerso en un proceso de informatización de la sociedad que permitirá, en sectores tan importantes como la salud, la educación, el comercio e incluso la defensa, aumentar el nivel de vida de los cubanos.

En vista de las exigencias de la industria del software de hoy en día, ha sido necesario y de suma importancia hacer software de una buena calidad para poder entrar en el mercado. Las Tecnologías de la Información y las Comunicaciones (TIC), producto de sus avances, han posibilitado el incremento rápido del desarrollo de la economía y la sociedad. Estos avances vienen dados gracias a la ambición de conocimiento del ser humano, a su naturaleza creadora, y son aplicados en gran medida en el desarrollo de la economía y la sociedad mediante la producción de software. Esta actividad demanda cada vez más un mayor estudio y una mejor planificación porque cada vez se solicitan programas más complejos y con mayor calidad. Nuestra nación, guiada por el uso y desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) invierte grandes recursos y esfuerzos en el proceso de informatización de la sociedad. La Universidad de la Ciencias Informáticas juega un papel de gran importancia en esta rama de la economía, por tal motivo en la universidad existe una gran cantidad de proyectos productivos de diferentes temas para poder lograr un óptimo nivel en la calidad de cada software que se produzca.

Actualmente en nuestro país se trabaja en la creación del producto Cedrux, un sistema integral para la gestión de recursos empresariales basado en los principios de independencia tecnológica y en las particularidades de la economía cubana. El mismo está diseñado para modelar y automatizar la mayoría de los procesos en las empresas así como facilitar la planificación de todos los recursos de estas instituciones. Esta solución nacional cuenta con un subsistema para el control de los activos fijos tangibles (AFT) mediante el cual se gestionan los activos de forma dinámica para estandarizar el tratamiento de estos a nivel nacional, que además no cuenta con un proceso que brinde la posibilidad de visualizar todas las operaciones que se han realizado con un activo. Los AFT son los activos palpables de larga duración, el término denota sustancia física donde no se incluyen los recursos naturales. De los activos que se conocen se pueden mencionar: edificios, equipos de oficina, mobiliario, maquinarias y terrenos. A los activos se les aplica un proceso de depreciación que no es más que la asignación del costo de un activo

fijo tangible al gasto de dicho activo en el período en que este brinda sus servicios. Entre los problemas que se han logrado detectar en las empresas cubanas con relación al activo fijo es que se continúa depreciando activos que ya están totalmente depreciados debido a que este proceso no está automatizado en dichas empresas.

Otro de los procesos que se llevan a cabo en la gestión de los activos fijos es el submayor del activo que no es más que el control que se lleva sobre este en el tiempo que se utilizó. En este subsistema se guardan las operaciones que se realizaron que pueden ser de altas, de bajas, de reparación y de traslado, donde este último está compuesto por los traslados internos y por los traslados de ocioso a activo y de activo a ocioso. El componente submayor es la recuperación más importante en cuanto a información del activo fijo tangible donde se tiene la vista de todos los activos y las operaciones que se han realizado con ellos. Actualmente en varias empresas cubanas estos procesos se realizan manualmente haciéndolo más lento y engorroso por lo que se hace necesario automatizar estos procesos para un mejor rendimiento de las empresas. De forma general se puede decir que al no existir un modelo estándar de movimiento de los activos no es posible agrupar los datos obligatorios. Por consiguiente la realización de operaciones sobre los activos sin dejar trazabilidad de los mismos provoca que el contador no tenga conocimiento de origen y destino o de la depreciación que se le ha realizado a los activos que controla.

Teniendo en cuenta lo antes expuesto se define el siguiente **problema:**

¿Qué implementación realizar a los componentes Depreciación y Submayor de forma que se garantice una solución integrada para la gestión de los AFT en una entidad?

Para darle solución al problema planteado se define como **Objetivo General:**

Implementar los componentes Depreciación y Submayor del subsistema Activo Fijo Tangible (AFT) del Sistema Integral de Gestión Cedrux de forma que se garantice una solución integrada para la gestión de los AFT en una entidad.

Objetivos Específicos:

- Analizar los procesos de Depreciación y Submayor y los sistemas que existen actualmente para su gestión así como las herramientas que se utilizarán para el desarrollo de la solución.
- Implementar los componentes Depreciación y Submayor del Subsistema AFT.

- Validar el resultado obtenido.

Objeto de estudio: Desarrollo de los sistemas para la gestión de AFT.

Campo de acción: Implementación de los procesos de Depreciación y Submayor.

Idea a Defender:

Si se realiza la implementación de los componentes Depreciación y Submayor, pertenecientes al subsistema Activo Fijo Tangible (AFT) del Sistema Integral de Gestión Cedrux, se garantizará una solución integrada para la gestión de los AFT en una entidad.

Tareas de investigación:

- Investigar el estado del arte de los procesos Depreciación y Submayor en las entidades que gestionan AFT a nivel nacional e internacional.
- Estudiar las tecnologías, los lenguajes y las herramientas propuestas para el desarrollo de la aplicación.
- Analizar los artefactos entregados por los analistas.
- Implementar las interfaces a partir del prototipo entregado por los analistas.
- Implementar la capa de negocio que dé respuesta a los requisitos propuestos por los analistas.
- Implementar la capa de acceso a datos.
- Implementar las validaciones y excepciones.
- Implementar los servicios incluidos dentro de sus responsabilidades que se necesiten para la implementación de otros módulos y componentes.
- Validar la implementación de los componentes obtenidos a través de métricas.
- Realizar pruebas de unidad a los componentes obtenidos.

El **aporte práctico** esperado del trabajo es: La obtención de los componentes Depreciación y Submayor pertenecientes al Subsistema AFT.

El documento se divide en 3 capítulos:

Capítulo 1: Fundamentación Teórica.

Se expone el estado del arte, mostrando los sistemas vinculados al campo de acción. Al mismo tiempo se describe el objeto de estudio. También se detallan tendencias y tecnologías actuales utilizadas para el desarrollo de los componentes y el motivo de su utilización. Se realiza una caracterización del modelo de desarrollo aplicado, así como de las herramientas utilizadas para desarrollar los artefactos.

Capítulo 2: Descripción de la solución propuesta.

Se especifica la propuesta del sistema describiéndose las principales clases. Se exponen los diagramas de clases, de componentes internos y de integración entre componentes que brindan una vista del funcionamiento del sistema. Se realizó el cálculo de la complejidad ciclomática de algoritmos, determinándose así el grafo de flujo asociado a un fragmento de código y se muestran las tres vías de calcular la complejidad ciclomática.

Capítulo 3: Validación de la solución propuesta.

Se valida la solución propuesta a través de la realización de pruebas. Este capítulo se encuentra dividido en dos partes: la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y de las pruebas de caja negra.

Capítulo 1: Fundamentación Teórica

1.1 Introducción.

En estos tiempos el ambiente competitivo del mundo empresarial se ha incrementado indudablemente, debido a que las entidades requieren promover procesos y entidades de negocio que brinden ventajas ante la competencia o que simplemente proporcionen ahorro y eficiencia económica. Hoy más que nunca se hace necesaria la utilización de herramientas que proporcionen un mejor control y centralización de la información con el fin de tomar las mejores decisiones para los procesos y estrategias del negocio. En Cuba y el resto del mundo se han desarrollado aplicaciones relacionados con la Gestión de los Activos Fijos, cuyo estudio ha servido para tener en cuenta las ventajas y defectos que tienen. A menudo surgen dudas sobre cuál usar, debido a que actualmente aparecen nuevas herramientas con mejorías, que se van imponiendo para el desarrollo de software. Es por esto que surge la necesidad de hacer un estudio de los sistemas existentes y las herramientas a utilizar, buscando obtener la información más actual de las mismas.

1.2 Conceptos generales.

1.2.1 Depreciación.

La depreciación es la asignación del costo de un AFT al gasto en los períodos en los cuales se reciben los servicios del activo. La finalidad de la depreciación es aplicar el principio de asociación, es decir, compensar el ingreso de un período contable con los costos de los bienes y servicios que son consumidos en el esfuerzo de generar ingresos. Esta va a ser originada por dos causas principales:

- El **deterioro físico**: resulta del uso, lo mismo que de la exposición al sol, al viento y a otros factores climáticos. Cuando un activo fijo ha sido conservado cuidadosamente, no es extraño que el propietario afirme que el activo está “como nuevo”. Tales afirmaciones no son literalmente ciertas. Aunque una buena política de reparación puede alargar mucho la vida útil de una máquina, ello no quiere decir que se elimine la necesidad de reconocer su depreciación.
- **Obsolescencia o deterioro moral**: es el proceso de quedar desactualizado, debido a la introducción de nuevas modificaciones o técnicas más avanzadas, que superan el desempeño anterior. Por ejemplo un avión puede quedar obsoleto, debido a la disponibilidad de mejores

diseños que implican mayor calidad, costos más bajos y menor contaminación. (Del Toro Ríos, 2008)

1.2.2 Submayor.

Es un registro donde se reflejan todos los movimientos que tienen los AFT, en él se guardan todas las operaciones que se han realizado, estas operaciones se clasifican en: (González Alvarez y otros, 2008)

- **Altas:** Es la adquisición de AFT nuevos, de uso, construidos con medios propios y contratados con terceros. Incluye además la conclusión del proceso inversionista, así como los AFT que fueron recibidos en condición de Donaciones.
- **Bajas:** Incluye a los AFT que habiendo terminado o no su vida útil programada, el informe técnico determina que no tiene valor de uso. Incluye además la entrega de AFT que se ceden en condición de Donación, así como la baja por obsolescencia tecnológica.
- **Traslado:** Incluye los traslados de activo a ocioso que es cuando un activo ha dejado de usarse para ser guardado y de ocioso a activo, cuando se utiliza un activo que estaba sin usar.

1.3 Situación actual de los procesos.

En la actualidad no existe en Cuba un sistema informático integral de gestión que cumpla con la totalidad de los requerimientos de funcionalidad, interoperabilidad y seguridad que espera el gobierno cubano de una solución de este tipo, de manera que pueda ser utilizada como herramienta para potenciar el cumplimiento de las funciones de las entidades a todos los niveles con un máximo de racionalidad y control de los recursos financieros, materiales y humanos. (Del Toro Ríos, 2009)

1.4 Sistemas automatizados existentes.

1.4.1 Sistemas nacionales.

1.4.1.1 Versat-Sarasola.

Además de ser un producto cubano es el primer Sistema Integral de gestión de contabilidad certificado, desarrollado para la gestión económica eficaz y fiable. Actualmente es utilizado en Cuba en alrededor de 200 entidades de varias provincias y en lo adelante será introducido en más de dos mil 500 unidades presupuestadas. Este sistema integrado cuenta con un conjunto de 12 módulos entre los que se encuentra el control de los activos fijos. Dicho módulo incluye entre otros: **Vista de documentos:** La vista muestra los documentos del período o los que estén en el rango de fechas seleccionado. La gestión de

documentos abarca los estados y tipos de documentos (predefinidos). **Vista de activos:** Garantiza la búsqueda de activos y la modificación de propiedades. **Codificadores:** En esta vista se crean, capturan y actualizan todos los codificadores del Sistema: (betsime.disaic.cu, 2007)

- Estados de los activos.
- Grupos de activos.
- Codificador de Activos fijos.
- Conceptos de documentos.
- Consecutivos.

En el módulo de **Control de Activos Fijos** se recogen las operaciones normales que en esta actividad se realizan que pueden ser de altas, bajas y las modificaciones de los medios. También cuenta con otras opciones novedosas, dentro de las cuales se encuentran la posibilidad de amortizar los activos por los 4 métodos más conocidos internacionalmente. Posee un asistente (wizard) para la configuración del subsistema, la tarea más compleja de este módulo.

Características.

- Es una aplicación de escritorio.
- Implementado en Delphi.
- Trabaja sobre el sistema operativo Windows.
- Soporte para base de datos SQL Server 2000.
- Diseñado para su empleo en cualquier tipo de entidad empresarial o presupuestada.
- Se logra establecer un proceso de interacción usuario-sistema.

1.4.1.2 Rodas XXI.

Es un sistema multiempresa y multiusuario creado por CITMATEL para la automatización de la gestión empresarial. Contiene diferentes módulos que pueden usarse integrados o independientes. Además de contar con el módulo Administrador, que brinda mayor integralidad al sistema y garantiza facilidades adicionales durante su instalación y explotación cuenta también con los siguientes módulos: (RodasXXI, 2010)

- Contabilidad.
- Efectivo, caja y banco.
- Nóminas.
- **Activos Fijos Tangibles.**
- Facturación.
- Finanzas.

El módulo de activos fijos permite tener un control detallado de los activos fijos de su entidad, realizando en el mismo momento que se registra un movimiento, su contabilización. Permite el control por separado de los activos fijos que se encuentran en almacén de los que se encuentran en explotación, además permite obtener el submayor de los activos fijos, listados y localización de medios de transporte de la entidad y la depreciación mensual acumulada de uno o de los activos fijos que desee.

1.4.2 Sistemas internacionales.

1.4.2.1 Condor.

- Es un sistema automatizado de alta complejidad y seguridad que abarca todos los aspectos del proceso contable de una entidad, este brinda mayor autonomía al cliente para efectuar cambios de estructura sin necesidad de la intervención de especialistas y está formado por varios módulos: (Condor Light, 2007)
- **Activos fijos.**
- Contabilidad general.
- Nóminas.
- Control de inventario.
- Recursos humanos.

Entre sus módulos se encuentra la Gestión de Activos Fijos la cual permite entre otras funcionalidades:

- Distintas clases de bienes de uso.
- Proceso de amortización automático.
- Generación de transacción de amortización.
- Ubicación y asignación de los bienes.

1.4.2.2 SAP (Sistemas, aplicaciones y procesos)

Este software se desarrolló en Alemania, por antiguos empleados de la IBM. Esta corporación se ha desarrollado hasta convertirse en la quinta más grande compañía mundial de software, cuenta con el módulo Controlling (CO), que permite llevar el control de los gastos generales, costes de producto, cuenta de resultados y centro de beneficios.

La gestión de Activos fijos en este sistema ERP[1] se encuentra dentro del área funcional que integra el módulo de Contabilidad Financiera, donde su objetivo principal es asegurar que exista un reflejo real de los movimientos y operaciones que se han realizado con las activos fijos. Entre las características principales de SAP se encuentran:

- Implementado en .NET y WebSphere.
- Ofrece una nueva plataforma tecnológica denominada SAP NetWeaver, esta plataforma tecnológica convierte a SAP en un programa Web-enabled, lo que significa que estaría totalmente preparado para trabajar con él mediante la web.
- Trabaja sobre el sistema operativo Windows.
- Soporte para bases de datos Oracle. (SAP Professionals, 2007)

1.5 Valoración crítica.

A partir del estudio de los sistemas de gestión de entidades mencionados anteriormente, que dentro de sus funcionalidades tienen incluido el control de AFT, se concluyó de que todos tienen como desventaja principal que están basados en plataformas de software propietario, aspecto negativo para el país ya que este se encuentra inmerso en un proceso de independencia tecnológica por lo que representarían gastos innecesarios por concepto de licencia y mantenimiento, son aplicaciones de escritorio por lo que el cliente debe instalar la aplicación en cada estación de trabajo, en cambio, con una aplicación web solo bastaría tener acceso a la red para acceder al sistema. Se analizaron además aspectos positivos de estos sistemas que sirvieron como ejemplo para el desarrollo de los componentes Depreciación y Submayor, específicamente a la hora de determinar qué implementación realizarle a dichos componentes. Un ejemplo de esto fueron los sistemas Versat-Sarasola, SAP y Rodas XXI, cuyos módulos de gestión de AFT

[1] Enterprise Resource Planning

permiten agrupar las operaciones que se realizan con los activos pero tienen como desventaja que no brindan la posibilidad de realizar la depreciación por área de responsabilidad. Por su parte el sistema Condor posee un módulo que abarca la gestión de los activos pero en él no se registran los movimientos sufridos por estos y tampoco permite realizar la depreciación por área de responsabilidad.

1.6 Modelo de desarrollo de software.

Para el desarrollo de la aplicación se utilizó un modelo de desarrollo orientado a componentes, definido por la subdirección de Arquitectura del Centro de Soluciones de Gestión. Este modelo cuenta con los Roles mostrados en la Figura

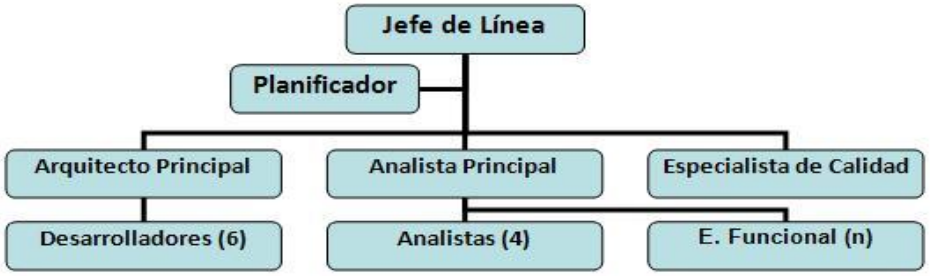

Figura 1 Estructuras de las líneas de desarrollo.

***Flujos de Trabajo/Artefactos*:**

Inicio

- Plan de Iteración (Jefe de la Línea y Planificador)
- Plan de Trabajo Individual (Lo elaboran Todos) (Lo controla el planificador y lo evalúa el Jefe de la Línea)

Análisis

- Arquitectura de Negocio
  - Mapa de Proceso (Analista Principal)
  - Descripción de Procesos de Negocio (Analista)
- Especificación de Requisitos (Analista)

- Casos de Prueba (Especialistas de Calidad, Analistas y Especialistas Funcionales)
- Arquitectura del Sistema (Arquitecto de Sistema)

Diseño e Implementación

- Modelo Conceptual (Arquitecto de Sistema) (Desarrolladores, Analistas)
- Modelo de Datos (Desarrollador)
- Diseño de Clases (Desarrollador)
- Diseño de Interfaz de Usuario (Desarrollador)
- Release (Arquitecto de Sistema)

Prueba

- Casos de Prueba (Equipo de Pruebas Central)
- Registro de no Conformidades (Especialista de Calidad responsable) (Equipo de Pruebas Central registra las No Conformidades)

Estabilización e Integración

- Historia de la Iteración (Jefe de la Línea)
- Informe de Integración (Arquitecto de Sistema)

Mantenimiento

- Peticiones de Actualizaciones (En teoría es el departamento de Implantación).
- Registro de errores (En teoría es el departamento de Implantación).

1.7 Herramientas, lenguajes y tecnologías.

1.7.1 Herramientas de desarrollo de software.

Las HDS[2] automatizan metodologías de software y desarrollo de sistemas. El soporte que brindan al proceso de desarrollo proporciona importantes ventajas para el equipo de trabajo las cuales se sintetizan en:

- Apoyar a las metodologías y métodos, integrando actividades y propiciando visión de continuidad entre fases metodológicas.
- Mejorar la comunicación entre los actores involucrados, facilitándoles compartir su trabajo y desempeñarlo de forma dinámica e iterativa.
- Establecer métodos efectivos para almacenar y utilizar los datos, lo que permite organizar y correlacionar componentes, para acceder a estos a través de un repositorio.
- Agregar eficiencia al mantenimiento, ya que los programas son construidos sobre las mismas estructuras y estándares, facilitando la adherencia a la disciplina de diseño y facilitan también la conversión automática de programas a versiones más recientes de lenguajes de programación.
- Automatizar porciones del análisis y diseño tediosos y propensos a error, con influencia sobre la generación de código, las pruebas y el control. Resalta la consideración de que los beneficios potenciales sólo pueden ser alcanzados si las HDS son utilizadas de forma correcta. (Alien Fernández, 2009)

1.7.1.1 Herramientas CASE.

Visual Paradigm.

Esta es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software en el análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.
Visual Paradigm es una herramienta CASE concebida para soportar el ciclo de vida completo del proceso de desarrollo del software a través de la representación de todo tipo de diagramas entre otras opciones.

[2] Herramientas de Desarrollo de Software.

Constituye una herramienta software libre de probada utilidad para el analista y nos muestra lo siguiente: (freedownloadmanager.org, 2008)

- Diagramas de Procesos de Negocio.
- Modelado colaborativo con CVS y Subversion.
- Distribución automática de diagramas - Reorganización de las figuras y conectores de los diagramas UML.
- Dibujo de diagramas UML con plantillas (stencils) de MS Visio.
- Editor de figuras.

UML

UML[3] es un lenguaje basado en una notación gráfica la cual permite: especificar, construir, visualizar y documentar los objetos de un sistema programado. UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones, ha sido ampliamente aceptado debido al prestigio de sus creadores. Hay que tener en cuenta que el estándar UML no es un proceso, ni una metodología de desarrollo, sino una notación, un lenguaje. (Gonzalez Brito, 2005)

Principales características:

- Lenguaje unificado para la modelación de sistemas.
- Tecnología orientada a objetos.
- El cliente participa en todas las etapas del proyecto.
- Aplicable para tratar asuntos de escala inherentes a sistemas complejos de misión crítica, tiempo real y cliente/servidor.

[3] Lenguaje Unificado de Modelado.

- Facilita a los integrantes de un equipo multidisciplinario participar e intercomunicarse fácilmente, estos integrantes siendo los analistas, diseñadores, especialistas de área y desde luego los programadores.

1.7.1.2 Base de Datos.

Un Sistema Gestor de Bases de Datos (SGBD) o DBMA (DataBase Management System) es una colección de programas cuyo objetivo es servir de interfaz entre la base de datos, el usuario y las aplicaciones. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. Entre ellos se pueden mencionar Oracle, DB2, PostgreSQL, MySQL y MS SQL Server.
Entre sus características principales se encuentran: Abstracción de la información, Independencia, Redundancia mínima, Consistencia, Seguridad, Integridad, Respaldo y recuperación y Control de la concurrencia. Estos sistemas deben permitir:

- **Definir una base de datos:** especificar tipos, estructuras y restricciones de datos.
- **Construir la base de datos:** guardar los datos en algún medio controlado por el mismo SGBD.
- **Manipular la base de datos:** realizar consultas, actualizarla, generar informes. (Cavsi.com, 2007)

PostgreSQL 8.3

PostgreSQL es un sistema de gestión de Bases de Datos objeto-relacional (ORDBMS) basado en el proyecto POSTGRES de la universidad de Berkeley. Está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo, publicado bajo la licencia BSD. Tiene como principales características:

- Implementación del estándar SQL92/SQL99.

- Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.
- Incorpora una estructura de datos array.
- Incorpora funciones de diversa índole: manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta querys complejos, incluyendo subselects, integridad referencial (Foreign Keys), triggers, vistas (Views), integridad transaccional (ACID), control de versionado concurrente (MVCC). (12)

Herramientas de Administración de Base de Datos

PostgreSQL Manager 2007

Es una aplicación de alto desempeño para la administración y desarrollo de PostgreSQL Database Server. El programa trabaja con cualquier versión de PostgreSQL y soporta todas las últimas características de PostgreSQL, incluyendo espacios de tablas, nombres de argumentos en funciones y más. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo. Incluye las herramientas básicas de mantenimiento y administración y tiene como principales características:

- Soporte completo para PostgreSQL hasta la versión 8.3
- Administración y navegación rápida de bases de datos.
- Administración fácil de todos los objetos PostgreSQL.
- Administración efectiva de seguridad.
- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento. (SQLManager, 2008)

1.7.1.3 IDE de desarrollo.

Zend Studio for Eclipse 6.0

Zend Studio para Eclipse posee las capacidades para crear poderosas herramientas web. Permite a los desarrolladores crear estrategias de integración más eficientes. Al proporcionar potentes capacidades de acción con el lenguaje PHP, mejoras de soporte con JavaScript y profunda integración a Zend Framework, el desarrollo de aplicaciones se realiza en un tiempo récord. Incluye un poderoso editor de código con soporte para todos los formatos web. Posee un fuerte manejo de la arquitectura Cliente/Servidor, excelente depuración, elaboración de perfiles y un código de cobertura. Zend Studio tiene todas las herramientas que un desarrollador necesita para asegurarse de que el código es correcto y empezar a diagnosticar problemas con antelación. Tiene características de depuración avanzadas, incluyendo: condiciones límites, visualización de errores, variables y buffer de salida. Asegura la protección máxima de ubicaciones de proyectos o en Internet con depuradores remotos. (Zend.com, 2007)

1.7.2 Tecnologías Web.

1.7.2.1 Lenguajes de programación del lado del cliente.

JavaScript.

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web, puede ser utilizado por profesionales y para quienes se inician en el desarrollo y diseño de sitios web. No requiere de compilación ya que el lenguaje funciona del lado del cliente, los navegadores son los encargados de interpretar estos códigos. Entre los diferentes servicios que se encuentran realizados con JavaScript en Internet se encuentran: Correo, Chat y Buscadores de Información. También se puede encontrar o crear códigos para insertarlos en las páginas como. (Maestrosdelweb.com, 2007)

JSON

JSON[4] es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de

[4] JavaScript Object Notation o Notación de Objetos de JavaScript.

lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. Este está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

JSON se ha convertido en un estándar en el desarrollo de aplicaciones web. Los servicios web proponen la utilización de JSON en vez de XML para permitir la integración de servicios en el navegador del usuario en vez de en el servidor. Este formato es seguramente la mejor opción para el intercambio de información entre el servidor y las funciones JavaScript. (Json.org, 2007)

CSS

CSS es la tecnología desarrollada por el World Wide Web Consortium (W3C) con el fin de separar la estructura de la presentación, permite crear páginas web de una manera más exacta, gracias a las CSS el desarrollador es mucho más dueño de los resultados finales de la página, pudiendo hacer muchas cosas que no se podía hacer utilizando solamente HTML, como fuentes, colores, márgenes, líneas, altura, anchura, imágenes de fondo, posicionamiento avanzado y muchos otros temas. Entre sus beneficios se encuentran:

- Control de la presentación de muchos documentos desde una única hoja de estilo.
- Control más preciso de la presentación.
- Aplicación de diferentes presentaciones a diferentes tipos de medios (pantalla, impresión, etc.).
- Numerosas técnicas avanzadas y sofisticadas. (Desarrolloweb.com, 2008)

1.7.2.2 Lenguaje de programación del lado del servidor.

PHP

PHP[5] es un lenguaje de programación interpretado usado normalmente para la creación de páginas Web dinámicas. PHP es un acrónimo recursivo que significa "PHP Hypertext Pre-processor". Es software libre. Se puede obtener en la Web y su código está disponible bajo la licencia GPL.

- Sencillo de aprender.
- Similar en sintaxis a C y a PERL
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.
- Se puede incrustar código PHP con etiquetas HTML.
- Excelente soporte de acceso a base de datos.
- Se puede hacer de todo lo que se pueda transmitir por vía HTTP.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.
- PHP no soporta directamente punteros, como el C, de forma que no existen los problemas de depuración provocados por estos.
- Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, gestión de un e-commerce, XML, creación de PDF, etc.).

[5] Hypertext Pre-processor.

- Al poderse encapsular dentro de código HTML se puede recoger el trabajo del diseñador gráfico e incrustar el código PHP posteriormente.

- Es multiplataforma, funciona en todas las plataformas que soporten apache. (Pérez Armas, 2008)

1.7.2.3 Arquitectura.

Arquitectura Cliente\Servidor.

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario. Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Entre las principales características de la arquitectura cliente/servidor se pueden destacar:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente. (Rosabal Carrión, 2006)

Patrón Modelo – Vista – Controlador (MVC)

MVC es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de

los conceptos para que el desarrollo este estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. (Alien Fernández, 2009)

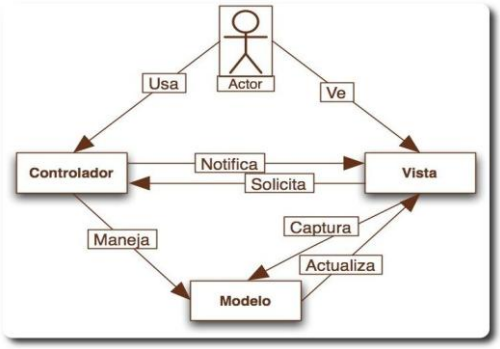

Figura 2 Estructura del patrón Modelo-Vista-Controlador (MVC)

Modelo: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista: Intercambia la información con el usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador: Recibe las entradas, traducidas a solicitudes de servicio para el modelo

El **Modelo** es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema).
- Llevar un registro de las vistas y controladores del sistema.
- Si se encuentra ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El **Controlador** es responsable de:

- Recibir los eventos de entrada.
- Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las **Vistas** son responsables de:

- Recibir datos del modelo y lo muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "Actualización", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Servidor Web Apache.

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. Tiene como características principales:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se desea ver que es lo que se está instalando como servidor, es posible verlo, sin ningún secreto, sin ninguna puerta trasera.
- Es un servidor altamente configurable de diseño modular. Es muy sencillo ampliar las capacidades del servidor. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para ser instalados para cuando se les necesite. Otra cosa importante es que cualquiera que posea una experiencia decente en la programación de C o Perl puede escribir un módulo para realizar una función determinada.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.

- Tiene una alta configurabilidad en la creación y gestión de logs. Este permite la creación de ficheros de log a medida del administrador, de este modo puedes tener un mayor control sobre lo que sucede en tu servidor. (Ciberaula.com, 2008)

1.7.2.4 Frameworks.

Zend Frameworks.

Es un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es código abierto y trabaja con PHP 5 y tiene como características principales: (Alien Fernández, 2009)

- Simplifica la gestión de archivos de configuración.
- Proporciona los componentes que forman la infraestructura del patrón Modelo- Vista-Controlador.
- Proporciona una capa de acceso a base de datos, construida sobre PDO pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos. (Echarte, 2007)

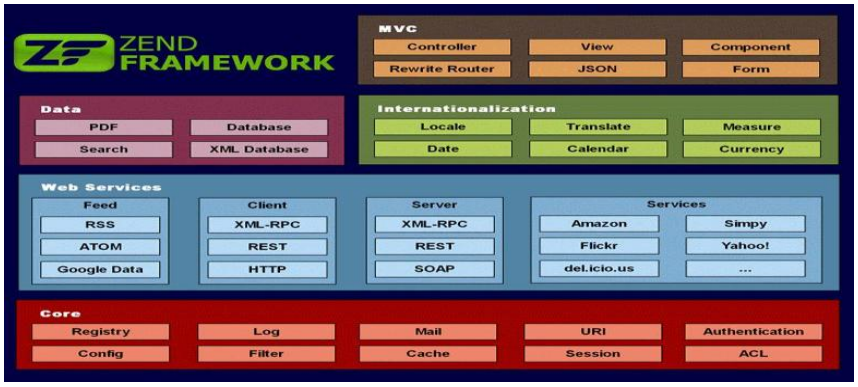

Figura 3 Estructura del Zend Framework.

Zend Ext Frameworks.

Es una librería construida con JavaScript que proporciona una interfaz a las famosas librerías de Yahoo, jQuery, y Prototype + Scriptaculous, su potencia radica en la rica colección de componentes para el diseño de GUI's del lado del cliente. Es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, ExtJS no se preocupará de lo que pase en el servidor. Hay docenas de widgets a escoger en ExtJS, incluyendo composiciones automáticas de páginas, pestañas, menús, barras de herramientas, diálogos, vistas en árbol. Proporciona un selector de nodos DOM extremadamente poderoso llamado DomQuery (puede usarse como una librería independiente, pero en el contexto de ExtJS se usará para seleccionar elementos para poder interactuar con ellos a través de la interfaz.

Ventajas.

- La orientación a objetos intensa hará modular todos los scripts.
- El diseño está completamente separado de la funcionalidad.
- Funciones comunes como validación, comboxes editables, ventanas arrastrables (con minimizar y maximizar), grillas editables, son muy fáciles de implementar.
- Buena y amplia documentación, así como también su comunidad.

Desventajas.

- Crear un sistema serio con esta herramienta requiere un previo uso prolongado, ya que se dificulta el manejo de los nuevos objetos en su extensa y bien documentada API. El tiempo de aprendizaje puede llegar a compararse con aprender a programar en un lenguaje nuevo.
- Al estar todo el sitio en JS, no podrá ser accesible para los buscadores, limitando su uso a sistemas y no sitios web.
- Si se desea algún objeto y no está, se debe asumir la compleja tarea de crear un nuevo objeto (sólo apto para programadores JS avanzados). (Wordpress.com, 2007)

UCID Frameworks.

Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJs Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación y el multilenguaje.

1.7.2.5 Navegador.

Un navegador, navegador red o navegador web (del inglés, *web browser*) es un programa que permite visualizar la información que contiene una página web (ya esté esta alojada en un servidor dentro de la World Wide Web o en uno local). El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

Mozilla Firefox 3.0

Mozilla Firefox es un navegador, con interfaz gráfica de usuario desarrollado por la Corporación Mozilla. El programa es multiplataforma y está disponible en versiones para Windows, Linux y otros sistemas operativos. El código fuente de Firefox está disponible libremente bajo la *triple licencia* de Mozilla como un programa libre y de código abierto. Firefox incorpora bloqueo de ventanas emergentes, navegación por pestañas, marcadores dinámicos, compatibilidad con estándares abiertos, y un mecanismo para añadir funciones mediante extensiones. (Montesino Alfonso, 2008)

1.7.3 Control de versiones.

Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas, proporcionándonos:

- Mecanismo de almacenaje de los elementos que deba gestionar (ej. archivos de texto, imágenes, documentación).

- Posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).

- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente pudiendo volver o extraer un estado anterior del producto).

Subversion 1.4.5

Subversión es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido. CVS[6] considerado su antecesor es uno de los controladores de versiones más utilizados en proyectos de software libre, sin embargo, a pesar de su amplio uso, el mismo diseño de CVS resultó ineficiente para diversos grupos de usuarios, y ante estas inconformidades se dio inicio al proyecto que hoy es conocido como Subversión, el mismo que ha empezado a socavar el dominio de CVS.

Ventajas.

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- La creación de ramas y etiquetas es una operación más eficiente; Tiene costo de complejidad constante (O (1)) y no lineal (O (n)) como en CVS.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).

Existen varias interfaces a Subversion, ya sea programas individuales como interfaces que lo integran en entornos de desarrollo.

- TortoiseSVN. Provee integración con el explorador de Windows. Es la interfaz más popular en este sistema operativo.
- Subversive. "Plugin" alternativo para Eclipse.
- ViewVC. Interfaz web, que también trabaja delante de CVS. (Osmosislatina.com, 2009)

[6] Concurrent Versions System.

1.8 Conclusiones.

En este capítulo se conocieron diferentes sistemas nacionales e internacionales que abarcan el control de los AFT. Ninguno de ellos realiza las operaciones de la forma en que se solicitan en el proyecto, elemento que sirvió de análisis al momento de realizar la propuesta de solución. Se sentó el basamento teórico de las herramientas a utilizar, propuestas por el equipo de Arquitectura del Centro de Soluciones de Gestión; teniendo como principales recursos las tecnologías libres, elemento fundamental en las nuevas concepciones de informatización en nuestro país. Están dadas las condiciones para conocer la solución propuesta, luego de introducidos los conceptos fundamentales para su comprensión.

Capítulo 2: Descripción de la Solución Propuesta

2.1 Introducción.

En este capítulo se abordará sobre los puntos más importantes en la implementación de los componentes Depreciación y Submayor. Se exponen los requisitos funcionales detectados por los analistas, se realizará además una profunda valoración de los artefactos propuestos por el analista del sistema, la forma en que está concebida la arquitectura y las posibilidades que proporcionan los marcos de trabajo que se utilizaron en la programación de la aplicación para poder facilitar la comprensión del funcionamiento de los componentes implementados. Se hace un análisis de las posibles implementaciones, componentes o módulos ya existentes, se describen las principales clases utilizadas en la implementación de los componentes y los estándares de codificación para una mejor legibilidad del código.

2.2 Valoración de los artefactos propuestos por los analistas.

En el proceso de desarrollo de software la descripción de los requisitos funcionales es de suma importancia, estos son brindados por el analista del sistema facilitando así un mejor entendimiento de los procesos a desarrollar, además permiten una mejor comprensión del problema en cuestión y facilitan una mejor identificación de las clases y funcionalidades que serán implementadas. Los requisitos funcionales propuestos por los analistas constituyen una entrada apropiada y un punto de partida para las actividades de implementación. Ver requisitos en los anexos.

2.3 Objeto de automatización.

En el proceso de submayor se lleva a cabo el control de todas las operaciones que se han realizado con los activos en un período de tiempo determinado, además se realiza la depreciación por área de responsabilidad. Para realizar el proceso de depreciación se calcula el valor de recuperación del AFT mediante la fórmula siguiente:

$$Vrec = Vi \times \%recuperación$$

Donde Vrec es el valor de recuperación y Vi es el costo total o valor inicial del activo. Luego se calcula la tasa de depreciación utilizando el método Línea Recta y haciendo uso de la siguiente formula:

$$TD = \frac{DA \times 100}{Vi - Vrec}$$

Donde TD es la tasa de depreciación y DA es la depreciación anual que se calcula con la siguiente fórmula:

$$DA = \frac{Vi - Vrec}{N}$$

Donde N es la vida útil del activo.

Este proceso calcula la depreciación semanal, quincenal, mensual y trimestral utilizando las siguientes fórmulas:

Depreciación semanal:

$$\text{Depreciación semanal} = \frac{DA \times TD}{52}$$

Depreciación quincenal:

$$\text{Depreciación quincenal} = \frac{DA \times TD}{24}$$

Depreciación mensual:

$$\text{Depreciación mensual} = \frac{DA \times TD}{12}$$

Depreciación trimestral:

$$\text{Depreciación trimestral} = \frac{DA \times TD}{4}$$

2.4 Propuesta del sistema.

El sistema contará con dos procesos en la gestión de los AFT, estos procesos son el Submayor y la Depreciación. El submayor brinda la posibilidad de conocer de forma general todas las operaciones que se han realizado con los activos, también muestra información de las operaciones que se han realizado con un activo específico en un periodo de tiempo determinado. Este proceso brinda la posibilidad de realizar la depreciación por área de responsabilidad. El proceso de depreciación brinda la posibilidad de

calcular la depreciación y la tasa de depreciación, y realizar la depreciación masiva. Este se realiza generalmente cuando se ejecutan operaciones como baja, ajuste por faltante y traslado de activo a ocioso.

2.5 Integración entre componentes.

2.5.1 Estrategias de integración.

La aplicación está definida por tres capas: presentación (view), negocio (controller) y acceso a datos (models). La arquitectura en tres capas consiste en el flujo de datos desde la vista hacia la capa de datos y viceversa. Esta consta de cuatro nodos de integración, el que se encuentra entre la vista y el controlador, el que está entre el controlador y el modelo, el que vincula el modelo con el framework doctrine y el que se encuentra entre el doctrine y la base de datos. Dentro de un mismo componente el código utiliza llamadas a métodos o eventos de forma directa. Los módulos y componentes se comunican mediante llamadas a la inversión de control. El IoC[7] especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario y para el conjunto de sucesos que tengan que ocurrir. Cada componente tiene su propio registro de los datos de los módulos en un fichero XML que será mapeado por el framework para el funcionamiento del mismo, este fichero se llama IoC y su función principal es registrar las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. A la base de datos se accede de forma directa mediante controladoras y los componentes rehusados se integran mediante interfaces sencillas para garantizar la integración de las capas en el sistema.

2.5.2 Diagrama de clases.

En las siguientes figuras se muestran los diagramas de clases de los componentes Depreciación y Submayor.

[7] IoC: Inversión de Control.

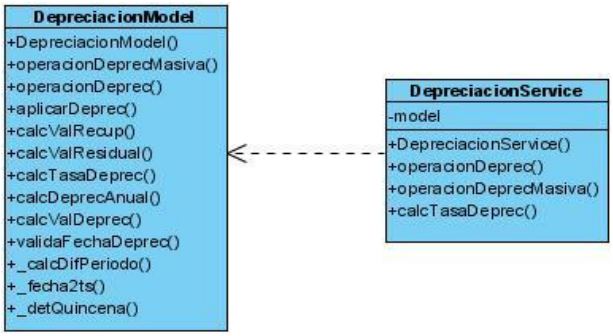

Figura 4 Diagrama de clases del componente Depreciación.

Este componente no tiene JS ni PHTML porque no es levantado por nadie ya que es un componente solamente de servicio y la relación que existe entre la clase service y la model es porque cuando al IoC le solicitan los servicios de este componente, dicha clase necesita los métodos de la clase model.

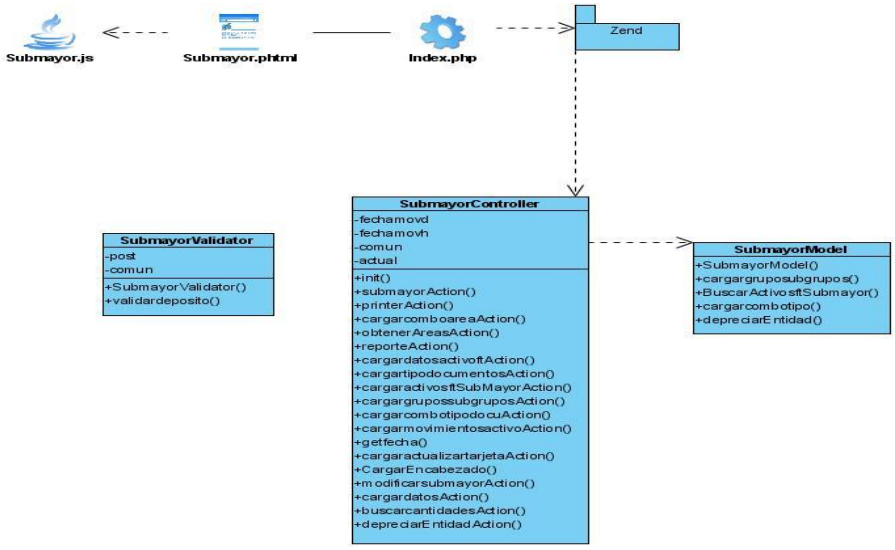

Figura 5 Diagrama de clases del componente Submayor.

En el diagrama de clases del componente Submayor, el componente Index se encarga de levantar la plantilla que sería la PHTML, donde los datos que son introducidos en ella se los envía a la JS y esta envía los datos a la clase controladora y dicha clase los envía a la clase modeladora que es donde se

implementan las funciones.

Estos diagramas constituyen una entrada para los desarrolladores, brindándoles un panorama de lo que contendrá cada clase haciendo más fácil y rápido el trabajo de estos ya que en los diagramas de clases se identifican los métodos y atributos por cada una, los parámetros que recibirán, así como la relación entre las clases.

2.5.3 Diagrama de componentes internos.

Estos diagramas representan la forma en que un sistema de software se divide en componentes mostrando así las dependencias entre dichos componentes. Los componentes físicos incluyen archivos, cabeceras, librerías compartidas, módulos, ejecutables y paquetes.

2.5.4 Diagramas de integración entre componentes.

En las siguientes figuras se muestran los diagramas de integración de los componentes Depreciación y Submayor.

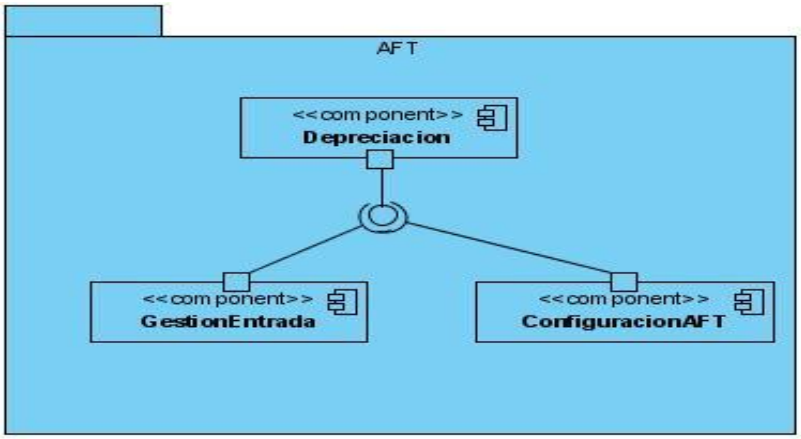

Figura 6 Diagrama de integración del componente Depreciación.

En este diagrama se muestran los componentes Gestión de Entrada y Configuración del AFT que utilizan servicio del componente Depreciación. Los servicios que utiliza ConfiguracionAFT del componente Depreciación son DevolverConfiguracionAFT y DevolverNomDepreciacion y el componente GestionEntrada toma le servicio calcTasaDeprec.

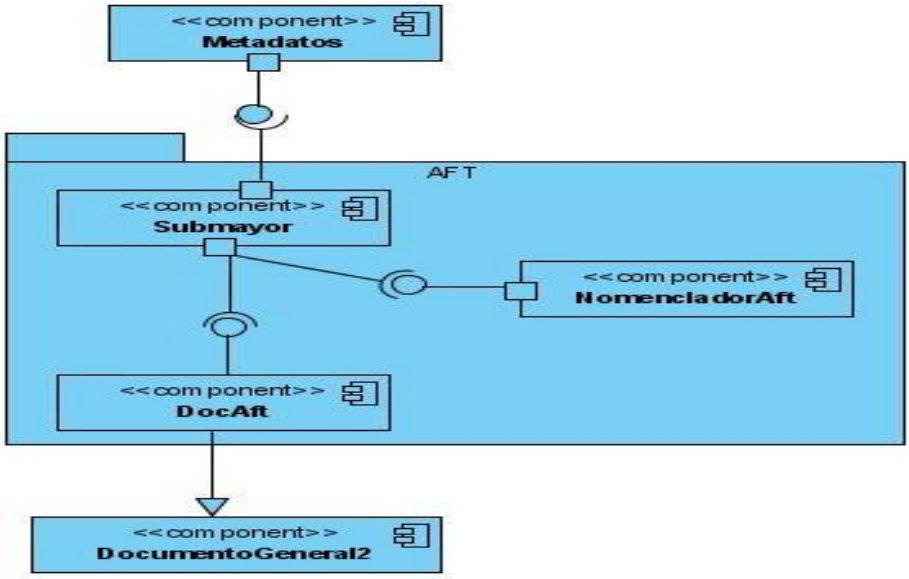

Figura 7 Diagrama de integración del componente Submayor.

En este diagrama se muestran los componentes que prestan servicios al componente Submayor. Metadatos presta el servicio DameEstructura, por su parte Nomenclador de AFT brinda el servicio ObtenerGrupo y devolverGruposubgrupos y el componente documento presta el servicio Buscartipodoc.
El diagrama de componentes internos brinda al desarrollador una vista más ampliada de los servicios provenientes de otros componentes que son necesarios para su correcto funcionamiento que además serán utilizados por el componente que se esté desarrollando.

2.6 Estándares de codificación.

Un estándar de codificación comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, que dé la impresión de que fue escrito por un solo programador, esta importante determinación se debe tomar al iniciar un proyecto haciendo que los implementadores trabajen de forma coordinada. La legibilidad del código fuente repercute directamente en el entendimiento que pueda tener otro programador del mismo, aspecto crucial ya que todo software tiene que someterse constantemente a mantenimiento y mejora de sus funcionalidades. El mejor método para lograr que un grupo de desarrolladores mantenga un código de calidad es establecer un estándar de codificación sobre el cual se realizarán revisiones rutinarias.

2.6.1 Nomenclatura de las clases.

Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto se pondrá con minúscula, cuando sea un nombre compuesto se utilizará la notación PascalCasing.

Ejemplo: Depreciación.

2.6.2 Nomenclatura según el tipo de clases.

Las clases controladoras deben llevar la palabra “Controller” después del nombre.

Ejemplo: SubmayorController.

Clases de los modelos.

Bussines.

Estas clases que se encuentran dentro de Bussines llevan la palabra “Model” después del nombre.

Ejemplo: DepreciacionModel.

Domain

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

2.6.3 Nomenclatura de las funciones.

Para estas funciones se debe escribir la primera palabra en minúscula, si es un nombre compuesto se empleará la notación CamelCasing y en caso de ser una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido del sufijo “Action”.

Ejemplo: printerAction

2.6.4 Nomenclatura de las variables.

El nombre de estas variables se escribe la primera palabra con minúscula, si es un nombre compuesto se utilizará notación CamelCasing donde se comienza con un prefijo.

Ejemplo: $objActivo

Prefijos para los tipos de datos.

Tipos de Datos	Prefijos

Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
float	flt
Boolean	Bool

Tabla 1 Prefijos a utilizar en la creación de variables.

2.6.5 Normas de comentariado.

Se debe comentariar todo lo que se haga dentro del desarrollo, establecer las pautas que conlleven a lograr un código más legible y reutilizable y así se pueda aumentar su mantenibilidad a lo largo del tiempo.

Nomenclatura de los comentarios.

Deben ser claros y precisos de forma tal que se entienda el propósito de los que se está desarrollando.

En las clases.

Antes de declarar una clase se brinda una descripción de esta donde se explica el propósito de la misma y se escribe de la siguiente manera:

Ejemplo:

```
/**
* Modelo para servicios de operaciones de depreciacion en
* Activos Fijos Tangibles (AFT).
* @author Elier Cruz Hernandez
* @package logistica
* @version 0.1
* @filesource DepreciacionModel.php
*/
```

Figura 8 Nomenclatura de los comentarios en las clases.

2.7 Tratamiento de errores.

El tratamiento de errores es de suma importancia para garantizar un correcto funcionamiento del sistema. Este último es el encargado de capturar todas las excepciones que son lanzadas y se le facilita un tratamiento para que no colapse.

De esta forma se le da solución a los problemas que implican lanzamiento de excepciones dentro del sistema según las peticiones del usuario, capturando así los tipos de errores lanzados y mostrándole al usuario mensajes de confirmación para saber con certeza lo que está incorrecto y como debe darle solución.

En el sistema existen varias funciones que necesitan validaciones previas para poder ejecutarse sin problemas, para dichas validaciones se utilizan ficheros de tipo XML, entre ellos el validator que contiene las llamadas a las precondiciones y poscondiciones a cumplirse. Por otro lado está el fichero exception que contiene los mensajes que se muestran según la excepción lanzada.

El otro fichero XML es el ManageException que captura excepciones previas que fueron definidas lo mismo para excepciones del Doctrine o del Framework. De esta manera se podrán realizar las operaciones deseadas y que se rectifique al cometer un error.

2.8 Descripción de las clases principales.

2.8.1 Clases controladoras.

En la siguiente tabla se muestra la clase controladora SubmayorController.

Nombre: SubmayorController	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:

init()	Constructor de la clase.
cargarcomboareaAction()	Cargar áreas de responsabilidad.
obtenerAreasAction()	Obtener áreas de responsabilidad.
reporteAction()	Imprimir modelo.
cargardatosactivoftAction()	Cargar los activos de la entidad.
cargartipodocumentosAction()	Cargar el tipo de documento previamente seleccionado.
cargaractivosftSubMayorAction()	Cargar lista de activo de la entidad.
cargargrupossubgruposAction()	Cargar lista de grupos y subgrupos.
cargarcombotipodocuAction()	Cargar tipos de documento.
cargarmovimientosactivoAction()	Cargar Movimiento de un activo.
cargaractualizartarjetaAction()	Cargar todos los movimientos que ha tenido un activo en un rango de tiempo.
CargarEncabezado($idestructuracomun)	Cargar datos del documento.
buscarcantidadesAction()	Buscar la cantidad de documento de submayor.
depreciarEntidadAction()	Depreciar los activos de todas las áreas de la entidad.

Tabla 2 Clase controladora SubmayorController.

2.8.2 Clases modelos de aplicación.

Nombre: SubmayorModel	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
SubmayorModel()	Constructor de la clase.
Cargargruposubgrupos()	Cargar lista de grupos y subgrupos.
BuscarActivosftSubmayor()	Cargar lista de activo de la entidad.
Cargarcombotipo()	Cargar tipos de documento.
depreciarEntidad($arrayreas,$index)	Depreciar los activos de todas las áreas de la entidad.

Tabla 3 Clase modelo de aplicación SubmayorModel.

Nombre: DepreciacionModel	
Tipo de clase: Controladora	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	Descripción:
DepreciacionModel()	constructor
calcValRecup($valinicial, $pcientorecup)	Calcula el valor de recuperación.
calcValResidual($valinicial, $depacumul)	Calcula el valor residual.
calcTasaDeprec($id_activo,$valrecup = 0)	Calcula la tasa de recuperación.
calcValDeprec($valinicial, $tasadeprec, $divconf)	Calcula el valor de la depreciación de un AFT.
validaFechaDeprec($activo)	Determina si es válido depreciar un activo teniendo en cuenta la última fecha de depreciación y la configuración establecida.
calcDifPeriodo($fechats1, $fechats2, $formret = 1)	Calcula el valor de la diferencia entre dos períodos.
detQuincena($fechaadquisicion, $fechaactual)	Determina la quincena a la que pertenece el día de una fecha con formato 'yyyy-mm-dd'.

Tabla 4 Clase modelo de aplicación DepreciacionModel.

2.8.3 Clase para validar.

Nombre: Validator

Tipo de clase: validator.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
SubmayorValidator()	Constructor de la clase.
validardeposito()	Valida la existencia del depósito.

Tabla 5 Clase SubmayorValidator.

2.9 Análisis de la complejidad de Algoritmos.

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo, es imprescindible primeramente tener el código o el diseño del procedimiento, luego marcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo, a continuación se representa el código con sus instrucciones enmarcadas y se enumeran las sentencias de código del procedimiento realizado sobre el método aplicarDepreciacion($id_activo).

```
public function aplicarDeprec( $id_activo ){
 $activo = $this->pIntegrator->activosFT->obtenerAftDadoIdAFT($id_activo); 1
 if ($this->validaFechaDeprec($activo) == true){ 2
   $depecanual = $activo->deprecanual; 3
   $valinicial = $activo->valorcontable; 3
   $depacumul  = $activo->depreciacionacum; 3
   $vidautil   = $activo->vidautil; 3
   $tasadepreciacion = $activo->tasadepreciacion; 3
   $pcientorecup = 0;  3
   $idgrupoaft = $this->pIntegrator->activosFT->devolverIdgrupoDadoIdActivo($activo->idactivofijotangible); 3
   $grupoaft = $this->pIntegrator->nomencladoresaft->ObtenerGrupo($idgrupoaft[0]->NomActivofijotangible[0]
   ['DatActivoentidad'][0]['idgruposubgrupo']); 3
   $tasafiscal = $grupoaft->tasafiscal; 3
   $idestructura = $this->global->Estructura->idestructura; 3
   $confestructura = $this->pIntegrator->configuracionaft->DevolverConfiguracionAFT($idestructura); 3
   $confaft = $this->pIntegrator->configuracionaft->DevolverNomDepreciacion($confestructura->idnomdepreciacio
       $divconf = $confaft->dividendo; 3
       $valrecup = 0;  3
       $valresidual = $this->calcValResidual( $valinicial, $depacumul ); 3
           if ($valresidual > $valrecup){ 4
               $deprec_c = ($depecanual * $tasadepreciacion) / intval($divconf); 5
               $deprec_f = ($depecanual * $tasafiscal) / intval($divconf); 5
               $activo->depreciacionacum += $deprec_c; 5
               $activo->depreciacion = $deprec_c; 5
               $activo->depreciacionacumfiscal += $deprec_f; 5
               $activo->depreciacionfiscal = $deprec_f; 5
               $activo->diferenciatasa = abs($tasadepreciacion - $tasafiscal); 5
               $activo->ultimodepre = $this->_fecha2ts($this->global->FechaContable->fecha); 5
               $this->pIntegrator->activosFT->modificarActivoFT($activo); 5
               $activo_r = $id_activo;
           }
           else{ $activo_r = 0;} 6
   }
```

Figura 9 algoritmo aplicarDepreciacion($id_activo).

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, su comprensión y brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo

- Nodo: Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hayan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.
- Aristas: Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

- Regiones: Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

En la figura se observa la representación de un grafo de flujo en el cual aparecen sus componentes.

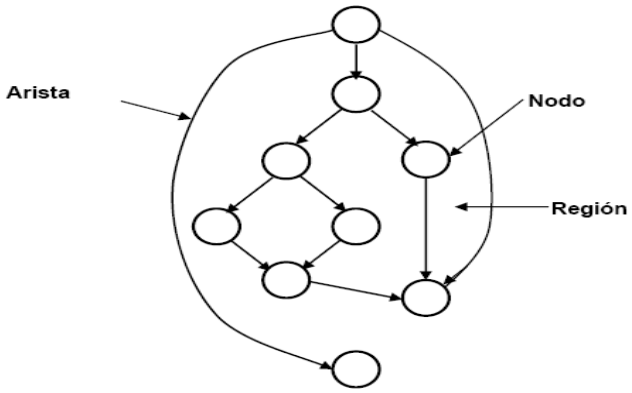

Figura 10 Componentes de los grafos de flujo.

La complejidad ciclomática es una métrica de software extremadamente útil, pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez.

En la siguiente figura se muestra el grafo de flujo asociado al código anterior.

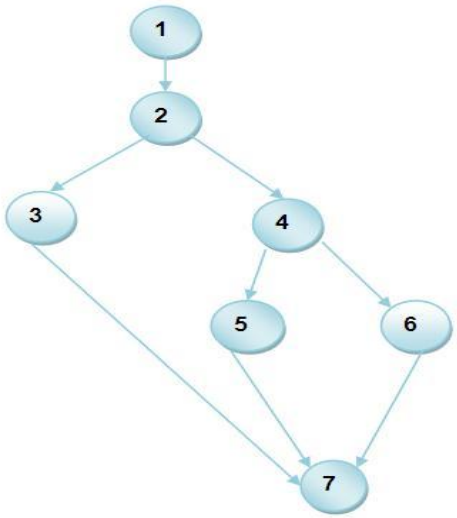

Figura 11 Grafo de flujo asociado el algoritmo aplicarDepreciacion($id_activo).

Formas fundamentales para calcular la complejidad ciclomática.

1. La complejidad ciclomática, V (G), se define como:
   V (G) = (A - N) + 2
   Donde A es el número de aristas del grafo y N es el número de nodos.

2. La complejidad ciclomática, V (G), también se define como:
   V (G) = P + 1
   Donde P es el número de nodos predicados contenidos en el grafo G.

3. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías o fórmulas, para concluir que el cálculo fue correcto es necesario que por las tres vías el resultado sea el mismo, las fórmulas para calcular son las siguientes:

V (G) = (A – N) + 2

V (G) = (8 - 7) + 2

V (G) = 3

Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G) = P + 1

V (G) = 2 + 1

V (G) = 3

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = R

V (G) = 3

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 3 que da la visión de que existen a lo sumo tres caminos lógicos por donde recorrer el algoritmo.

2.10 Conclusiones.

Con la realización de este capítulo se especificó la propuesta del sistema donde se describieron las principales clases y así facilitar la comprensión de la implementación realizada. Se comentaron los diagramas de clases de los componentes Depreciación y Submayor que brindan una vista del funcionamiento del sistema. El diagrama de integración de componentes permite conocer las relaciones de dependencias entre los componentes y las interfaces. Se conoció además todo lo referente a la integración de los componentes mediante el IoC y la arquitectura en capas utilizadas y se analizaron los estilos de código debido a que hacen más legible el programa fuente.

Capítulo 3 Validación de la Solución Propuesta

3.1 Introducción.

El desarrollo de un software es algo complejo y son innumerables las posibilidades de errores, por lo que este ha de ir acompañado de alguna actividad que garantice la calidad; las pruebas son un elemento crítico para la garantía de calidad del software. Es por eso que se utilizan técnicas como las pruebas unitarias que no son más que una forma de verificar el correcto funcionamiento del código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa ya que requiere, un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema. En este capítulo se realiza la validación a la solución propuesta en el capítulo anterior, de manera que se puede comprobar la eficiencia de las clases u operaciones utilizadas para dar respuesta a los distintos requisitos planteados por el cliente.

3.2 Validación de la implementación.

3.2.1 Pruebas de Software.

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Estas involucran las operaciones del sistema evaluando los resultados bajo condiciones controladas, lo que hace que la realización de pruebas al software sea un factor de vital importancia. Antes de liberar el producto es necesario tener la certeza de que este se encuentra libre de errores, aunque se considera que no se puede estar 100% seguro de esto, pero mientras más pruebas se le realicen al software más cerca se podrá estar de lograr un producto con la calidad esperada por los usuarios. Esta etapa de pruebas implica los siguientes aspectos:

- Verificar la interacción de componentes.
- Verificar la integración adecuada de los componentes.
- Verificar que todos los requisitos se han implementado correctamente.
- Identificar y asegurar que los defectos encontrados se han corregido antes de entregar el software al cliente.

- Diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo.

3.2.2 Pruebas de unidad.

Es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos y módulos de clase. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura (Pressman, 1998).

Los casos de pruebas que se diseñen para este nivel de pruebas, deben descubrir errores como:

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o procedencia incorrecta.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Las variables o comparaciones incorrectas.
- Terminaciones de bucles inapropiadas o inexistentes.
- Fallo de salida cuando se encuentra una iteración divergente.
- Bucles que manejan variables modificadas de forma inapropiada.

Para probar los componentes implementados como unidades individuales, se realizan las pruebas de especificación o de caja negra, que verifican el comportamiento de la unidad observable externamente y pruebas de estructura, o de caja blanca, que verifican la implementación interna de la unidad (Santos Lebeque, 2008).

Prueba del camino básico.

Alcance: Se le aplicará a la función BuscarActivosftSubmayor.

A continuación se enumeran las sentencias de código del procedimiento.

```
function BuscarActivosftSubmayor($objFiltro){
    $arrActivos = $this->pIntegrator->activosFT->obtenerTodosAFTParaSubmayor($objFiltro);[1]
    foreach ( $arrActivos as $index => $value ) [2]
    {
        $areas  = $this->integrator->metadatos->DameAreasPorId($value[arearesponsabilidad]); [3]
        $arrActivos [$index]['arearesponsabilidad']  = $areas[0]['denominacion']; [3]
        $tasacalculada = $this->pIntegrator->depreciacion->calcTasaDeprec($value['idactivofijotangible']); [3]
        $arrActivos [$index]['tasacalculada']   = $tasacalculada; [3]
        $arrActivos [$index]['codigo'] = $this->pIntegrator->activosFT->obtenerCodigoAFTDadoId
        ( $value['idactivofijotang'] ); [3]
    }

    $result =  array('cantfilas' => count($arrActivos), 'datos' => $arrActivos); [4]
    return $result; [4]
}
```

Figura 12 Algoritmo BuscarActivosftSubmayor.

Seguidamente en la Figura 11 se muestra el grafo de flujo asociado al código anterior.

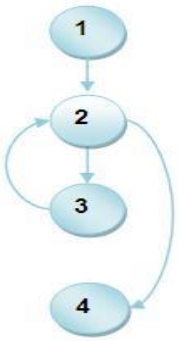

Figura 13 Grafo de flujo asociado al algoritmo BuscarActivosftSubmayor ($objFiltro)

Cálculo de la complejidad ciclomática a partir de un segmento de código.

La complejidad ciclomática es una métrica de software extremadamente útil pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Para calcular la complejidad ciclomática se deben tener en cuenta varios parámetros como son la cantidad total de aristas del grafo y la cantidad total de nodos, quedando reflejados en la siguiente fórmula.

V (G) = (A – N) + 2

V (G) = (4 - 4) + 2

V (G) = 2

Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

Se puede usar también:

V (G) = P + 1

V (G) = 1 + 1

V (G) = 2

Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = R

V (G) = 2

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del cálculo.

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 2 que da la visión de que existen a lo sumo dos caminos lógicos por donde recorrer el algoritmo.

En la siguiente tabla se muestran los caminos básicos.

Número	Camino básico
1	1 – 2 – 4
2	1 – 2 – 3 – 2 – 4

Tabla 6 Caminos básicos del flujo.

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 y es necesario que se cumplan 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.

3.3 Pruebas de aceptación.

Las pruebas de aceptación constituyen pruebas de caja negra que se centran en el cumplimiento de los requisitos definidos para el sistema una vez concluido. Por lo tanto, estas pruebas constituyen la validación de la aplicación por parte del usuario final y según Pressman plantean lo siguiente:

"La validación del software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requisitos. Un plan de prueba traza la clase de pruebas que se han de llevar a cabo, y un procedimiento de prueba define los casos de prueba específicos en un intento por descubrir errores de acuerdo con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen todos los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta e inteligible y que se alcanzan otros requisitos (por ejemplo, portabilidad, compatibilidad, recuperación de errores, facilidad de mantenimiento)". (Pressman, 1998)

Dentro de las pruebas de aceptación no formales, realizadas en software a la medida del cliente y muy convenientes cuando se desarrollan aplicaciones que van a ser utilizadas por muchos clientes, con el objetivo de descubrir errores que parezca que sólo el usuario final puede descubrir, se encuentran las pruebas alfa y beta. Estas pruebas pueden aplicarse durante semanas o meses y van desde un informal "paso de prueba" hasta la ejecución continua de una serie de pruebas bien planificadas.

Prueba Alfa.

"La prueba alfa se lleva a cabo, por un cliente, en el lugar de desarrollo. Se usa el software de forma natural con el desarrollador como observador del usuario y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado." (Pressman, 1998)

Prueba Beta.

"La prueba beta se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador." (Pressman, 1998)

Casos de pruebas.

CP #1: Listar AFT en el Submayor.

Nombre del requisito	**Descripción general**	**Escenarios de pruebas**	**Flujo del escenario**
1: Listar AFT en el Submayor.	Se muestra un listado de todos los AFT registrados en el sistema.	EP 1.1: Submayor.	El sistema muestra los datos del activo: No. Consecutivo Código Descripción Área de responsabilidad No. Inventario Vida útil Tasa fiscal Tasa calculada Depreciación acumulada/TC Depreciación acumulada/TF Valor contable

Tabla 7 CP #1: Listar AFT en el Submayor.

3.5 Conclusiones.

En el presente capítulo se realizó un análisis de las pruebas de unidad, en específico las pruebas de camino básico. Se diseñaron casos de pruebas específicos para los principales algoritmos, comprobándose que el flujo de trabajo de los mismos estuvo correcto ya que cumplieron con las condiciones necesarias que se habían planteado y se realizaron las pruebas de caja negra específicamente las pruebas Alfa y Beta donde se mostraron los casos de pruebas correspondientes a las pruebas. .

Conclusiones Generales

A modo de conclusión se plantea que el estudio realizado alrededor del proceso de la Gestión de AFT en las entidades arroja que este es de gran importancia a la hora de tomar decisiones vitales por los directivos empresariales. Esta complejidad ha propiciado el surgimiento de sistemas informáticos que simplifiquen el problema, sin embargo, no abarcan las completas necesidades de las empresas de nuestro país, ni son desarrollados sobre tecnología libre. Se detallaron las herramientas designadas para la implementación de los componentes. Para identificar e implementar las necesidades de integración con otros componentes dentro del subsistema AFT, se realizó un estudio de aquellos que podían ser reutilizados, lo que posibilitó acceder a funcionalidades que hubieran sido necesarias implementar, lo que hubiese provocado un esfuerzo mucho mayor para lograr la solución. Se logró dar cumplimiento al objetivo general al obtener una aplicación funcional que viabiliza los procesos de Depreciación y Submayor del Subsistema AFT, dentro del entorno empresarial cubano, definiendo para ello estándares de implementación que posibilitaron la excelente comunicación entre implementadores. Para darle cumplimiento al último objetivo específico se aplicaron las pruebas de caja blanca, específicamente la prueba del camino básico y las pruebas de caja negra, específicamente Alfa y Beta. Esto apoya la afirmación de que el desarrollo de los componentes Depreciación y Submayor se puede considerar como aceptable, y el código puede ser reutilizado en futuras implementaciones.

Recomendaciones

Las recomendaciones propuestas para la continuidad del presente trabajo son:

- Arreglar las no conformidades detectadas en las pruebas pilotos con el objetivo de refinar la solución.
- Ampliar las funcionalidades del subsistema con los nuevos requerimientos que surjan por necesidades de los clientes.

Referencias Bibliográficas

Alien Fernández. 2009. Implementación del Módulo Inventario Físico del Subsistema Inventario del Sistema de Gestión Integral Cedrux. 2009.

Bergin, Joseph. Building Graphical User Interfaces with the MVC pattern. [En línea] http://csis.pace.edu/ bergin/mvc/mvcgui.html..

—. 2010. Building Graphical User Interfaces with the MVC pattern. [En línea] 2010. http://csis.pace.edu/ bergin/mvc/mvcgui.html..

betsime.disaic.cu. 2007. Módulo de Activos Fijos. [En línea] 2007. http://www.betsime.disaic.cu/secciones/eco_enemar_07.

Burbeck, Steve. 2009. Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC). . [En línea] 2009. http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html..

Cavsi.com. 2007. Que es un sistema gestor de bases de datos. [En línea] 2007. http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

Ciberaula.com. 2008. Introducción a Apache. [En línea] 2008. http://linux.ciberaula.com/articulo/linux_apache_intro/..

Condor Light. 2007. Módulo de Activos Fijos. [En línea] 2007. http://www.open-sol.com.ar/php/lightmodulos.pdf.

Craig Larman. 2003. *UML y Patrones: Introducción al análisis y programación orientada a objetos. .* 2003.

Del Toro Ríos, Jose Carlos. 2008. Depreciación. 2008.

Del Toro Ríos, José Carlos. 2009. *Documento Visión. Proyecto ERP-Cuba.* La Habana : Universidad de las Ciencias Informáticas : s.n., 2009.

Desarrolloweb.com. 2008. CSS. [En línea] 2008. http://www.desarrolloweb.com/articulos/26.php..

Doctrine-Project.org. 2008. [En línea] 2008. http://www.doctrine-project.org..

Echarte, Patxi. 2007. Frameworks de Zend para el desarrollo de aplicaciones PHP. 2007.

freedownloadmanager.org. 2008. Visual Paradigm. [En línea] 2008. http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_(M%C3%8D)_14720_p ..

Fresno. 2006. Representación Autocontenida de Documentos HTML: una propuesta. [En línea] 2006. http://www.escet.urjc.es/~vfresno/phd_sp.html..

González Alvarez, Larisa y otros. 2008. GESTIÓN DEL SUBSISTEMA ACTIVOS FIJOS TANGIBLES DEL. 2008.

Gonzalez Brito, Henry Raúl y Lezcano Lozada, Yuniesky. 2005. *Sistema de Inventario Participativo de la UCI.* Ciudad de la Habana, Cuba : UCI, 2005. : s.n., 2005.

Hernández Cisneros, Sergio y Sarduy Pérez, Mileidy Magalys. 2009. Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión. 2009.

Json.org. 2007. Sitio oficial de JSON. [En línea] 2007. http://www.json.org/json-es.html..

Maestrosdelweb.com. 2007. Qué es javascript. [En línea] 2007. http://www.maestrosdelweb.com/editorial/%C2%BFque-es-javascript/..

Márquez Alpízar, Yaimí y Valdés Echavarría, Yenni. 2008. *Procedimiento general de pruebas de Caja Blanca aplicando la técnica del Camino Básico.* Ciudad de La Habana : s.n., 2008.

Martínez, Rafael. 2009. Portal de Postgres en Español. Portal de Postgres en Español. [En línea] 2009. http://www.postgresql-es.org/sobre_postgresql..

Montesino Afonso, Yasmany y Parra Lubín, Orestes. 2008. Implementación del submódulo de Recuperaciones de Información del Módulo de Recuperaciones Dinámicas. Ciudad de la Habana : Universidad de las Ciencias Informáticas : s.n., 2008.

Osmosislatina.com. 2009. Subversion. [En línea] 2009. http://www.osmosislatina.com/subversion/basico.htm..

Pérez Armas, Armando Javier y Martínez Borges, Andy Vidal. 2008. Subsistema de Configuración del Generador de Recuperaciones Dinámicas para aplicaciones Web. La Habana : Universidad de las Ciencias Informáticas, : s.n., 2008.

Pressman, R. S. 1998. *Ingeniería de software. Un enfoque práctico.* 1998.

Ríos, Jose Carlos Del Toro. 2008. Depreciación. 2008.

RodasXXI. 2010. Módulo de Activos Fijos. [En línea] 2010. http://www.rodasxxi.cu/activos%20fijos.

Rosabal Carrión, Yarisbel y Palacios Martínez, Orlando. 2006. Sistema de Contabilidad Material para la Actividad Presupuestada en las Fuerzas Armadas Revolucionarias. Módulo de entrega de medios materiales. Ciudad de la Habana: Universidad de las Ciencias Informáticas. : s.n., 2006.

Santos Lebeque, Adriana y Hernández Valladares, Danaisy. 2008. *Estrategia de Pruebas para Juegos de Realidad Virtual.* Ciudad de la Habana : s.n., 2008.

SAP Professionals. 2007. Módulo de Activos Fijos. [En línea] 2007. http://www.sap.com/argentina/ecosystem/sap_professionals/modules/index.epx.

SQLManager. 2008. [En línea] 2008. http://sqlmanager.net/..

Wordpress.com. 2007. Libreria ExtJs. [En línea] 2007. http://vargasti.wordpress.com/2007/08/06/libreria-extjs/..

—. 2007. Libreria ExtJs. [En línea] 2007. http://vargasti.wordpress.com/2007/08/06/libreria-extjs/..

XXI, Rodas. [En línea] www.rodasxxi.cu/activos%20fijos.

Yzquierdo Herrera, Raykenler y Lazo Ochoa, René. 2007. *El modelo de diseño del sistema HyperWeb. Módulos de Tratamiento Farmacológico y Configuración.* Ciudad de la Habana: Universidad de las Ciencias Informáticas, 2007. : s.n., 2007.

Zend.com. 2007. Zend Studio for Eclipse. [En línea] 2007. http://www.zend.com/products/studio/..

Glosario de Términos

Algoritmo: Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

Componente: El componente es la unidad de construcción elemental del diseño físico. Las características de un componente son:

- Se define según como interactúa con otros.
- Encapsula sus funciones y sus datos.
- Es reutilizable a través de las aplicaciones.
- Puede verse como una caja negra.
- Puede contener otros componentes.

Framework: Conjunto de APIs[8] y herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista.

Implementación: Proceso por el cual se escribe (en un lenguaje de programación), se prueba, se depura y se mantiene el código fuente de un programa informático.

Logística: Conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa o de un servicio.

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad. Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema.

Software: Se refiere a los programas y datos almacenados en un ordenador.

- Los programas dan instrucciones para realizar tareas al hardware o sirven de conexión con otro software.
- Los datos solamente existen para su uso eventual por un programa.

[8] Interfaz de Programación de Aplicaciones.

Anexos

Anexo 1: Interfaz principal del Sistema Integral de Gestión Cedrux.

Figura 14 Interfaz para el acceso al subsistema AFT.

Anexo 2: Interfaz principal para la gestión del Submayor de AFT.

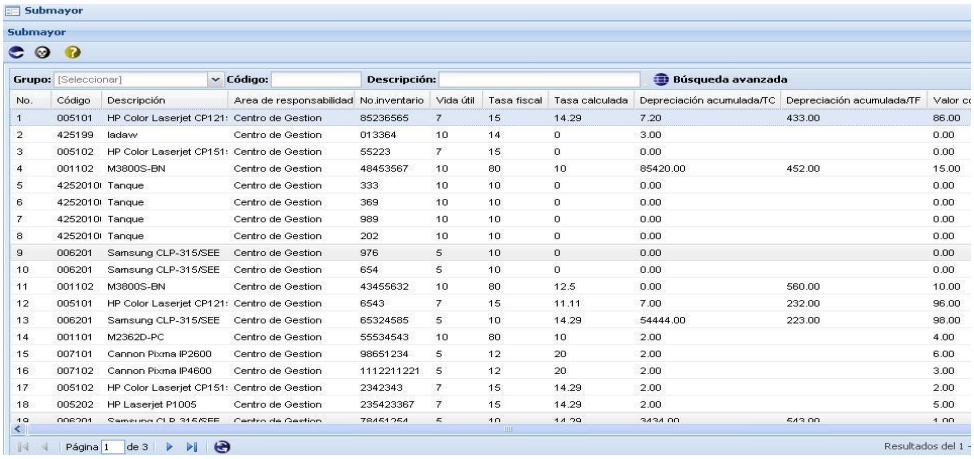

Figura 15 Interfaz principal para la gestión del Submayor de AFT.

Anexo 3: Interfaz que muestra los datos del AFT seleccionado.

Figura 16 Interfaz que muestra los datos del AFT seleccionado.

Anexo 4: Interfaz que muestra la vista previa de los movimientos de los activos.

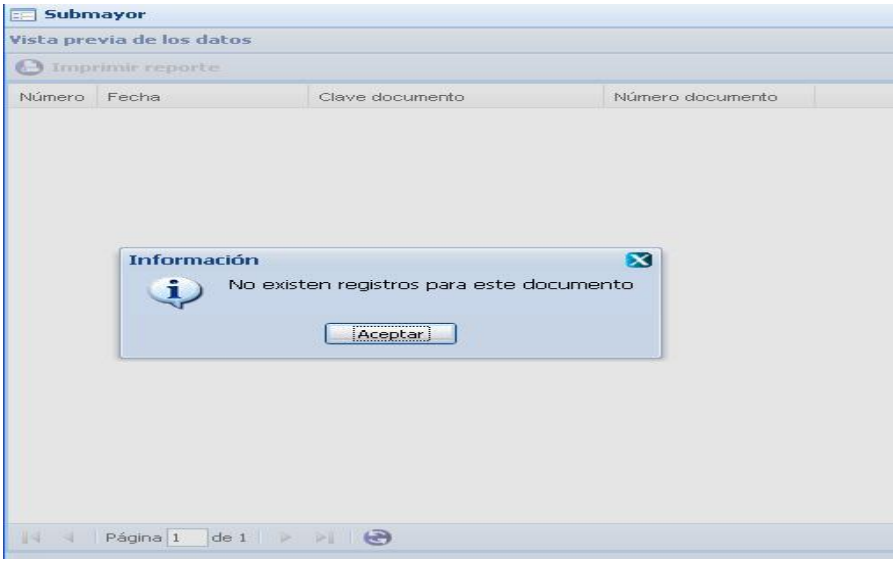

Figura 17 Interfaz que muestra la vista previa de los movimientos de los activos.

Anexo 5: Interfaz que muestra los movimientos por rangos de fecha.

Figura 18 Interfaz que muestra los movimientos por rangos de fecha.

Anexo 6: Interfaz que muestra los movimientos por períodos.

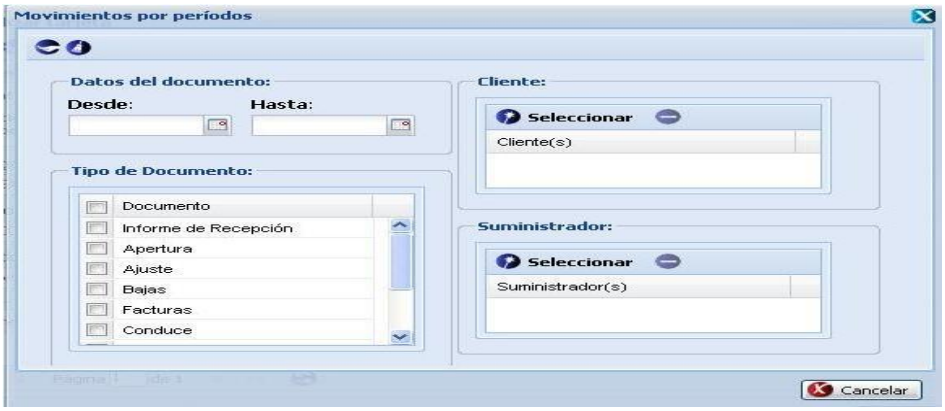

Figura 19 Interfaz que muestra los movimientos por períodos.

Anexo 7: Interfaz que muestra la depreciación por área.

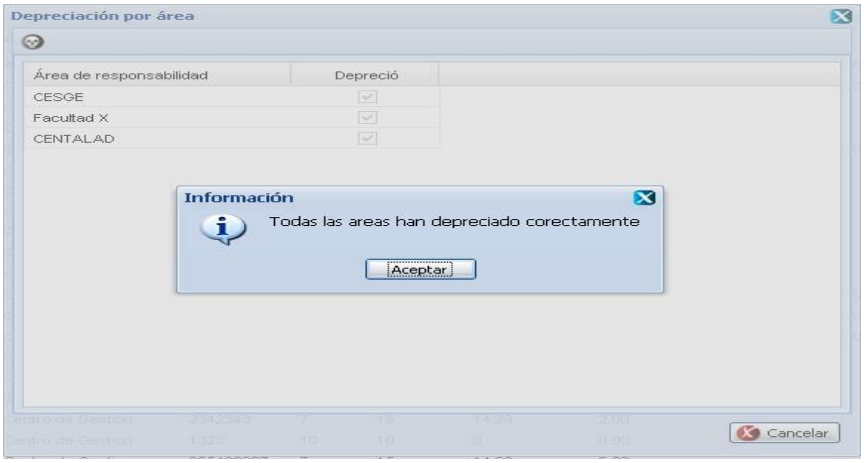

Figura 20 Interfaz que muestra la depreciación por área.

Anexo 9: Especificación de requisitos del componente Submayor.

Especificación de requisito Listar AFT en el submayor.

Descripción textual del requisito.

Precondiciones	Se ha registrado al menos un AFT en el sistema.	
Flujo de eventos		
Flujo básico		
1	El sistema muestra un listado de los activos fijos tangibles registrados en el sistema. De cada AFT se muestra: No. Consecutivo Código Descripción Área de responsabilidad No. Inventario Vida útil Tasa fiscal Tasa calculada Depreciación acumulada/TC Depreciación acumulada/TF Valor contable	
2	Concluye el requisito.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo		
1	N/A	
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A

Conceptos	**AFT**	Visibles en la interfaz: Descripción Valor Número de inventario Fecha de explotación Fecha de adquisición Deprecia Utilizados internamente: Estado
	Área de responsabilidad	Visibles en la interfaz: Descripción Utilizados internamente: N/A
	Grupo-subgrupo	Visibles en la interfaz: Descripción Código Tasa fiscal Vida útil Deprecia Utilizados internamente: N/A
	Depreciación	Visibles en la interfaz: Método de depreciación Depreciación acumulada Depreciación diaria Depreciación mensual Tasa de depreciación Diferencia de tasas Depreciación semanal Depreciación quincenal Depreciación trimestral Depreciación anual Fecha depreciación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 8 Requisito Listar AFT en el submayor.

Especificación de requisito Consultar AFT en el Submayor.

Descripción textual del requisito.

Precondiciones	Se ha registrado al menos un AFT en el sistema.	
Flujo de eventos		
Flujo básico		
1	El sistema permite seleccionar un AFT.	
2	El sistema muestra los siguientes datos: Código Descripción Vida útil Tasa fiscal Depreciación acumulada No. Inventario Depreciación Grupo Valor contable Estado Cuenta	
3	Concluye el requisito.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo		
1	N/A	
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A

Conceptos	**AFT**	Visibles en la interfaz: Descripción Valor Número de inventario Fecha de explotación Fecha de adquisición Deprecia Utilizados internamente: Estado
	Cuenta AFT	Visibles en la interfaz: N/A Utilizados internamente: N/A
	Depreciación	Visibles en la interfaz: Método de depreciación Depreciación acumulada Depreciación diaria Depreciación mensual Tasa de depreciación Diferencia de tasas Depreciación semanal Depreciación quincenal Depreciación trimestral Depreciación anual Fecha depreciación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 9 Requisito Consultar AFT en el Submayor.

Especificación de requisito Mostrar movimientos de los AFT.

Descripción textual del requisito.

Precondiciones		Se ha registrado al menos un AFT en el sistema.
Flujo de eventos		
Flujo básico		
1	El sistema permite seleccionar un AFT.	
2	El sistema muestra los siguientes datos del AFT seleccionado: No. Consecutivo Fecha Clave del documento No. Documento	
3	Concluye el requisito.	
Pos-condiciones		
1	N/A	
Flujos alternativos		
Flujo alternativo		
1	N/A	
Pos-condiciones		
1	N/A	
Validaciones		
1	N/A	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**AFT**	Visibles en la interfaz: Descripción Valor Número de inventario Fecha de explotación Fecha de adquisición Deprecia Utilizados internamente: Estado

	Documento	Visibles en la interfaz: Número documento Fecha operación Estado Entidad Tipo Nombre aprueba Cargo aprueba Observaciones Creado por Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 10 Requisito Mostrar movimientos de los AFT.

Especificación de requisito Depreciar los AFT por área de responsabilidad.

Descripción textual del requisito.

Precondiciones	Se ha registrado al menos un AFT en el sistema.
Flujo de eventos	
Flujo básico	
1	El sistema muestra las áreas de responsabilidad de la entidad.
2	El usuario selecciona la opción de depreciar.
3	El sistema calcula la depreciación de los AFT de las áreas de responsabilidad de la entidad y las marca como que ya depreciaron.
4	Concluye el requisito.
Pos-condiciones	
1	N/A
Flujos alternativos	
Flujo alternativo	
1	N/A
Pos-condiciones	

1	N/A	
Validaciones		
1	N/A	
Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**AFT**	Visibles en la interfaz: Descripción Valor Número de inventario Fecha de explotación Fecha de adquisición Deprecia Utilizados internamente: Estado
	Depreciación	Visibles en la interfaz: Método de depreciación Depreciación acumulada Depreciación diaria Depreciación mensual Tasa de depreciación Diferencia de tasas Depreciación semanal Depreciación quincenal Depreciación trimestral Depreciación anual Fecha depreciación Utilizados internamente: N/A
Requisitos especiales	N/A	
Asuntos pendientes	N/A	

Tabla 11 Requisito Depreciar los AFT por área de responsabilidad.

Anexo 11: CP #2: Consultar AFT en el Submayor.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Consultar AFT en el Submayor.	Se consultan los activos fijos.	EP 1.1: Mostrar Tarjeta.	• Se selecciona un activo. • Se selecciona el botón **Mostrar**. • Se muestran los datos del activo seleccionado: Código Descripción Vida útil Tasa fiscal Depreciación acumulada No. Inventario Depreciación Grupo Valor contable Estado Cuenta
		EP 1.2: Mostrar Tarjeta.	• Se selecciona un activo • Se selecciona el botón **Mostrar.** • Se muestran los datos del activo seleccionado. • Se selecciona el botón **Cancelar.**
		EP 1.3: Mostrar Tarjeta sin seleccionar un activo.	• El botón Mostrar no realiza ninguna función quedando deshabilitado.

Tabla 12 CP #2: Consultar AFT en el Submayor.

Anexo 12: CP #3: Mostrar movimientos de los AFT.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Mostrar movimientos de los AFT.	Se muestran los movimientos sufridos por los activos.	EP 1.1: Movimientos.	• Se selecciona un activo • Se selecciona el botón **Movimientos.** • El sistema muestra los datos del activo fijo seleccionado: No. Consecutivo Fecha Clave del documento No. Documento
		EP 1.2: Movimientos.	• Se selecciona un activo • Se selecciona el botón **Movimientos.** • Se muestra el mensaje: No existen registros para este documento. • Se selecciona el botón **Cancelar**

Tabla 13 CP #3: Mostrar movimientos de los AFT.