Universidad de las Ciencias Informáticas

Facultad 9

Título: Implementación del Componente Entrada del Módulo Activo Fijo Tangible del Sistema Integral de Gestión Cedrux.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Yasmany Antonio Bermúdez Pérez

Tutor: Ing. Yaima Álvarez Márquez

Ing. Javier Ramírez Hernández

Ciudad de la Habana. Junio de 2010

"Año 52 de la Revolución"

Pensamiento

"No hay más que asomarse a las puertas de la tecnología y la ciencia contemporáneas para preguntarnos si es posible vivir y conocer ese mundo del futuro sin un enorme caudal de preparación y conocimientos…"

Fidel Castro Ruz

Declaración de Autoría

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo la presente a los _____ días del mes de mayo del año 2010.

_____ _____

Yasmany Antonio Bermúdez Pérez **Ing. Tte. Yaima Álvarez Márquez**

Datos de contacto

Tutor: Ing. Tte. Yaima Álvarez Márquez.

Categoría Científica:

Categoría Docente:

Correo electrónico: yaimam@uci.cu

Síntesis del Tutor: Ingeniero en Ciencias Informáticas. Tres años de graduado. Diseñador de sistemas en el proyecto ERP-Logística.

Agradecimientos

A mi madre, a mi padre y a mis dos hermanos por darme el apoyo necesario para llevar a cabo esta tarea. A mis demás familiares que de una forma u otra han contribuido con la culminación de este proceso. A mi novia que ha estado siempre ahí cuando mas falta me ha hecho y a Yela (mi cuñi) que ha sido una tutora mas para mi.

A mis amigos de la UCI, que son demasiado grandes para mí, de veras. Me han apoyado desde los primeros años de universidad de manera incondicional. No quisiera hacer esto pero debo resaltar mi agradecimiento a Adrian (El Chino) que más que un amigo, para mí, ha sido un hermano más.

A mis amigos de allá de la tierra (Esperanza), que aunque están lejos de aquí he sentido siempre su apoyo. Yandy, sabes que te llevas gran parte de ese crédito.

En el UCID, a ese magnífico equipo de trabajo que me supo acoger cuando prácticamente estaba desnudo en ese mundo. A mis tutores Javier y Yaima, a Pedro, a Erik.

A todos los que de una forma u otra han hecho posible que esto sucediera caballeros son demasiados.

Agradecer de manera especial a mi BigBorther, sin él, no se donde estuviera en estos momentos; la guía que cualquiera desearía tener.

Yasmany

Dedicatoria

A toda mi familia en general, en especial a mis dos abuelitas lindas, Manuela que en paz descansa en un lugar muy cerquita de mi corazón, y a mi abuela Fello que vive conmigo en estos momentos. A ambas le debo en gran parte, lo que soy. Te extraño muncho abu M, fue un duro golpe aquel día en que nos dejaste, desearía pudieras ver en lo que me he convertido.

Yo

Resumen

En el presente trabajo de diploma se expone la implementación del componente de Entrada del Subsistema Activos Fijos Tangibles 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 propender 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 del desarrollo de éste componente se deberá lograr optimizar el proceso de gestión de entradas de 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 finalizado el componente se valida 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.

Índice de Contenido

Índice de Tablas y Figuras

Introducción

Es visible hoy en día el auge que han alcanzado las nuevas tecnologías de la Información y las Telecomunicaciones a nivel mundial. Es por eso que instituciones tales como las dedicadas a la gestión empresarial se han inclinado hacia la realización de sistemas automatizados que hagan más eficiente y económico el trabajo en estas entidades.

Es meritorio señalar el crecimiento que ha experimentado la economía cubana en estos últimos años, teniendo como uno de los principales motores impulsores el área empresarial donde se han estado buscando nuevas soluciones para poder perfeccionar los procesos relacionados con la gestión de los Activos Fijos Tangibles (AFT) de la empresas, además de la puesta en práctica de nuevas formas para el control de estos medios materiales.

Son los Activos Fijos Tangibles una parte cuantitativamente valiosa en todo proceso empresarial, por lo que, llevar el control de las entrada de los mismos en una entidad se convierte en una actividad crítica para lograr una administración exitosa. Muchas empresas en el mundo han sustituido estos procesos de forma manual por software que disminuyen considerablemente los costos tanto en materia de finanzas como de recursos humanos.

Se han convertido el software en herramientas fundamentales de muchos negocios, estos se adaptan a sus características y necesidades específicas demostrando así su gran tipología. En la actualidad se ha hecho imprescindible el uso de los Sistemas ERP (Enterprise Resource Planning), con el cual los directivos de las empresas esperan tener integradas todas las áreas o departamentos de la entidad que apoyan para la generación de sus productos y servicios, son una solución robusta para aquellas empresas que buscan una solución universal a la centralización de su información.

ERP Cuba surge por un acuerdo del Consejo de Ministros producto de la necesidad existente en el país de fortalecer la gestión de las entidades y la informatización de la sociedad cubana, enmarcado en las entidades presupuestadas y empresariales a escala nacional utilizando plataformas confiables y eficientes.

Son muchos los recursos de los que hay que disponer para llevar a cabo una propuesta como ésta, además de una cantidad considerable de desarrolladores comprometidos, debido a que son muchas las áreas que ocupa este producto. La gestión de las entradas de los AFT a las entidades es una de las actividades que lleva a cabo la logística. Es por eso que surge una línea dentro del ERP con ese nombre, donde uno de sus módulos es AFT, el cual se encarga del control de los activos en el tiempo que permanecen en la entidad. Dentro de este módulo se decidió informatizar el proceso de gestión de las entradas de AFT, el cual es de mucha importancia ya que es donde se fija una apertura inicial; se registran los datos de los AFT adquiridos, de uso, donados a la entidad. Hasta que no finalice este proceso es inútil el trabajo con estos activos.

En las entidades cubanas la situación actual en temas de gestión está marcada por el uso de sistemas antiguos, problemas de control interno, hechos de indisciplinas, ilegalidades y manifestaciones de corrupción. En el área de los AFT también se evidencian numerosos problemas en este sentido lo que conlleva muchas veces a pérdidas de los mismos. Entre los mayores problemas que se han logrado detectar en las empresas cubanas relativa a la gestión de los AFT se encuentran los que se listan a continuación:

- El control de los AFT en las entidades no automatizadas se lleva de manera manual, provocando que este proceso resulte lento.
- Poco o ningún control de la información procesada manualmente.
- Posible deterioro, daño o pérdida por desastres naturales e incendios.
- Realización de operaciones sobre los activos sin dejar trazabilidad de los mismos provocando que el contador no tenga conocimiento del origen o destino de los activos que controla.
- La búsqueda de información relativa a un activo resulta engorrosa debido a los grandes volúmenes de información existentes.

Por todo lo expuesto anteriormente se define como **problema a resolver**:

Implementación del Componente Entrada del Subsistema Activo Fijo Tangible, partiendo de los requisitos capturados del componente Entrada del Subsistema AFT del Sistema Integral de Gestión Cedrux[1].

El problema está enmarcado en el **objeto de estudio**: los procesos de gestión de AFT y como **campo de acción** el proceso de gestión de entradas de AFT.

El **objetivo** general de este trabajo es Implementar el componente Entrada del Subsistema AFT del Sistema Integral de Gestión Cedrux.

Para darle cumplimiento a este objetivo se plantearon los siguientes **objetivos específicos**:

- Analizar el proceso de Entrada de AFT para una entidad y los sistemas que existen actualmente para su gestión, así como las herramientas que se utilizarán para el desarrollo de la solución.
- Implementar un módulo para la gestión del Entrada de los AFT en cualquier entidad.
- Validar la solución propuesta.

Las **tareas** que se realizarán para dar solución a los objetivos específicos planteados son:

- Analizar el proceso para dar entrada a los AFT en una entidad.
- Análisis de los componentes de entrada de los sistemas de Gestión de AFT nacional e internacional.
- Estudio de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
- Análisis de los artefactos entregados por los analistas.
- Realización de pruebas de unidad a los componentes obtenidos.
- Implementación de las interfaces a partir del prototipo entregado por los analistas.
- Implementación de la capa de negocio que dé respuesta a los requisitos propuestos por los analistas.
- Implementación de la capa de acceso a datos.
- Implementación de las validaciones y excepciones.

[1] **Cedrux:** Vocablo formado por la unión de las palabras “Cedro” (fortaleza, resistencia) y “Linux” (tecnología libre).

- Implementación de los servicios incluidos dentro de sus responsabilidades que se necesiten para la implementación de otros módulos y componentes.
- Validación de la implementación de los componentes obtenidos a través de métricas.

Para la realización de esta investigación se ha planteado la siguiente **Idea a defender**:

Si se realiza la implementación del componente Entrada, perteneciente al subsistema AFT del Sistema Integral de Gestión Cedrux, se obtendrá un producto integrado para la gestión de dichos procesos.

Resultados esperados:

Se contará con un componente funcional, con todos los requerimientos analizados y priorizados por los usuarios funcionales del mismo e integrado con otros módulos y componentes que son necesarios para su correcto funcionamiento y flujo de información.

Además permitirá realizar todas las operaciones de una forma ágil y cumpliendo todo lo establecido.

Los resultados esperados con el sistema al ponerse en práctica son:

- Mayor control sobre los activos en una entidad.
- Mayor caracterización de los activos en una entidad.
- Disminuirá de manera gradual el tiempo del proceso de gestión de las entradas de activos a una entidad.
- Permitirá reducirlos costos.
- Facilitará el trabajo en grupo de los integrantes del área, ayudando así a una mayor organización.

.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

Es indudable que el ambiente competitivo en el que se vive en el ámbito empresarial actualmente, requiere de promover los procesos y actividades de negocio que generan las ventajas competitivas de las compañías ante sus más fuertes competidores.

Hoy más que nunca las empresas requieren de herramientas que les proporcionen control y centralización de su información, esto con el fin de tomar las mejores decisiones para sus procesos y estrategias de negocios. Los ERP son una solución robusta para aquellas empresas que buscan la centralización de su información, estos sistemas automatizan los procesos para el Control de las Entradas de AFT.

Además de los sistemas mencionados anteriormente, también se han desarrollado software relacionados con el Control de Entradas de AFT que no forman parte de paquetes integrados, cuyo análisis ha servido para tener en cuenta las ventajas y desventajas que presentan. A esto se debe sumar la política que implementa Cuba para migrar a software libre con el objetivo de lograr su soberanía tecnológica producto a restricciones impuestas por el bloqueo de Estados Unidos de América.

Debido a la diversidad de sistemas existentes, surge la necesidad de hacer un estudio tanto en nuestro país como en el mundo y las herramientas a utilizar, buscando obtener la información más actual de las mismas.

1.2 Conceptos generales

1.2.1 ¿Qué es un ERP?

Los sistemas de Planificación de Recursos Empresariales (Enterprise Resource Planning, ERP) son sistemas de gestión de información gerenciales que integran, automatizan y manejan muchas de las acciones asociadas con las operaciones de producción y distribución de una compañía dedicada a la producción de bienes o servicios.

Los ERP pueden intervenir en el control de muchas actividades de negocios, tales como ventas, compras, entregas, pagos, producción, contabilidad, administración de inventarios y de recursos humanos. Funcionan en todo tipo de empresas. Para ello, integran todos los departamentos funcionales de la

empresa, herramientas de mercadotecnia y administración estratégica en un solo sistema (Wailgum, 2008).

1.2.1.1 Características de los sistemas ERP.

Un ERP se distingue de otro software empresarial porque es un sistema integral, con modularidad y adaptable (Wailgum, 2008).

- Integrales: Todos los departamentos o áreas de una empresa se relacionan entre sí, permitiendo de esta forma controlar los distintos procesos de la entidad, donde la salida de un proceso puede ser la entrada de otro.
- Modulares: Se dividen en módulos, que son las áreas que se integran como un todo para el manejo óptimo de la información. Estos módulos se instalan según las necesidades del cliente.
- Adaptables: Son diseñados para ser configurables según lo que cada empresa necesite.
- Base de datos centralizada.
- Sus componentes interactúan entre sí consolidando todas las operaciones.
- En un sistema ERP los datos se ingresan sólo una vez y deben ser consistentes, completos y comunes.
- Las empresas que lo implanten suelen tener que modificar alguno de sus procesos para alinearlos con los del sistema ERP. Este proceso se conoce como Reingeniería de Procesos, aunque no siempre es necesario.

1.2.1.2 Importancia.

Aplicando un ERP en una empresa se brinda apoyo a los clientes del negocio, se ofrecen respuestas rápidas a los problemas y se optimiza el manejo de información que permita la toma oportuna de decisiones y disminución de los costos totales de operación. Esto se obtiene gracias a que es un sistema integrado. De esta forma se evita tener varios programas que controlen todos los procesos de una empresa, situación en la cual es más difícil lograr que la información no se duplique, que no aumente el margen de contaminación en la información y que no se cree un escenario favorable para malversaciones. Con un sistema ERP la información no se manipula y se encuentra protegida.

1.2.2 ¿Qué es una Entrada?

Una Entrada es donde fija una apertura inicial; se registran todos los datos de los Activos Fijos Tangibles adquiridos, de uso, donados a la entidad.

1.3 Situación actual de los procesos

Actualmente en Cuba no existe 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, et al., 2009).

1.4 Sistemas automatizados existentes

1.4.1 Sistemas nacionales

1.4.1.1 Versat-Sarasola

Es un producto cubano cuyo Sistema Integral de gestión es el primer sistema de contabilidad certificado, desarrollado para la gestión económica eficaz y fiable. Actualmente es utilizado en nuestro país 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 encuentran:

- Configuración y seguridad.
- Contabilidad general y de gastos.
- Costos y procesos.
- Análisis económico empresarial.
- **Control de activos fijos.**
- Finanza y caja.
- Planificación y presupuestos.
- Control de inventarios.

- Pago de salario.
- Paquete de gestión.
- Contratación.
- Facturación.

(del Toro Ríos, et al., 2009)

El módulo de **Control de activos fijos** 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:

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

Características de Versat-Sarasola:

- Es una aplicación de escritorio.
- Implementado en Delphi.
- Trabaja sobre el sistema operativo Windows.
- Soporte para base de datos SQL Server 2000.

1.4.1.2 Rodas XXI

Sistema multiempresa y multiusuario creado por CITMATEL para la automatización de la gestión empresarial. Contiene diferentes módulos que pueden usarse integrados o independientes:

- Contabilidad.
- Efectivo caja y banco.

- Nóminas.
- **Activos fijos**
- Inventarios.
- Cobros y pagos.
- Facturación.
- Finanzas.
- Tele-cobranza.

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.
- Realizar operaciones de activos fijos en el momento que se desee, generando el documento asociado al movimiento de que se trate de forma automática previa configuración del sistema para ello.
- Permite el control por separado de los activos fijos que se encuentran en almacén de los que se encuentran en explotación.
- El comprobante de depreciación se genera, al igual que con los movimientos de activos fijos, de forma automática.
- Permite obtener el submayor de activos fijos, listados y localización de los medios de transporte de la entidad, la depreciación mensual y acumulada de uno o de los activos fijos que desee, el acta de responsabilidad material de cada una de las áreas.
- Visualiza información correspondiente a períodos anteriores, tan sólo con cambiar de período contable a periodos anteriores ya cerrados, aunque en dichos periodos no podrá realizar ninguna operación.

Además, cuenta con el módulo Administrador, que brinda mayor integralidad al sistema y garantiza facilidades adicionales durante su instalación y explotación (RodasXXI.cu. 2010).

1.4.1.3 Valoración crítica.

Una vez analizados los sistemas implantados en Cuba, se concluye que no resultan soluciones factibles para las entidades cubanas debido a que fueron desarrollados sobre plataformas de software propietario, lo que implica incrementos de gastos en licencias de uso y mantenimiento del software. Además, las soluciones nacionales constituyen aplicaciones de escritorio lo que trae como desventaja que el usuario deba instalar la aplicación en cada estación de trabajo. Son productos que se caracterizan por abordar solamente partes del problema de la gestión de la empresa o la unidad presupuestada, no soportan mecanismos estándares de integración con otras aplicaciones donde la mayoría fueron desarrollados para un ambiente multiusuario, casi ninguno bajo conceptos de informática multicapa y distribuida en la red.

1.4.2 Sistemas internacionales

1.4.2.1 Openbravo

Openbravo ERP ha sido específicamente diseñado para ayudar a las empresas a mejorar su rendimiento. La cobertura funcional del producto incluye todas las áreas típicas de un sistema de gestión integrado, destacando la solución de contabilidad.

Además, esta aplicación se integra de manera natural con otras áreas como la gestión de relaciones con clientes o CRM[2], inteligencia de negocio o BI (Business Intelligence) y terminales punto de venta o POS (Point of Sale).

Openbravo ERP utiliza tecnologías modernas, pero sólidas y suficientemente probadas, para cumplir los requerimientos estrictos de rendimiento y escalabilidad de cualquier entorno empresarial:

- Java y Java script
- SQL y PL/SQL
- XML
- HTML

Los procesos de gestión de almacenes que incorpora Openbravo ERP permiten que las existencias en su organización estén siempre al día y correctamente valoradas. La posibilidad de definir la estructura de

[2] **CRM**: Customer Relationship Management

almacenes de su organización hasta el mínimo nivel (ubicación) facilita que los stocks estén siempre perfectamente localizados. Adicionalmente, las capacidades para gestionar los lotes de mercancías y la posibilidad de utilizar números de serie aseguran el cumplimiento de los requisitos de trazabilidad impuestos en la mayoría de industrias (Openbravo.com, 2009).

La Gestión de almacenes incluye:

- Almacenes y ubicaciones (multi-almacén).
- Stock por producto en doble unidad (por ejemplo, en kilogramos y cajas).
- Atributos del producto en almacén personalizables (color, talla, descripción de calidad, etc.).
- Lote y número de serie.
- Impresión de etiquetas. Códigos de barras (EAN, UPC, UCC, Code, otras.).
- Gestión de bultos en almacén.
- Control de reposición.
- Trazabilidad configurable por producto.
- Movimiento entre almacenes.
- Gestión automática de salidas de stock (vaciado según existencias, con reglas de prioridad por caducidad, ubicación, etc.).
- Inventario físico. Planificación de inventarios. Inventario continuado.
- **Informes de movimientos, seguimiento, stocks, entradas/salidas, caducidades, inventario, ubicaciones, etc. Informes personalizables.**
- Integrado con Openbravo POS.
- Sincronización y control del stock en la misma tienda

1.4.2.2 Condor (Light)

Sistema automatizado de alta complejidad y seguridad que abarca todos los aspectos del proceso contable de una entidad, tales como la dualidad de moneda y el pago por resultados. Está formado por varios Módulos:

- **Activos fijos.**
- Contabilidad general.
- Nóminas.
- Control de inventarios.
- Condexce.
- Recursos humanos.

Este brinda mayor autonomía al cliente para efectuar cambios de estructura sin necesidad de la intervención de especialistas, quedando registrados de forma que puedan ser auditables. Incluye la contabilidad multimonedas (del Toro Ríos, et al., 2009).

Entre sus módulos se encuentra el de Activos Fijos, el 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.
  (open-sol.com, 2010)

1.4.2.3 Valoración crítica

Desde el punto de vista de la solución, estos sistemas serían capaces de resolver el problema del módulo de Entrada de AFT debido a su alto nivel de configuración y servicios que proveen, pero tienen la desventaja de que algunos de ellos utilizan tecnologías que no son accesibles a Cuba debido a las restricciones impuestas por Estados Unidos. Sistema como OpenBravo está basado en la plataforma

J2EE cuya máquina virtual es propiedad de SUN, empresa norteamericana, aunque ha comenzado a liberar el código sigue estando bajo las leyes de su gobierno; además J2EE requiere un consumo de memoria elevado en comparación con PHP/Apache. Como otra desventaja aparece que el diseño de estos ERP ha sido para empresas capitalistas que tienen un modelo de gestión y de procesos muy diferente a las empresas o unidades presupuestadas cubanas donde la economía es centralizada y operan otros mecanismos. Por último, el resto de los software propietarios no constituyen una opción viable pues representan gastos muy elevados al país por conceptos de licencias y mantenimiento.

1.5 Modelo de desarrollo de software.

El modelo de desarrollo de software propuesto describe la secuencia de actividades de alto nivel para la construcción y desarrollo de soluciones. Se logra con la combinación entre los modelos basado en Componentes, el Iterativo y el Incremental. Se emplearán las técnicas de prototipado, si son requeridas, para los requerimientos del usuario de los que no existe una visión clara por parte de estos, con el objetivo de desarrollar una definición mejorada de los requisitos del usuario para el sistema.

Desarrollo iterativo e incremental: Es un enfoque en el que el ciclo de vida está compuesto por iteraciones, estas son pequeños procesos compuestos de varias actividades cuyo objetivo es entregar una parte del sistema parcialmente completo, probado, integrado y estable. Todo el software es integrado en cada entrega de cada iteración hasta obtener el producto de software completo en la última iteración. En cada iteración se obtiene como resultado un incremento.

"Un componente es una unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio" [Szyperski, 1998]

Desarrollo basado en componentes: Nos lleva a alcanzar un mayor nivel de reutilización de software, aún en contextos distintos a aquellos para los que fue diseñado. Permite que las pruebas sean ejecutadas probando cada uno de los componentes antes de probar el conjunto completo de componentes ensamblados. Cuando existe un débil acoplamiento entre componentes, el desarrollador es libre de actualizar y/o agregar componentes según sea necesario, sin afectar otras partes del sistema. Dado que un componente puede ser construido y luego mejorado continuamente, la calidad de una aplicación basada en componentes mejorará con el paso del tiempo. (PDS, 2009)

1.6 Herramientas y tecnologías utilizadas

1.6.1 Herramientas de desarrollo de software

Actualmente se considera a las Herramientas de Desarrollo de Software (HDS) como herramientas basadas en computadoras que asisten el proceso de ciclo de vida de software, consolidadas en la literatura en la forma de ingeniería de software asistida por computadora (CASE, por sus siglas en inglés).

Permiten automatizar acciones bien definidas, reduciendo también la carga cognitiva del ingeniero, quien requiere libertad para concentrarse en los aspectos creativos del proceso. Este soporte se traduce en mejoras a la calidad y la productividad en el diseño y desarrollo. Las HDS automatizan metodologías de software y desarrollo de sistemas y se vinculan con los diferentes conceptos involucrados en el desarrollo.

El soporte que brindan al proceso de desarrollo proporciona importantes ventajas para el equipo de trabajo. Estas mejoras se sintetizan en:

- Apoyan a las metodologías y métodos, integrando actividades y propiciando visión de continuidad entre fases metodológicas.
- Mejoran la comunicación entre los actores involucrados, facilitándoles compartir su trabajo y desempeñarlo de forma dinámica e iterativa.
- Establecen métodos efectivos para almacenar y utilizar los datos, lo que permite organizar y correlacionar componentes, para acceder a estos a través de un repositorio.
- Agregan 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.
- Automatizan 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.

1.6.1.1 Herramientas CASE

Visual Paradigm for UML 6.1

Visual Paradigm para UML[3] es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Constituye una herramienta software libre de probada utilidad para el analista. Dentro de sus características se aprecia que soporta BPMN y UML versión 2.1. Muestra también:

- Diagramas de Procesos de Negocio.
- Modelado colaborativo con CVS y Subversión.
- Generador de informes para generación de documentación.
- 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.

(freedownloadmanager.org, 2008)

Lenguajes para el modelado

El lenguaje de modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos o parte de este.

Algunas organizaciones los usan extensivamente en combinación con una metodología de desarrollo de software para avanzar de una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores. El uso de un lenguaje de modelado es más sencillo que la

[3] **UML:** (Unified Modeling Language) Lenguaje Unificado de Modelado

auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo.

UML (Lenguaje Unificado de Modelado)

UML (Unified Modeling Lenguaje) o Lenguaje de Modelación Unificado 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 aceptada debido al prestigio de sus creadores. Hay que tener en cuenta que el estándar UML no es un proceso, no es una metodología de desarrollo, sino una notación un lenguaje (Gonzalez Brito, y otros, 2005).

De forma general las principales características son:

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

1.6.1.2 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.

Algunos ejemplos de SGBD son Oracle, DB2, PostgreSQL, MySQL, MS SQL Server, etc.

Las características de un Sistema Gestor de Base de Datos SGBD son: Abstracción de la información, Independencia, Redundancia mínima, Consistencia, Seguridad, Integridad, Respaldo y recuperación, Control de la concurrencia.

Un SGBD debe 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.

A continuación se enumeran las principales características de este gestor de bases de datos:

- 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).

(Netpecos.org, 2008)

Herramientas de Base de Datos

pgAdmin III

pgAdminIII es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usan en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como Pervasive Postgres, EnterpriseDB, Mammoth Replicator y SRA PowerGres. Incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados, soporte para el motor de replicación Slony-I y mucho más. La conexión al servidor puede hacerse mediante conexión TCP/IP[4] o Unix Domain Sockets (en plataformas *nix), y puede encriptarse mediante SSL[5] para mayor seguridad (Guia-ubuntu.org, 2007).

SQLmanager 2007

SQL Manager 2007 para PostgreSQL es una poderosa herramienta gráfica para administración de servidor de PostgreSQL y el desarrollo. Su fácil de usar interfaz gráfica y un montón de características que hacen su trabajo con el PostgreSQL es tan fácil como puede ser (Sqlmanager.net, 2007).

Principales características:

- Soporte de datos UTF8.
- Excelente herramientas visuales y texto de consulta para la construcción.
- Nuevo estado de la técnica interfaz gráfica de usuario.
- Rápida gestión de bases de datos y la navegación.
- Avanzadas herramientas de manipulación de datos.
- Soporte completo de PostgreSQL hasta la versión 8.3.

[4] **TCP/IP:** Transmission Control Protocol/Internet Protocol (Protocolo de control de transmisión/Protocolo de Internet)

[5] **SSL:** Secure Sockets Layer (Protocolo de Capa de Conexión Segura)

- Mejora de la base de datos para facilitar su explorador gestión de todos los objetos de PostgreSQL.
- Eficaz de gestión de la seguridad.
- Potente base de datos de diseño visual.
- Conexión de puerto local a través de la transmisión a través de un túnel SSH[6].
- Acceso a servidor PostgreSQL a través de protocolo HTTP[7].

1.6.1.3 IDE de desarrollo

Un IDE[8] es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica GUI. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias Visual Basic en forma de macros para Microsoft Word.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C.

Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. Este es el caso de Eclipse, al que mediante pluggins se le puede añadir soporte de lenguajes adicionales.

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

[6] **SSH:** Secure SHell, (Intérprete de órdenes seguro)

[7] **HTTP:** HyperText Transfer Protocol (Protocolo de Transferencia de Hipertexto)

[8] **IDE:** Integrated Development Environment (Entorno de desarrollo integrado)

acción con el lenguaje PHP, mejoras de soporte con Java Script 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 estudio 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).

1.6.2 Tecnologías Web

1.6.2.1 Lenguajes de programación del lado del cliente

HTML

HTML[9] es el lenguaje con el que se definen las páginas web. Básicamente se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web.

Es un lenguaje de marcación de elementos para la creación de documentos hipertexto, muy fácil de aprender, lo que permite que cualquier persona, aunque no haya programado en la vida, pueda enfrentarse a la tarea de crear una web. HTML es fácil y pronto podremos dominar el lenguaje. Más adelante se conseguirán los resultados profesionales gracias a nuestras capacidades para el diseño y nuestra vena artista, así como a la incorporación de otros lenguajes para definir el formato con el que se tienen que presentar las webs, como CSS[10] (Desarrolloweb, 2007).

Java Script

Java Script 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.

[9] **HTML:** HyperText Markup Language (Lenguaje de Marcas de Hipertexto)

[10] **CSS:** Cascading Style Sheets (Hojas de Estilo en Cascada)

Entre los diferentes servicios que se encuentran realizados con Java Script en Internet se encuentran: Correo, Chat, Buscadores de Información.

También podemos encontrar o crear códigos para insertarlos en las páginas como: Reloj, Contadores de visitas, Fechas, Calculadoras, Validadores de formularios, Detectores de navegadores e idiomas (Maestrosdelweb.com, 2007).

Ajax

AJAX, acrónimo de Asynchronous Java Script And XML[11] (Java Script y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador del usuario, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX es una combinación de tres tecnologías ya existentes:

XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.

DOM[12] accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como Java Script y Script, para mostrar e interactuar dinámicamente con la información presentada.

El objeto XMLHttpRequest[13] para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Como el DHTML, LAMP o SPA, AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente (Webexperto.com, 2006).

[11] **XML:** Extensible Markup Language (Lenguaje de Etiquetado Extensible)

[12] **DOM:** Document Object Model (Modelo en Objetos para la representación de Documentos)

[13] **XMLHttpRequest**: Interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB.

Json

JSON [14] 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 Java Script. 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 lenguajes C, incluyendo C, C++, C#, Java, Java Script, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON 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.

El formato JSON es el más adecuado para la respuesta del servidor cuando la acción Ajax debe devolver una estructura de datos a la página que realizó la llamada de forma que se pueda procesar con Java Script. Este mecanismo es útil por ejemplo cuando una sola petición Ajax debe actualizar varios elementos en la página.

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. El formato JSON es seguramente la mejor opción para el intercambio de información entre el servidor y las funciones Java Script (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 somos mucho más dueños 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.

[14] **JSON:** Java Script Object Notation (Notación de Objetos de JavaScript)

Entre los beneficios concretos de CSS encontramos:

- 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)

1.6.2.2 Lenguaje de programación del lado del servidor

Un lenguaje de programación es un lenguaje que puede ser utilizado para controlar el comportamiento de una computadora. Consiste en un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un lenguaje de programación permite a uno o más programadores especificar de manera precisa: sobre qué datos una computadora debe operar, cómo deben ser estos almacenados, transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural.

Se les clasifica como lenguajes del lado del servidor a los lenguajes de programación en la tecnología cliente servidor que se ejecutan del lado del servidor y de los que los cuales los usuarios solo obtienen el beneficio del procesamiento de la información.

PHP

PHP es el acrónimo de Hipertext Preprocesor. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Se escribe dentro del código HTML, lo que lo hace realmente fácil de utilizar. Es independiente de plataforma, puesto que existe un módulo de PHP para casi cualquier servidor web. Esto hace que cualquier sistema pueda ser compatible con el lenguaje y significa una ventaja importante

Algunas de las más importantes capacidades de PHP son: compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, y ODBC[15], por ejemplo. Incluye funciones para el envío de correo electrónico, upload de archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

No es un lenguaje de marcas y la meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas.PHP también soporta el uso de otros servicios que usen protocolos como IMAP[16], SNMP[17], POP3[18], HTTP y derivados. También se pueden abrir sockets de red directos (raw sockets) e interactuar con otros protocolos.

Al ser un lenguaje libre dispone de una gran cantidad de características que lo convierten en la herramienta ideal para la creación de páginas Web dinámicas: Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader), analizar código XML. Perceptiblemente más fácil de mantener y poner al día que el código desarrollado en otros lenguajes.

(Desarrolloweb, 2009)

1.6.2.3 Arquitectura

Una Arquitectura de Software, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Esta se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para

[15] **ODBC:** (Open Database Connectivity) Estándar de acceso a Bases de datos desarrollado por Microsoft Corporation.

[16] **IMAP:** (Internet Message Access Protocol) Protocolo de acceso a mensajes almacenados en un servidor.

[17] **SNMP:** (Simple Network Management Protocol) Protocolo de la capa de aplicación que facilita el intercambio de información de administración.

[18] **POP3:** (Post Office Protocol) Protocolo 3 de Correo, está diseñado para recibir correos, no para enviarlo.

determinadas arquitecturas. Por ejemplo, no es viable emplear una arquitectura de software de tres capas para implementar sistemas en tiempo real.

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. MVC sugiere la separación del software en 3 estratos.

Modelo: Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en un contexto del sistema proveen de información al usuario o a la aplicación misma.

Vista: Es la representación del modelo en forma grafica disponible para la interacción con el usuario. En el caso de una aplicación web la "Vista" es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

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).

En la siguiente figura se muestra la estructura del patrón Modelo-Vista-Controlador.

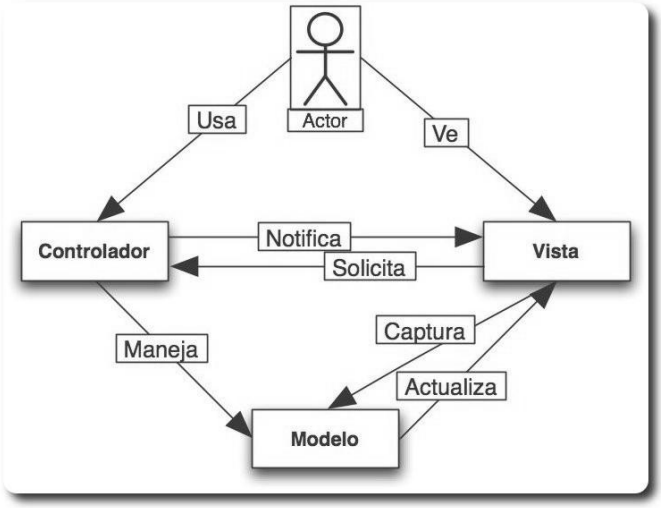

Figura 1 Estructura Modelo-Vista-Controlador

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo, independientemente de la representación visual.

Ventajas:

Soporte de múltiples vistas: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos mostrado de maneras diferentes.

Adaptación al cambio: Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación, o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.

Desventaja:

Costo de actualizaciones frecuentes: Si el modelo experimenta cambios frecuentes, por ejemplo, podrían desbordar las vistas con una lluvia de requerimientos de actualización.

Arquitectura Cliente-Servidor

Esta arquitectura consiste básicamente en un programa cliente que realiza peticiones a otro programa -servidor- que le da respuesta. Aunque esta idea se puede aplicar a programas que se ejecutan sobre una sola computadora es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

Ventajas

- Centralización del control: los accesos, recursos y la integridad de los datos son controlados por el servidor de forma que un programa cliente defectuoso o no autorizado no pueda dañar el

sistema. Esta centralización también facilita la tarea de poner al día datos u otros recursos (mejor que en las redes P2P[19]).

- Escalabilidad: se puede aumentar la capacidad de clientes y servidores por separado. Cualquier elemento puede ser aumentado (o mejorado) en cualquier momento, o se pueden añadir nuevos nodos a la red (clientes y/o servidores).
- Fácil mantenimiento: al estar distribuidas las funciones y responsabilidades entre varios ordenadores independientes, es posible reemplazar, reparar, actualizar, o incluso trasladar un servidor, mientras que sus clientes no se verán afectados por ese cambio (o se afectarán mínimamente). Esta independencia de los cambios también se conoce como encapsulación.
- Existen tecnologías, suficientemente desarrolladas, diseñadas para el paradigma de C/S que aseguran la seguridad en las transacciones, la amigabilidad del interfaz, y la facilidad de empleo.

Desventajas

- La congestión del tráfico ha sido siempre un problema en el paradigma de C/S. Cuando una gran cantidad de clientes envían peticiones simultaneas al mismo servidor, puede ser que cause muchos problemas para éste (a mayor número de clientes, más problemas para el servidor). Al contrario, en las redes P2P como cada nodo en la red hace también de servidor, cuantos más nodos hay, mejor es el ancho de banda que se tiene.
- El paradigma de C/S clásico no tiene la robustez de una red P2P. Cuando un servidor está caído, las peticiones de los clientes no pueden ser satisfechas. En la mayor parte de redes P2P, los recursos están generalmente distribuidos en varios nodos de la red. Aunque algunos salgan o abandonen la descarga; otros pueden todavía acabar de descargar consiguiendo datos del resto de los nodos en la red.
- El software y el hardware de un servidor son generalmente muy determinantes. Un hardware regular de un ordenador personal puede no poder servir a cierta cantidad de clientes.

[19] **P2P: (**Peer to Peer) Red de pares, son redes que funcionan sin clientes ni servidores fijos.

Normalmente se necesita software y hardware específico, sobre todo en el lado del servidor, para satisfacer el trabajo. Por supuesto, esto aumentará el coste.

- El cliente no dispone de los recursos que puedan existir en el servidor. Por ejemplo, si la aplicación es una Web, no podemos escribir en el disco duro del cliente o imprimir directamente sobre las impresoras sin sacar antes la ventana previa de impresión de los navegadores.

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.

Entre sus principales características se encuentran:

- 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 queremos ver que es lo que estamos instalando como servidor, lo podemos saber, 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 que los instalemos cuando los necesitemos. 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[20]. 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.

[20] **Logs:** Registro de errores.

(Ciberaula.com, 2008)

1.6.2.4 Frameworks

Un frameworks es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Los frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Algunos tipos frameworks desarrollados en PHP:

Un framework permite separar en capas la aplicación:

- La lógica de presentación que administra las interacciones entre el usuario y el software.
- La lógica de dominio o de negocio, que manipula los modelos de datos de acuerdo a los comandos recibidos desde la presentación.
- La lógica de datos que permite el acceso a un agente de almacenamiento persistente u otros.

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.

Presenta entre otras las siguientes características:

- Simplifica la gestión de archivos de configuración.
- Proporciona los componentes que forma la infraestructura del patrón Modelo-Vista-Controlador.

- Proporciona una capa de acceso a base de datos, construida sobre PDO[21] pero ampliándola con diferentes características.
- Cuenta con módulos para manejar archivos PDF, canales RSS, Web Services (Amazon, Flickr, Yahoo).
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Clientes para servicios web, incluidos Google Data APIs y StrikeIron. (Echarte, 2007)

En la siguiente figura se muestra la estructura de Zend Framework.

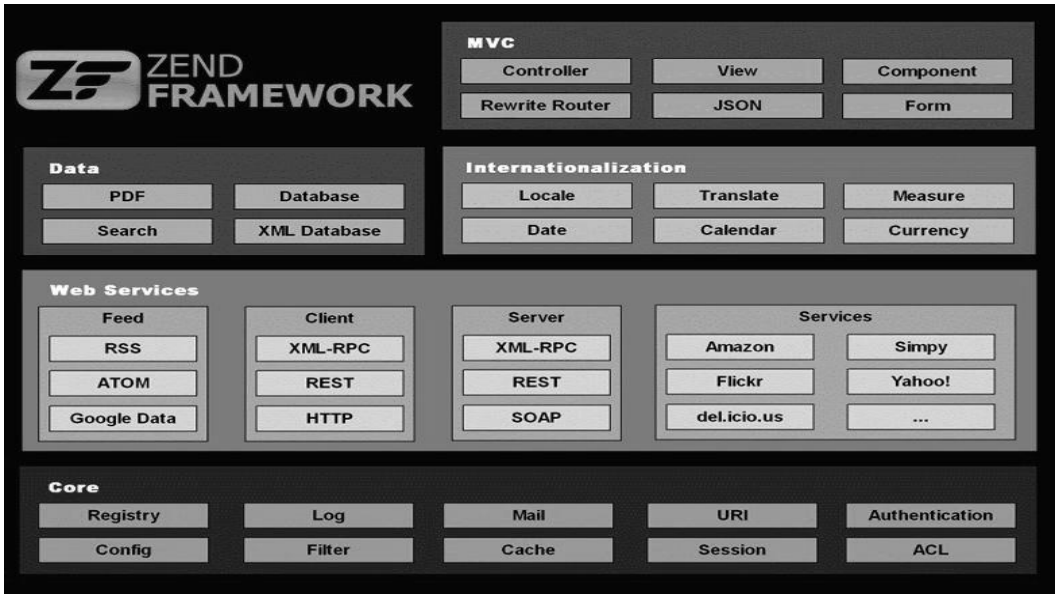

Figura 2 Estructura de Zend Framework

Zend_Ext Framework.

Es un framework open Source, que está diseñado para php 5 y buenas capacidades de ampliación. Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizada por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor, se le incluyo el IoC para la comunicación entre los

[21] **PDO:** PHP Data Objects (capa de abstracción de acceso a datos para PHP)

módulos o componentes. Se le incorporó la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos y el ExtJs Framework para el desarrollo de las vistas.

Doctrine Framework

Es un potente y completo sistema ORM (Object Relational Mapper) para PHP 5.2+ que incorpora una DBL (capa de abstracción a base de datos). Uno de sus rasgos importantes es la habilidad de escribir opcionalmente las preguntas de la base de datos orientado a objeto. Esto les proporciona una alternativa poderosa a diseñadores de SQL que mantiene un máximo de flexibilidad sin requerir la duplicación del código innecesario. También permite exportar una base de datos existente a sus clases correspondientes y también convierte clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos (Doctrine-Project.org, 2008).

En la siguiente figura se muestra la estructura de Doctrine ORM.

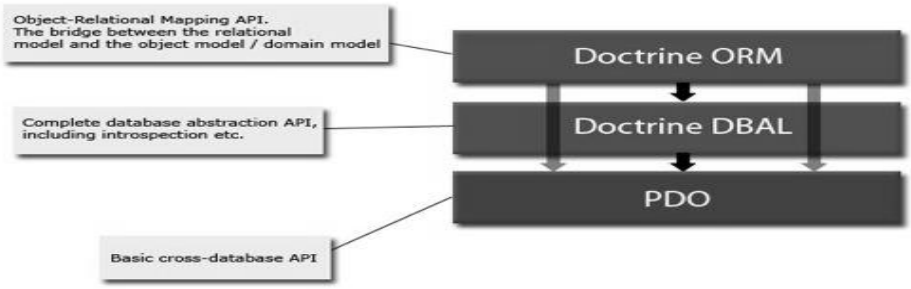

Figura 3 Doctrine ORM

ExtJs Framework

Es una librería construida con Java Script 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 haciendo uso extensivo de Ajax.

ExtJS 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 Element, contiene mucho de los métodos y propiedades de DOM que se necesitará proporcionando una interfaz conveniente, unificada y multinavegador).

Entre los componentes que esta librería ofrece encontramos cuadros de diálogo, menús, tablas editables, layouts, paneles, pestañas y todo lo necesario para construir atractivos desarrollos al estilo de Web 2.0.

Ventajas:

- La orientación a objetos intensa te hará modular todos tus scripts (por si es que a estas alturas ya no lo hacías).
- El diseño está completamente separado de la funcionalidad.
- Funciones comunes como validación, comboxes editables, ventanas arrastables (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 te perderás con muchos nuevos objetos en su extensa y bien documentada API (que por cierto también hace gala de sus mismas librerías). El tiempo de aprendizaje puede llegar a compararse con a aprender a programar en un lenguaje nuevo.
- Al estar todo tu sitio en JS, no podrá ser accesible para los buscadores, limitando su uso a sistemas y no sitios web.
- Si existiese algún objeto que desearas y no existiera, te verás en la compleja tarea de crear un nuevo objeto (sólo apto para programadores JS avanzados).

(Wordpress.com, 2007)

UCID Framework

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, el multilenguaje.

1.6.2.5 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.

1.6.2.5.1 Mozilla Firefox 3.0

Es el nuevo e innovador navegador open Source del que todo el mundo está hablando. Firefox ha sido creado por el proyecto Mozilla, un esfuerzo open Source sin ánimo de lucro que incluye a miles de voluntarios alrededor del mundo. La misión del proyecto Mozilla es preservar la elección y la innovación en Internet. El apoyo organizativo del proyecto Mozilla es proporcionado por Mozilla Foundation (en los Estados Unidos de América), Mozilla Europe y Mozilla Japón.

Por nombrar algunas de las posibilidades que ofrece Firefox y que no ofrece IE, están:

- Es Software libre.
- En Firefox no existen la cantidad de bugs que posee el catastrófico IE, inmediatamente se encuentra un bug en el producto es notificado al Proyecto Mozilla para que sea reparado el problema.
- Navegación por tabs: Esta es una de las principales características que tiene Firefox.
- También existen excelentes extensiones de fácil instalación, estos mejoran la usabilidad y el aspecto del navegador, cosa que no se logra en IE el cual siempre permanece con los mismos colores. (Firefox, 2009)

1.6.3 Control de Versiones

Se llama control de versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas

Un sistema de control de versiones debe proporcionar:

- 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 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, 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).

- Puede ser servido, mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversión en forma transparente.
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, LDAP, PAM, etc.)

Existen varias interfaces a Subversión, 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.
- Subclipse. "Plugin" que integra Subversión al entorno Eclipse.
- Subversive. "Plugin" alternativo para Eclipse.
- ViewVC. Interfaz web, que también trabaja delante de CVS.

Para Mac, pueden emplearse los interfaces SvnX, RapidSVN y Zigversion

(Osmosislatina.com, 2009)

1.7 Conclusiones Parciales

En el presente capítulo se ha tratado de forma general y resumida los diferentes sistemas que abarcan la gestión de entradas de AFT, determinando las ventajas, desventajas y factibilidad de su utilización en las entidades cubanas; además se realizó un estudio de las herramientas, tecnologías y la metodología propuestas por la dirección de arquitectura del proyecto, permitiendo sentar las bases para el desarrollo.

Capítulo 2: Descripción de la solución propuesta

2.1 Introducción

En este capítulo se realiza una profunda valoración de los artefactos propuestos por el analista de sistema exponiendo las principales ventajas y desventajas que ofreció la obtención de los mismos y se delimitan varios puntos importantes en el desarrollo del componente como:

- Se hace un análisis de posibles implementaciones, componentes o módulos ya existentes.
- Se describen las principales clases utilizadas en la implementación del mismo.
- Se describen los estándares de codificación para una mejor legibilidad del código.

2.2 Flujo de procesos

2.2.1 Valoración crítica de los artefactos propuestos por los analistas

La obtención de la especificación de requisitos propuestos por los analistas, resultó de gran importancia, pues permitió una mejor comprensión de los aspectos relacionados con los requisitos funcionales, componentes reutilizables. Se pretende mostrar cómo está aprovechada la arquitectura y las posibilidades que proporciona el marco de trabajo utilizado en la programación de la aplicación, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. Se muestra el diagrama de componentes y el diagrama de despliegue propuesto por el equipo de arquitectura.

Los requisitos funcionales descritos por los analistas y que posibilitaron la identificación de las clases y las funcionalidades a implementar son:

Componente Entrada

- RF 1: Realizar Apertura
- RF 2: Realizar Cierre de Apertura.
- RF 3: Gestionar documento.
    - RF 3.1: Adicionar documento.
    - RF 3.2: Modificar documento.

- RF 3.3: Eliminar documento.
- RF 3.4: Listar documento.
- RF 3.5: Confirmar documento.
- RF 3.6: Contabilizar documento.
- RF 3.7: Cancelar documento.
- RF 3.8: Imprimir documento.
- RF 3.9: Bbúsqueda avanzada de documento.
- RF 3.10: Consultar documento.

➢ RF 4: Gestionar AFT a documentos

- RF 4.1: Adicionar AFT nuevos a documentos.
- RF 4.2: Adicionar AFT existentes a documentos.
- RF 4.3: Modificar AFT en el documento.
- RF 4.4: Eliminar AFT del documento.
- RF 4.5: Buscar AFT.
- RF 4.6: Búsqueda avanzada de AFT.

➢ RF 5: Gestionar Movimientos.

- RF 5.1: Realizar Alta.
- RF 5.2: Realizar Ajuste por sobrante.

➢ RF 6: Gestionar Ubicación física.

- RF 6.1: Ubicar AFT.
- RF 6.2: Modificar Ubicación física de AFT

2.2.2 Objetos de automatización

Para realizar el proceso de entrada de AFT en las entidades, se recogen los datos de estos activos de forma manual. Estas actividades se harán de forma automatizada permitiendo gestionar la cantidad real de un AFT en las entidades.

2.2.3 Propuesta del sistema

Al iniciar la sesión el sistema verificará los privilegios que posee el usuario logueado, permitiendo acceder solo a las entidades y Subsistemas que tenga acceso, así como a las funcionalidades y las distintas acciones definidas según el rol que desempeña. Para poder llevar a cabo los procesos de Apertura, Alta, Ajuste por sobrante y Cierre de Apertura (procesos que abarca el componente Entrada) presentes en el Subsistema de AFT se muestra la funcionalidad Gestionar Documentos.

Si se selecciona esta funcionalidad en el sistema se carga un gestor de documentos donde se puede realizar las siguientes acciones:

- Permite gestionar (adicionar, modificar, eliminar, listar, confirmar, contabilizar, cancelar, imprimir, consultar) documentos (apertura y movimientos).
- Permite realizar búsquedas simples y avanzadas de documentos.
- Permite adicionar nuevos activos, darle una ubicación y además modificar esta ubicación.
- Permite eliminar activos de un documento.
- Permite modificar activos en un documento.
- Permite realizar búsquedas simples y avanzadas de activos fijos tangibles.

2.3 Estándares de codificación

2.3.1 Nomenclatura de las clases

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

Ejemplo: Gestionarentrada

2.3.2 Nomenclatura según el tipo de clases

Clases controladoras

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

Ejemplo: EntradaController

Clases de los modelos

Business

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

Ejemplo: EntradaModel

Domain

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

Ejemplo: DatMovmonedaaf

Generated

Las clases que se encuentran dentro de Generated el nombre comienza con el prefijo “Base” y seguido el nombre del dominio correspondiente.

Ejemplo: BaseDatMovmonedaaf

2.3.3 Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará una variante modificada de la notación CamelCasing*, y con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: modificaractivolote

En caso de ser una acción de la clase controladora se especifica el nombre de la acción todo en minúscula y seguida el sufijo”Action”.

Ejemplo: cargaraftsAction

2.3.4 Nomenclatura de las constantes

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

Ejemplo: ENTRADA.

2.3.5 Nomenclatura de las variables

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

Ejemplo: $arrMoneda

Prefijos para los tipos de datos

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

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
float	flt
Boolean	Boo

Tabla 1 Prefijos a utilizar en la creación de variables.

2.4 Tratamiento de errores

Para garantizar el correcto funcionamiento del sistema se debe tener en cuenta un tratamiento de errores, el sistema captura todas aquellas excepciones que son lanzadas y se le facilita un tratamiento para que el sistema no colapse.

De esta manera se da solución a las problemáticas de los lanzamientos de excepciones que se dan en el sistema según las peticiones del usuario, capturando los tipos de errores lanzados y mostrando de manera visible al usuario mensajes de confirmación, esto le dará una idea de qué es lo que está incorrecto y como solucionarlo.

Mediante la interfaz Web se impedirá que el usuario asuma un papel activo en la introducción de la información, para esto se contará con cuadros de opción, menú de selección, lo cual facilitará la entrada de datos. La información que requiera ser adicionada por el usuario se validará mediante funciones o expresiones regulares que garanticen que sea válida y que el cuadro de texto no esté vacío si es obligatorio llenarlo. Si hay un error en la información le saldrá al usuario un mensaje en pantalla indicándole el error, al oprimir Aceptar el mensaje desaparece y el usuario podrá seguir introduciendo los datos en el formulario. También se validarán las opciones correspondientes a la extracción o modificación de datos del servidor de base datos. Si se desea eliminar algún elemento de la BD se preguntará al usuario si está seguro de realizar dicha acción, al igual que cuando desee modificar alguna información, antes de actualizarla se le preguntará si desea realizarla o no.

En el sistema existen funciones que necesitan validaciones previas para que se ejecuten sin ningún problema, para estas validaciones se usan varios ficheros XML entre ellos el *validator*, el cual contiene las llamadas a las precondiciones y poscondiciones que deben cumplirse, igualmente esta el fichero *exception* el cual contiene los mensajes que serán mostrador según la excepción lanzada, otro de los ficheros XML es el *ManageException* el cual captura excepciones previas que definimos ya sean para excepciones del Doctrine o del Framework. Así se logra que se realicen las operaciones que se desean y que se rectifique al cometer un error.

2.5 Conclusiones Parciales

En este capítulo se especificó la propuesta del componente Entrada, se conoció el objeto de automatización así como el flujo de los procesos. Se hizo referencia a la descripción de las principales clases para facilitar la comprensión de la implementación realizada.

Por último se analizaron los estilos de código debido a que hacen más legible el programa fuente, con el objetivo de ayudar la comunicación entre desarrolladores.

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 por ejemplo 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, cuando estos no involucran vidas humanas, puesto que en este último caso el costo suele superar el 80% siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.

Las pruebas y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las mismas, es decir, la determinación de cuando se han realizado las suficientes pruebas. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

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. Además se utilizan métricas para la validación del modelo de diseño como son:

- Tamaño operacional de clase (TOC).
- Relaciones entre clases (RC).

3.2 Análisis de reutilización de componentes, código o módulos

A continuación se muestran los componentes que son reutilizados para la implementación del componente Entrada.

3.2.1 Componente Documento

Dentro de los componentes que se utilizan en la implementación del Componente de Entrada se encuentra el *Componente Documento* que tiene como función la de gestionar todos los documentos del Subsistema AFT. Este se realizó debido a que la mayoría de los módulos existentes en dicho Subsistema realizaban las mismas acciones pero con un comportamiento diferente, siendo este el padre de los documentos de Entrada. Cada módulo tiene su clase documento donde hereda de la clase documento modelo general, donde se encuentran todas las generalidades de dichos documentos (Apertura, Alta, Ajuste por sobrante) y se acceden a estos mediante llamadas por el Integrador (IoC) y un patrón proxy que realiza la función de interfaz entre el integrador y los documentos modelos de cada módulo.

3.2.2 Componente AFT

Dentro de los componentes que se utilizan en la implementación del componente Entrada se encuentra el *Componente AFT* que es el encargado de la gestión de los activos dentro de las entidades, ya sea los que están nomenclados, como los nuevos que se incluyen. Este tiene un funcionamiento parecido al Componente Documento en lo que respecta a estructura y el acceso al mismo, una clase AFT Modelo general donde heredan de él varias clases de diferentes módulos, y se acceden a estos mediante llamadas por el Integrador (IoC) y un patrón proxy que realiza la función de interfaz entre el integrador y los documentos modelos de cada módulo. Este brinda todos los servicios necesarios para operar sobre los activos, que a través de los movimientos se relacionan con los documentos.

3.2.3 Componente MovimientoAFT

Dentro de los componentes que se utilizan en la implementación del componente Entrada se encuentra el *Componente MovimientoAFT* que tiene una función muy importante, ya que es el encargado de la unión entre los documentos y los activos. Dicho componente facilita todas las operaciones que son necesarias para operar con los movimientos de ese documento. Un movimiento es una copia que se le hace al producto para no afectar los valores reales del mismo y una vez contabilizado el documento serán afectados los valores de producto con los del movimiento.

3.3 Integración entre componentes

3.3.1 Estrategias de integración

La aplicación está definida por 3 capas: capa de presentación (view), negocio (controller) y acceso a datos (models), esta arquitectura posibilita un trabajo seguro, rápido y eficiente. La integración vertical o llamada arquitectura en 3 capas consiste en el flujo de los datos desde la vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatro nodos de integración, el que encontramos 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. Todo el código dentro un mismo componente utiliza llamadas a métodos o eventos de forma directa. La comunicación entre diferentes módulos y componentes se realiza mediantes llamadas a la inversión de control. El IoC 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 registro de los datos de los módulos en un fichero XML que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre IoC y registra las funcionalidades que ofrecen los métodos de las clases control de los componentes del sistema. La base de datos es accedida de forma directa mediante controladoras y los componentes rehusados son integrados mediantes interfaces sencillas, garantizando así una total integración de las capas en el sistema.

3.3.2 Diagrama de componentes

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

Un usuario hace una solicitud mediante un evento del navegador en un formulario JS, este envía una notificación a la clase controladora, mediante JSON o Atributos. Esta setea entidades y se comunica con la clase gestora Bussines responsable de la petición, la cual ha debido instanciar. La gestora o Bussines es la encargada de la lógica propia del negocio, cálculos y procesamiento de negocio, y también puede comunicarse con la Doctrine Domain. Esta última se conecta mediante PDO (Doctrine Record) a la DBO

(Capa interna de doctrine basada en PDO nativo), y esta DBO es la que se conecta directamente a la base de datos.

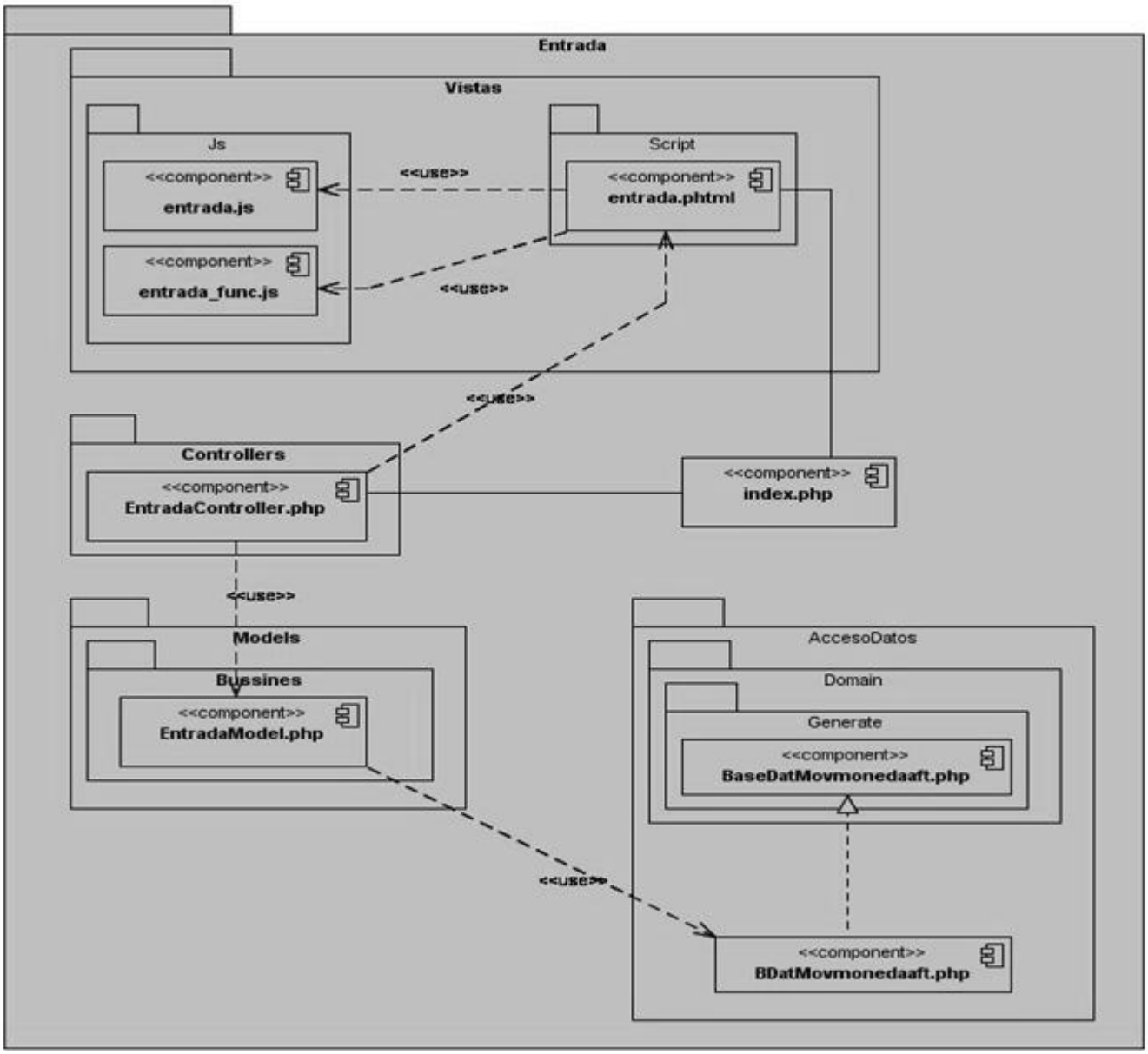

Figura 4 Diagrama de Componentes Vista Interna

3.3.3 Diagrama de integración de componentes

Aquí encontramos el componente Documento, que es el encargado de gestionar los documentos Entrada creados, así como las operaciones que se realicen sobre él. Por otra parte está el componente AFT, que es el que provee al componente Entrada los activos fijos tangibles que intervienen en este proceso. Por su parte, el componente MovimientoAFT es el encargado de registrar todas las operaciones que se realicen sobre los activos existentes, o por existir (guardar la asignación de los activos) en los documentos de Entrada.

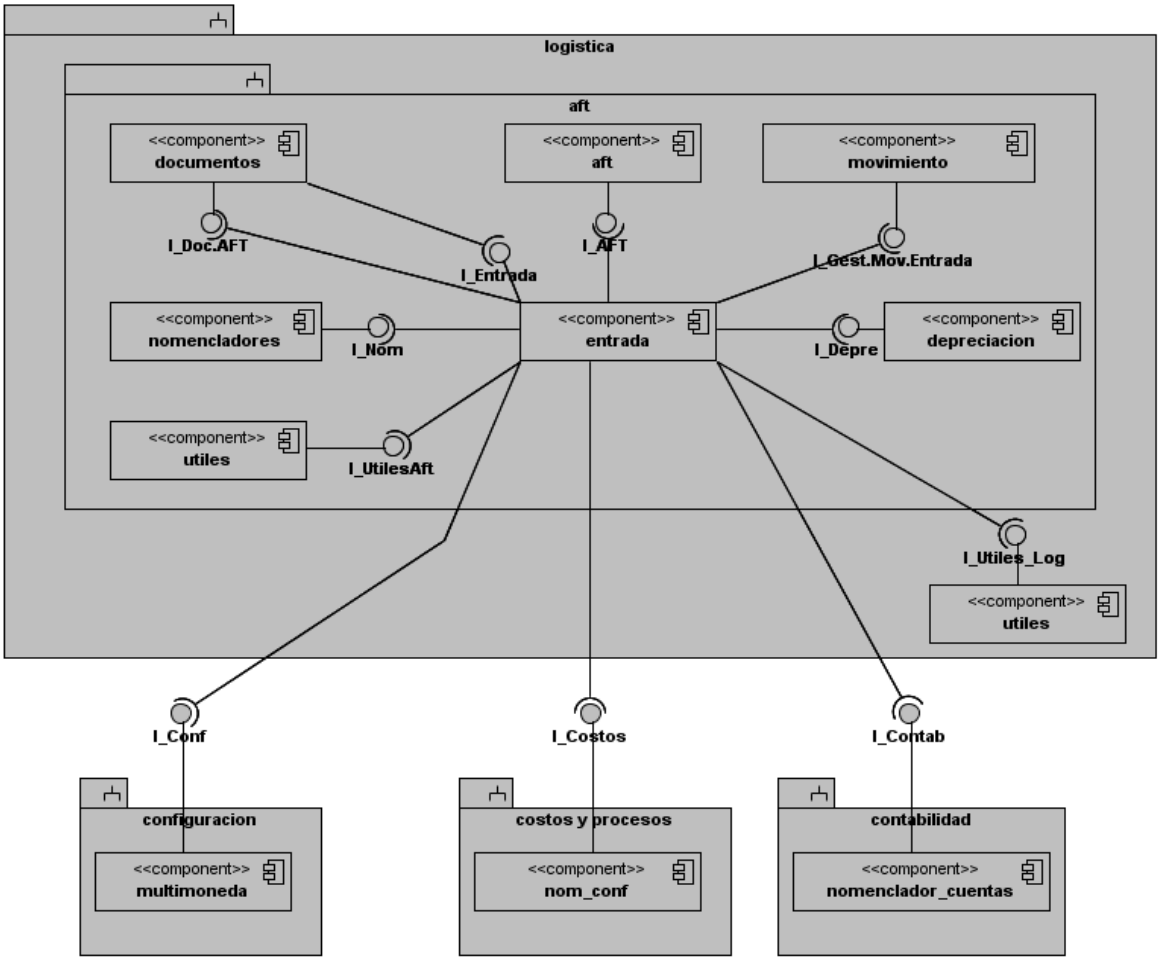

Figura 5 Diagrama de Componentes Vista externa

3.3.4 Matriz de Integración de Componentes Interna

	Componentes Internos	
Componentes Internos	**EntradaController.php**	**EntradaModel**
Entrada_func.js	adicionaraftAction() actualizarmonedaAction() registrarmonedaAction() adicionarcuentaegAction() adicionarmovimientoentradaaftAction() cuentasElemgastoAction() subsistemaAction() adicionarcuentaAction() confirmarAction() eliminarloteAction() eliminaractivoAction() eliminarrmonedaAction()	
Entrada.js	cargaraftsAction() cargarmonedasAction() monedasentidadAction() cargargrupossubgruposAction() cuentasElemgastoAction() cargarDatosAction()	
EntradaController.php		adicionarmovimientoentradaaft () cargaraftsAction() confirmarDocEntrada() adicionaraft() modificaractivolote() eliminarlote() cargarDatos() registrarmoneda() cargarmonedas() modificaractivolote()

Tabla 2 Matriz de Integración de Componentes Interna

3.3.5 Matriz de Integración de Componentes Externa

	Componentes externos						
Componentes Internos	**AFT**	**Nomen**	**MovEntAFT**	**DocAFT**	**Utiles**	**CostoP**	**Contab**
EntradaController.php	EC_AFT1 EC_AFT2 EC_AFT3 EC_AFT4 EC_AFT5	EC_NOM1 EC_NOM2 EC_NOM3 EC_NOM4 EC_NOM5	EC_MOV1 EC_MOV2 EC_MOV3 EC_MOV4 EC_MOV5 EC_MOV6 EC_MOV7 EC_MOV8 EC_MOV9	EC_DAFT1	EC_Utiles1 EC_Utiles1 EC_Utiles3	EC_CP1	EC_CONT1
EntradaModel.php	EM_AFT1 EM_AFT2 EM_AFT3 EM_AFT4 EM_AFT5 EM_AFT6 EM_AFT7	EM_NOM1	EM_MOV1 EM_MOV2 EM_MOV3 EM_MOV4 EM_MOV5 EM_MOV6	EM_ DAFT1 EM_ DAFT2 EM_ DAFT3 EM_ DAFT4 EM_ DAFT5 EM_ DAFT6 EM_ DAFT7:			

Tabla 3 Matriz de Integración de Componentes Interna

EC_AFT1: ObtenerGrupoPerteneceAFT()
EC_AFT2: ModificarNumInvAFT()
EC_AFT3: EliminarActivosFT()
EC_AFT4: DevolverAFTtxIdaft()
EC_AFT5: MostrarCmpNomAFT()
EC_NOM1: ObtenerAtributos()
EC_NOM2: BuscarValorAtributoActivo()
EC_NOM3: ModificarValorAtributoActivo
EC_NOM4: AdicionarValorAtributoActivo()
EC_NOM5: DevolverGruposubgrupos()
EC_Utiles1: ObtenerRegistNumerico()
EC_Utiles2: UbicarProducto()
EC_Utiles3: MostrarCuentas()

EC_MOV1: BuscarMovimientoActivo()
EC_MOV2: Buscarmovaftporid()
EC_MOV3: Buscarmovinventarioaft()
EC_MOV4: BuscarmovaftporDoc()
EC_MOV5: Buscarmovinventarioaft()
EC_MOV6: BuscarMovimientoEnearidadMonetaria()
EC_MOV7: CrearMovimientoEnearidadMonetaria()
EC_MOV8: ModificarMovimientoEnearidadMonetaria()
EC_MOV9: EliminarMovimientoEnearidadMonetaria()
EC_DAFT1: CargarDatosEncabezadoModificarAFT()
EC_CP1: obtenerCuentasPorAsoc()
EC_CONT1: ObtenerCuentaPorId()

EM_DAFT1: AdicionarDocAFT_Model()
EM_DAFT2: AdicionarDocAFT()
EM_DAFT3: ModificarDocAFT_Model()
EM_DAFT4: ModificarDocAFT()
EM_DAFT5: ContabilizarDocAFT()
EM_DAFT6: CancelarDocAFT()
EM_DAFT7: ConfirmarDocAFT()
EM_MOV1: BuscarMovimientosPorDocumentoAft()
EM_MOV2: BuscarMovimientoActivo()
EM_MOV3: EliminarMovimientoAft()
EM_MOV4: CrearMovimientoAft()
EM_MOV5: AdicionarMovimientoAft()
EM_MOV6: CrearMovimientoActivo()
EM_AFT1: EliminarActivosFT()
EM_AFT2: ObtenerActivoDadoNroInventario()
EM_AFT3: ObtenerAftDadoIdAFT()
EM_AFT4: ObtenerInstanciaDAFT()
EM_AFT5: InsertarActivosFT()
EM_AFT6: ModificarActivoFT()
EM_AFT7: ObtenerActivoDadoNroInventario()
EM_NOM1: ObtenerAtributos()

3.4 Pruebas de Software

3.4.1 Objetivos

El objetivo de la etapa de pruebas es garantizar la calidad del producto desarrollado. Esto implica:

- 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.

3.4.2 Diseño de casos de prueba

3.4.2.1 Condiciones de ejecución (Adicionar AFT)

- El usuario debe tener los permisos necesarios para realizar las operaciones.
- Deben haber documentos adicionados para poder insertar activos.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar AFT	Se insertan nuevos activos a partir de activos insertados previamente y además se insertan de forma dinámica atributos de estos activos.	EP 1.1: Adicionar activos correctamente.	– Se selecciona en la interfaz el botón Adicionar AFT. – El sistema permite seleccionar los posibles activos a insertar. – El sistema muestra un mensaje indicando que la operación se realizó de manera correcta. –
		EP 1.2: Adicionar activos presionando el botón Aplicar.	– Se selecciona en la interfaz el botón Adicionar AFT. – El sistema permite seleccionar los posibles activos a insertar. – El sistema muestra un mensaje indicando que la operación se realizó de manera correcta

			y mantiene la interfaz para insertar nuevos activos.
		EP 1.3: Cancelar operaciones.	– Se selecciona en la interfaz el botón Adicionar AFT. – El sistema muestra los activos a agregar. – El usuario selecciona o no los activos y presiona el botón Cancelar.
		EP 1.4: Llenar campos de los activos.	– El sistema muestra los campos a llenar en la ventana principal. – El usuario llena los datos correctamente. – El sistema muestra un mensaje indicando que la operación se realizó de manera correcta.

Tabla 4 Caso de Prueba Adicionar AFT

3.4.2.2 Condiciones de ejecución (Eliminar AFT)

- El usuario debe tener los permisos necesarios para realizar las operaciones.
- Deben haber documentos adicionados.
- Deben haber a activos insertados.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Eliminar AFT.	Se eliminan activos que existan en el documento.	EP 1.1: Eliminar activos.	– Se selecciona el activo que se desea eliminar. – Se presiona botón **Eliminar**. – El sistema muestra un mensaje de confirmación. – El usuario presiona el botón **Aceptar** del mensaje. – El sistema muestra un mensaje indicando que la operación se realizó de manera correcta y se actualiza el listado de activos.
		EP 1.2: Cancelar operaciones.	– Se selecciona el activo a eliminar. – El sistema muestra

			un mensaje de confirmación. – El usuario presiona el botón **Cancelar** del mensaje.

3.4.2.3 Condiciones de ejecución (Propiedades Dinámicas)

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Adicionar atributos dinámicos	Se insertan los atributos a partir de activos insertados previamente.	EP 1.1: Adicionar atributos dinámicos	– Se selecciona el grupo de activos a los cuales se les quiere poner los atributos. – Se selecciona en la interfaz el botón Propiedades Dinámicas. – El sistema muestra los campos a llenar. – El usuario llena los datos correctamente y presiona el botón Aceptar. – El sistema muestra un mensaje indicando que la operación se realizó de manera correcta.

		EP 1.2: Adicionar atributos dinámicos presionando el botón Aplicar.	– Se selecciona el grupo de activos a los cuales se les quiere poner los atributos. – Se selecciona en la interfaz el botón Propiedades Dinámicas. – El sistema muestra los campos a llenar. – El usuario llena los datos correctamente y presiona el botón Aplicar.. – El sistema muestra un mensaje indicando que la operación se realizó de manera correcta y se mantiene la interfaz para insertar nuevos atributos.

Tabla 5 Caso de Prueba Adicionar atributos dinámicos

Juego de datos a probar

Id del escenario	Escenario	Nombre	Cartucho	Color	SO	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar atributos dinamicos	Prueba	Si	Azul	Win	Valor(es) adicionado/modificado(s).	Valor(es) adicionado/modificado(s).
EP 1.2	Adicionar atributos dinamicos (Aplicar)	Prueba	No	Rojo	Lin	Valor(es) adicionado/modificado(s).	Valor(es) adicionado/modificado(s).

Tabla 6 Juego de datos para **Adicionar atributos dinámicos.**

3.5 Pruebas de unidad

Las pruebas unitarias son una forma de probar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

Fomentan el cambio: facilitan que el programador cambie el código para mejorar su estructura, puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.

Simplifica la integración: permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente.

Documenta el código: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.

Separación de la interfaz y la implementación: Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro.

Los errores están más acotados y son más fáciles de localizar: dado que tenemos pruebas unitarias que pueden desenmascararlos.

3.6 Pruebas de caja blanca

Las pruebas de caja blanca se realizan sobre las funciones internas de un módulo en concreto, están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos, pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos, comprobación de bucles.

En la siguiente tabla se representa las pruebas de Caja Blanca.

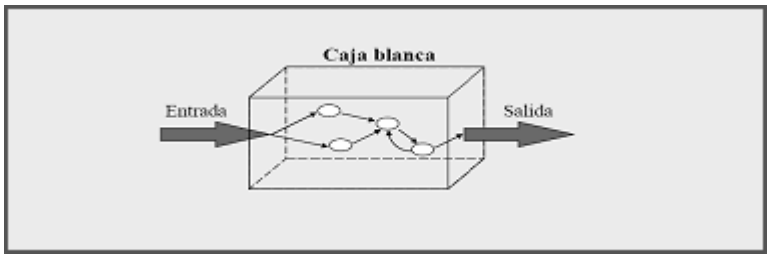

Figura 6 Representación de pruebas de Caja Blanca.

Tipos de prueba de caja blanca:

Prueba de Condición: Es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de Flujo de Datos: Se selecciona caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de Bucles: Es una técnica de prueba de caja blanca que se centra exclusivamente en la validez de las construcciones de bucles.

Prueba del Camino Básico: Esta técnica permite obtener una medida de la complejidad lógica de un diseño y usar esta medida como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el Grafo de Flujo asociado y se calcula su complejidad ciclomática.

La técnica del camino básico permite obtener una medida de la complejidad lógica del código de cada método, programa o módulo dado. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes que existen en la codificación por los cuales puede circular el flujo de control. Es además una de las más eficientes en cuanto a cobertura de código, pues logra que se ejecuten todos los bucles en sus límites operacionales (Márquez Alpízar, 2008).

Los pasos que se siguen para aplicar esta técnica son:

- A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
- Se calcula la complejidad ciclomática del grafo.
- Se determina un conjunto básico de caminos independientes.
- Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba obtenidos del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Para aplicar la técnica del camino básico se debe introducir la notación para la representación del flujo de control, este puede representarse por un grafo de flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.
- Todo segmento de código de cualquier programa se puede traducir a un grafo de flujo.
- Se calcula la complejidad ciclomática del grafo.

Para construir el grafo se debe tener en cuenta la notación para las instrucciones.

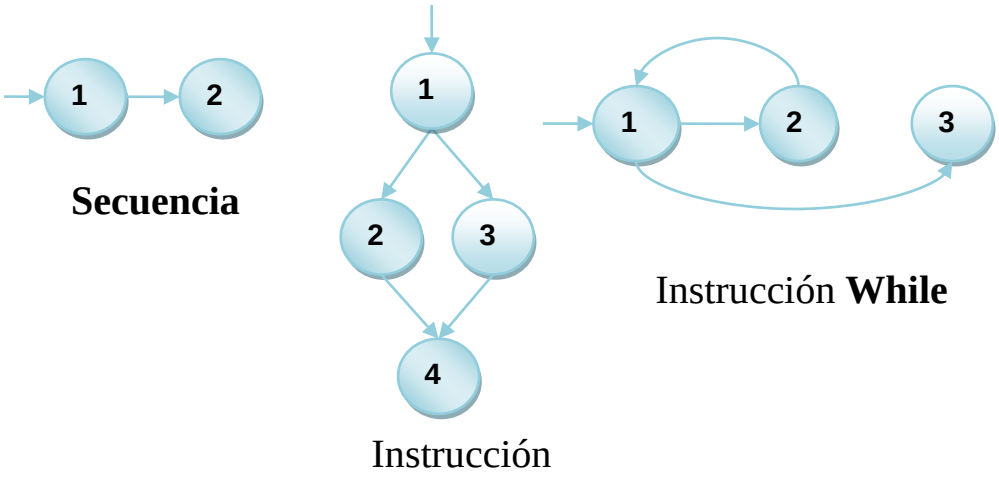

Figura 7 Notación de grafos de flujo para las instrucciones: Secuenciales, If, While.

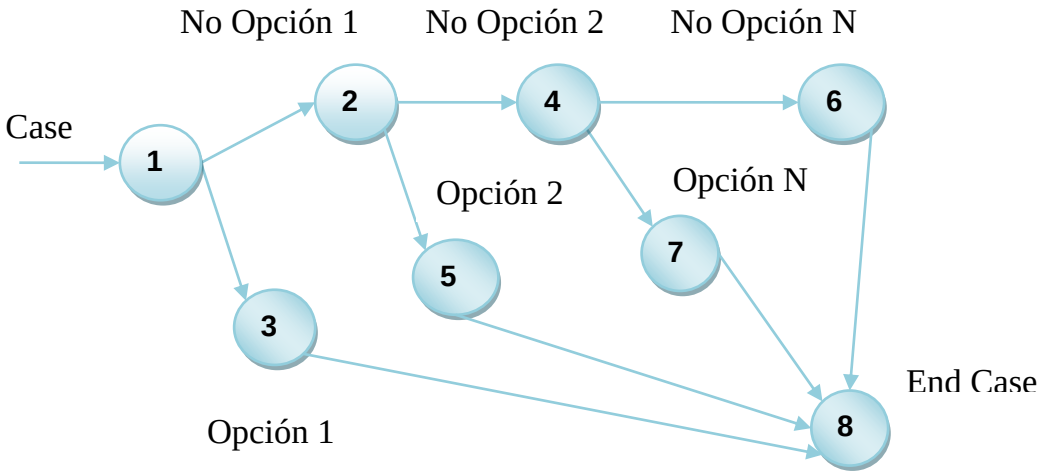

Figura 8 Notación de grafos de flujo para la instrucción Case.

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: son los círculos representados en el grafo de flujo, el cual representa una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.

Aristas: son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.

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

Para realizar la prueba de caja blanca específicamente la prueba del camino básico es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumera las sentencias de código del procedimiento realizado sobre el método cargarmonedas($datos) el cual se encarga de cargar las monedas soportadas por el movimiento de Entradas en cuestión.

```
public function adicionarmovimientoentradaaft($iddocumentoaft,$arearesponsabilidad,$arrAft,$cronologia)
    {
      $codMsg=1; (1)
      $mensaje='Activos adicionados satisfactoriamenete'; (1)
      foreach ( $arrAft as $value ) (2)
            {

                $movimiento = $this->pIntegrator->movimientosEntradaAft->crearMovimientoAft (); (3)
                $movimiento->iddocumentoaft = $iddocumentoaft; (3)
                $movimiento->cronologia = $cronologia; (3)
                $movimiento->idactivofijotang = $value->idactivofijotang; (3)
                $movimiento->descripcion = $value->descripcion; (3)

                if($value->vidautil) (4)
                $movimiento->vidautil = $value->vidautil; (5)

                $mov = $this->pIntegrator->movimientosEntradaAft->adicionarMovimientoAft ( $movimiento ); (6)

               if(!$mov) (7)
                {
                  $codMsg=3; (8)
                  $mensaje='No se pudo insertar el activo en el documento'; (8)
                } (9)

            } (10)
        $resultado['codMsg']=$codMsg; (11)
        $resultado['mensaje']=$mensaje; (11)
        return $resultado; (11)
    }
```

Figura 9 Representación del algoritmo adicionarmovimientoentradaaft($iddocumentoaft,$arearesponsabilidad,$arrAft,$cronologia).

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

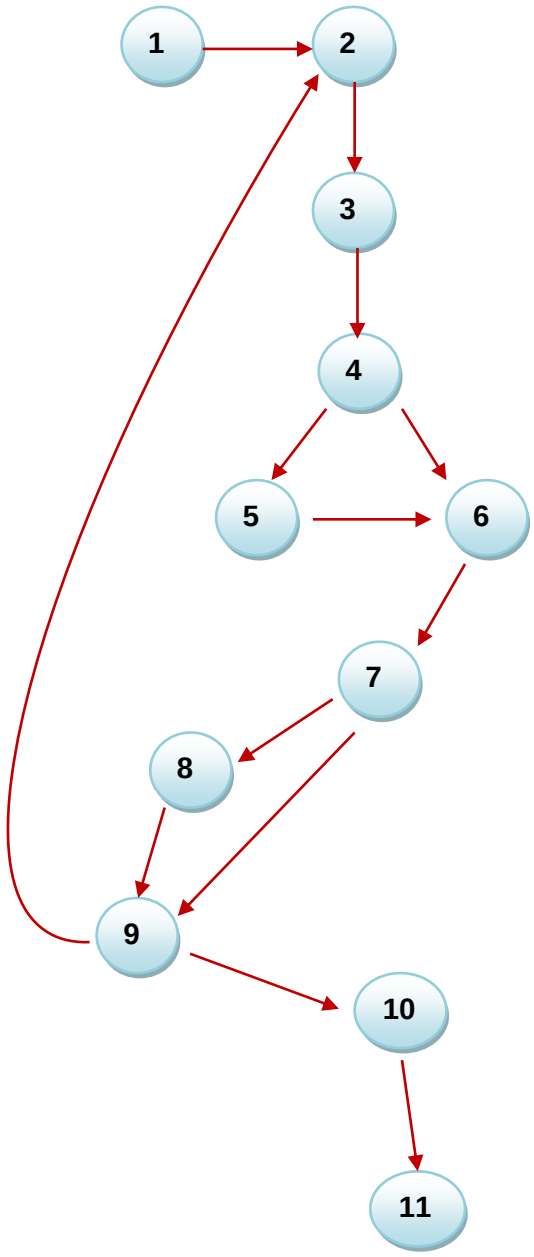

Figura 10 Grafo de flujo asociado al algoritmo adicionarmovimientoentradaaft($iddocumentoaft,$arearesponsabilidad,$arrAft,$cronologia).

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 efectuar el cálculo de la complejidad ciclomática del código es necesario tener varios parámetros como son la cantidad total de aristas del grafo, cantidad total de nodos para la siguiente fórmula:

V (G) = (A– N) + 2

V (G) = (13 – 11) + 2

V (G) = 4

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

Se puede usar también:

V (G) = P + 1

V (G) = 3 + 1

V (G) = 4

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

V (G) = R

V (G) = 4

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

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, por lo que se puede plantear que la complejidad ciclomática del código es de 4, lo que significa que existen tres posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

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

En la siguiente tabla se muestran los caminos básicos.

Número	Camino básico
1	1– 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11
2	1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 11
3	1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 11
4	1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11

Tabla 7 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.

Para realizarlos es necesario cumplir con las siguientes exigencias:

- Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- Entrada: Se muestran los parámetros que entran al procedimiento.
- Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico 1:

Camino 1: [1– 2 – 3 – 4 – 6 – 7 – 8 – 9 – 10 – 11]

Descripción:

- Los datos de entrada cumplirán con los siguientes requisitos:

El parámetro arrAft será un arreglo vacio, el arearesponsabilidad tendra valor de 1, 2 ó 3, los valores de iddocumentoaft y cronologia serán constantes para todas las pruebas del camino básico

Condición de ejecución:

- El arrAft será un arreglo vacio.
- El área de responsabilidad será 2,
- El iddocumentoaft tendrá valor 90000000008.

Entrada:

- $ arrAft = array ().
- $ arearesponsabilidad = 2.
- $iddocumentoaft = 90000000008.

Resultados esperados:

Se espera que no se inserte ya que el arreglo que debe contener los activos esta vacio.

Caso de prueba para el camino básico 2:

Camino 1: [1 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 2 – 3 – 4 – 5 – 6 – 7 – 9 – 10 – 11]

Descripción:

- Los datos de entrada cumplirán con los siguientes requisitos:

  El parámetro arrAft será un arreglo con un activo, el arearesponsabilidad tendra valor de 1, 2 ó 3, los valores de iddocumentoaft y cronologia serán constantes para todas las pruebas del camino básico

Condición de ejecución:

- El arrAft será un arreglo con un activo.
- El área de responsabilidad será 1,
- El iddocumentoaft tendrá valor 90000000008.

Entrada:

- $ arrAft = array (1 producto).
- $ arearesponsabilidad = 1.
- $iddocumento = 90000000008.

Resultados esperados:

Se espera que se lance un mensaje con el siguiente texto: “Activos adicionados satisfactoriamente”.

Caso de prueba para el camino básico 3

Camino 3: [1 – 2 – 6 – 7]

Descripción:

- Los datos de entrada cumplirán con los siguientes requisitos:

  El parámetro arrAft será un arreglo con un activo, el arearesponsabilidad tendra valor 4, los valores de iddocumentoaft y cronologia serán constantes para todas las pruebas del camino básico

Condición de ejecución:

- El arrAft será un arreglo con un activo.
- El área de responsabilidad será 4,
- El iddocumentoaft tendrá valor 90000000008.
- .

Entrada:

- $ arrAft = array (1 producto).
- $ arearesponsabilidad = 4.
- $iddocumento = 90000000008.

Resultados esperados:

Se espera que se lance un mensaje con el siguiente texto: “No se pudo insertar el activo en el documento”.

Luego de aplicar los distintos casos de pruebas, se pudo comprobar que el flujo de trabajo de la función está correcto ya que cumple con las condiciones necesarias que se habían planteado.

3.7 Conclusiones parciales

En el presente capítulo se explicaron los componentes que fueron reutilizados para el desarrollo de la aplicación. Además se conoció todo lo referente a la integración de ellos mediante el IoC y la arquitectura en capas utilizada.

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.

Finalmente se elaboraron instrumentos inspirados en métricas para calidad del diseño como el tamaño operacional de clase (TOC) donde mediante su empleo permitió afirmar que el diseño realizado se puede valorar de aceptable debido a que el valor de los atributos de calidad, responsabilidad, complejidad de implementación, estuvo por encima del 62% en todos los casos.

Conclusiones generales

A manera de conclusión se plantea:

- Se analizaron soluciones existentes descubriendo deficiencias en los mismos.
- Se mostraron los aspectos más significativos de la solución propuesta como el análisis de reutilización de componentes, la descripción de las principales clases a utilizar, la descripción de la implementación y los estándares de codificación utilizados evidenciado la obtención de un producto funcional a partir de los requisitos propuestos por los analistas.
- Se realizó la validación de la solución propuesta mediante el diseño y aplicación de las pruebas de caja blanca para validar la calidad de la solución propuesta arrojando resultados positivos. También se hizo una evaluación de la implementación mediantes métricas donde arrojaron valores aceptables en los atributos de reutilización, facilidad de mantenimiento, complejidad del diseño, complejidad de implementación, cohesión, acoplamiento y cantidad de pruebas.

Por todo lo antes mencionado se evidencia el cumplimiento de los objetivos propuestos en el presente trabajo de diploma, lo cual conlleva a un cumplimiento del objetivo general.

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 módulo con los nuevos requerimientos que surjan por necesidades del cliente.
- Poner al alcance de todos, este documento, como material de estudio, guía y apoyo para su posterior continuación.