UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Desarrollo de los procesos de Apertura y Cierre de Apertura, Alta y Baja en el Subsistema Activo Fijo Intangible del Sistema Cedrux.

Autores:

Javier Santiago Rodríguez

Carlos Miguel Morera Pérez

Tutores:

Ing. Osnier Ramírez Alea

Ing. Noidis Barroso Hidalgo

La Habana, Junio de 2014.

Año 56 de la Revolución.

"No se vive celebrando victorias, sino superando derrotas."

Ernesto "Che" Guevara de la Serna

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de_____ del año_____.

_____	_____
Javier Santiago Rodríguez	Carlos Miguel Morera Pérez
(Autor)	(Autor)
_____	_____
Ing. Osnier Ramírez Alea	Ing. Noidis Barroso Hidalgo
(Tutor)	(Tutora)

Datos del contacto

Datos del Tutor(es):

Ing. Osnier Ramírez Alea

Ingeniero en Ciencias Informáticas. Graduado en la Universidad de las Ciencias Informáticas en el 2009. Actualmente trabaja como desarrollador en la línea Logística del Centro de Informatización para la Gestión de Entidades, Facultad 3. Correo: oramirez@uci.cu

Ing. Noidis Barroso Hidalgo

Ingeniero en Ciencias Informáticas. Graduada en la Universidad de las Ciencias Informáticas en el 2010. Actualmente trabaja como analista en la línea Logística del Centro de Informatización para la Gestión de Entidades, Facultad 3. Correo: nbarroso@uci.cu

Agradecimientos

Javier Santiago Rodríguez

A mis padres y hermanos por apoyarme y motivarme siempre.

A mis abuelos por ser mi fuente de inspiración.

A mis tíos y primos por apoyarme.

A mis amigos y compañeros por siempre estar a mi lado en los buenos y malos momentos.

A mis tutores y todos los que me ayudaron en el proyecto.

A mi compañero de Tesis.

A mi novia por ser parte de mi vida y apoyarme y quererme incondicionalmente.

¡Muchas Gracias!

Agradecimientos

Carlos Miguel Morera Pérez

A mis padres por darme tanto amor, confianza y apoyo en toda mi vida y principalmente en estos 5 años de carrera.

A mi abuela Nene por cuidarme y preocuparse por mí desde que nací.

A mis abuelos, hermano, tíos, primos y resto de la familia por ser fuente de inspiración y estar siempre pendiente de mí.

A mis amigos y compañeros por siempre estar a mi lado en los buenos y malos momentos.

A mis tutores y todos los que me ayudaron en el proyecto.

A mi compañero de tesis.

¡Muchas Gracias!

Dedicatoria

Javier Santiago Rodríguez

A mis padres y a mi familia en general.

A todos los que me apoyaron durante estos 5 años.

Carlos Miguel Morera Pérez

A mis padres, mi hermano y a mi familia en general.

A todos los que me apoyaron durante estos 5 años.

Resumen

Los Activos Fijos Intangibles se caracterizan por ser activos de carácter no monetario y sin apariencia física, estos se adquieren para ser utilizados en la producción o suministro de bienes y servicios o para funciones relacionadas con la administración de las entidades. Su objetivo fundamental es la obtención de ganancias. Estos activos han tenido gran auge en los últimos años en el escenario empresarial, trayendo consigo factores que dan poder y ventaja competitiva a las empresas. En las entidades se realizan una serie de procesos, entre los cuales se encuentran Apertura y cierre de apertura, Alta, Baja y Amortización de los Activos Fijos Intangibles, entre otros. Esta investigación se centra en los tres primeros procesos. Como consecuencia de la importancia que tiene la gestión de los Activos Fijos Intangibles, en la Universidad de la Ciencias Informáticas, específicamente el Centro de Informatización para la Gestión de Entidades, se trazó la meta de incluir dentro de su producto principal Cedrux el subsistema de Activos Fijos Intangibles.

Para guiar el desarrollo de la solución se utilizó el modelo de desarrollo del Centro de Informatización para la Gestión de Entidades, el cual brinda un conjunto de fases y disciplinas a transitar, así como los artefactos a realizar en cada una de ellas. Se definieron los procesos de negocio y los requisitos, los cuales permitieron obtener la información requerida para realizar el diseño, sentando las bases para la implementación de la solución propuesta. Esta solución fue probada mediante las pruebas internas definidas en la investigación.

Palabras claves: Activos Fijos Intangibles, Cedrux, procesos.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

En los últimos años surge un nuevo escenario empresarial donde la riqueza no se basa solamente en los bienes materiales, apareciendo otros factores que dan poder y ventaja competitiva a las empresas y que se agrupan bajo la denominación de Activos Fijos Intangibles (AFI) (1). Tomando como referencia la Norma Internacional de Contabilidad NIC 38, “un activo intangible es un activo identificable, de carácter no monetario y sin apariencia física, que se posee para ser utilizado en la producción o suministro de bienes y servicios, para ser arrendado a terceros o para funciones relacionadas con la administración de la entidad.” (2)

En Cuba la mayoría de las entidades han pasado por alto la explotación de los AFI, limitándose solamente a su registro y protección legal, tratándolos erróneamente como gastos y carentes de beneficios. Sin embargo estos expresan un valor y su propósito es la obtención de ganancias. Con los AFI se puede realizar una serie de acciones para su gestión y control. En la presente investigación se hizo énfasis en los procesos de Apertura y cierre de apertura, Alta y Baja de los AFI, los cuales se dividen en procesos de entrada y no entrada. La ejecución de dichos procesos se realiza mediante documentos en los que se registran los activos, permitiendo llevar un seguimiento de los AFI durante su transcurso en la entidad.

La gestión de los AFI genera grandes volúmenes de información debido a que por cada operación que se le realice a estos se genera un documento, de esta forma se hace difícil mantener el control sobre el registro de los mismos. Esta información que es recogida puede encontrase de forma no centralizada, lo cual trae consigo que la búsqueda de documentación sea engorrosa y que en el peor de los casos se extravíe algún documento. Estas dificultades afectarían la trazabilidad de los AFI, provocando que no se tenga conocimiento de su origen o destino, de los movimientos que sobre ellos se realizan y del beneficio que pudieran brindar para la entidad que los posea.

Como consecuencia de la importancia que tiene la gestión de los AFI, en la Universidad de la Ciencias Informáticas (UCI), específicamente el Centro de Informatización para la Gestión de Entidades (CEIGE), se trazó la meta de incluir dentro de su producto principal Cedrux el subsistema de AFI. Para ello se hizo necesario desarrollar los componentes Documento, Movimiento, Activos Fijos Intangibles, Entrada y No Entrada, ya que los mismos constituyen la base para el desarrollo de los procesos de los AFI.

La situación anteriormente descrita, deriva el siguiente **problema a resolver:** ¿Cómo contribuir a la informatización de los procesos del Subsistema de Activos Fijos Intangibles para el Sistema Integral de Gestión Cedrux?

En consecuencia se define como **objeto de estudio** los procesos de los Activos Fijos Intangibles, enmarcando como **campo de acción** los sistemas informáticos que gestionan los procesos de los Activos Fijos Intangibles en Cuba.

Para dar solución al problema planteado se define como **objetivo general:** Desarrollar los procesos de Apertura y Cierre de Apertura, Alta y Baja, para contribuir a la informatización de los procesos del Subsistema de Activos Fijos Intangibles para el Sistema Cedrux.

El objetivo general se desglosa en los siguientes **objetivos específicos**:

- Elaborar la fundamentación teórica de la investigación a partir del estudio de los procesos de los AFI.
- Realizar el modelado de negocio y descripción de requisitos.
- Realizar el análisis y diseño a partir de los artefactos generados en las disciplinas modelado de negocio y requisitos.
- Realizar la implementación de la solución propuesta a partir de los artefactos generados en el diseño.
- Validar la solución propuesta mediante pruebas de caja blanca a partir de la técnica Camino Básico y pruebas de caja negra a partir de la técnica Partición de equivalencia.

Con el cumplimiento de los objetivos se propone como **idea a defender**: Con el desarrollo de los procesos de Apertura y Cierre de Apertura, Alta y Baja, se contribuirá a la informatización de los procesos del Subsistema de Activos Fijos Intangibles para el Sistema Cedrux.

La investigación está sustentada en la utilización de un conjunto de métodos de carácter teórico y empírico, entre los que se encuentran:

Métodos teóricos:

- Analítico-Sintético: Permitió realizar un estudio de los procesos de los AFI, analizar las diferentes fuentes bibliográficas para el estudio de las herramientas y modelo de desarrollo de software empleados para dar cumplimiento a la solución propuesta.
- Modelación: Se evidenció en el Modelo Conceptual y en los Diagramas de Procesos de Negocio, ya que permitieron crear abstracciones de la realidad a través de modelos.

Métodos empíricos:

- Entrevista no estructurada: Permitió obtener información sobre los procesos de los AFI y la forma en que estos se pueden gestionar a través de una herramienta informática para analizar su funcionamiento.

El trabajo está estructurado en tres capítulos:

Capítulo 1. Fundamentación teórica: En este capítulo se analizan varios conceptos asociados a los AFI. Además de estudiar algunos aspectos teóricos que dan sustento a la investigación, tales como: algunos sistemas informáticos vinculados a la gestión de los AFI, el modelo de desarrollo de software definido por el Departamento de Tecnología del centro CEIGE, así como, las herramientas y tecnologías a utilizar para el cumplimiento de la solución.

Capítulo 2. Análisis y Diseño: En este capítulo se realizarán las disciplinas Modelado de negocio, Requisitos, Análisis y Diseño de la solución propuesta, donde se obtienen artefactos como: el modelo conceptual, mapa de procesos, descripción de los requisitos funcionales y no funcionales, el diagrama de clases del diseño, entre otros.

Capítulo 3. Implementación y pruebas: En este capítulo se especifican aspectos de interés para el proceso de implementación y validación. En la validación se comprueba el correcto funcionamiento de la solución.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

Introducción

El presente capítulo expone los principales conceptos relacionados con la investigación, en él se hace un análisis de soluciones existentes tanto nacionales como extranjeras que gestionan los procesos de los AFI. Además, se describen las herramientas y tecnologías a tener en cuenta para el desarrollo de la solución.

1.1 Conceptos fundamentales

1.1.1 Activo Fijo Intangible

La Norma Internacional de Contabilidad NIC 38 define varios aspectos importantes relacionados con los Activos Fijos Intangibles que permitirán comprender la investigación. A continuación se enuncian algunos de ellos:

“Un activo es un recurso controlado por la entidad como resultado de sucesos pasados; y del que la entidad espera obtener, en el futuro, beneficios económicos.” (2)

“Se puede reconocer un activo intangible si cumple las dos características siguientes: El costo del activo puede ser medido en forma fiable. Se espera obtener, en el futuro, beneficios económicos para la entidad.” (2)

“Un activo intangible es un activo identificable, de carácter no monetario y sin apariencia física, que se posee para ser utilizado en la producción o suministro de bienes y servicios, para ser arrendado a terceros o para funciones relacionadas con la administración de la entidad.” (2)

La NIC 38 también especifica que la empresa debe evaluar la probabilidad de obtener beneficios futuros utilizando hipótesis razonables y fundadas, que represente las mejores estimaciones de la gerencia respecto al conjunto de condiciones económicas que se darán a lo largo de la vida útil del activo.

Según la Resolución Técnica Nº 9 de la Federación Argentina de Consejos Profesionales en Ciencias Económicas (F.A.C.P.C.E) define a los activos intangibles como” (…) aquellos representativos de franquicias, privilegios u otros similares, incluyendo los anticipos por su adquisición, que no son bienes tangibles ni derechos contra terceros, y que expresan un valor cuya existencia depende de la posibilidad futura de producir ingresos.” (3)

En otras bibliografías se plantea que:

"Un activo fijo intangible es el que no tiene existencia física, cuyo valor radica en los derechos conferidos como resultado de la titularidad y propiedad de los mismos." (4)

"Los activos intangibles, son los activos que complementan a los tangibles para lograr una medición más correcta del valor total de una organización. Tienen una capacidad de servicio que no se agota ni se consume con su primer empleo sino a lo largo del tiempo; mientras están en uso o se licencian no se transforman en otros bienes ni están destinados a la venta." (5)

Los autores de la presente investigación toman como referencia el tercer concepto definido por la Norma Internacional de Contabilidad NIC 38 enunciado en este sub-epígrafe, debido a que abarca un grupo de características que describe a los AFI y se enmarca en las funciones del mismo dentro de las entidades.

1.1.2 Los procesos de los Activos Fijos Intangibles

La Real Academia Española (RAE), define como proceso el *"conjunto de las fases sucesivas de un fenómeno natural o de una operación artificial". (6)*

Por lo tanto, se establece como concepto de los procesos de los Activos Fijos Intangibles, el conjunto de acciones que se realizan con los Activos Fijos Intangibles, desde su llegada a las entidades y hasta que deja de aportar beneficios a la misma.

1.2 Sistemas que gestionan los procesos de los Activos Fijos Intangibles

Son muchos los sistemas para la gestión que se utilizan en el mundo, pero la mayoría de ellos han sido desarrollados sobre plataformas propietarias y aplicaciones de escritorio. A continuación se mostrará una descripción de algunos de los principales sistemas extranjeros y usados en Cuba que incluyen dentro de sus funcionalidades la gestión de los Activos Fijos Intangibles.

1.2.1 Sistemas extranjeros

Openbravo ERP

Es un ERP basado en aplicación web como solución de negocio para la pequeña y mediana empresa. Openbravo ERP consta de dos versiones: Openbravo Community Edition (libre y gratuita) con soporte y funciones limitadas, así como actualizaciones restringidas y sin garantía de corrección de fallos y Openbravo Network Edition (con elementos privativos y comerciales) que requiere la compra de una licencia, esta versión soportada provee actualizaciones de código, funcionalidad, incluye los módulos comerciales (no incluidos en la versión libre) y soporte directo. (7)

Los módulos principales que presenta Openbravo son:

- Gestión de datos maestros.
- Gestión de aprovisionamiento.
- Gestión de almacenes.
- Gestión de proyectos y servicios.
- Gestión de la producción.
- Gestión comercial y de las relaciones con los clientes.
- Gestión financiera y contable de la empresa.
- Además de contar con un sistema de producción y planificación y una sección de Business Intelligence.

Dentro del módulo de Gestión financiera y contable de la empresa, se encuentra la gestión de activos fijos, donde se incluyen tanto los activos fijos tangibles como los intangibles. Permite llevar un control detallado de los procesos de amortización y depreciación de sus activos. (8)

SAP

SAP es uno de los ERP más utilizados mundialmente. Tiene también una gran incursión en el mercado de empresas pequeñas y medianas con una solución de *software-as-a-service* (software como servicio). (9)

Módulos principales del sistema SAP: Finanzas (FI), Controlling (CO), Gestión de materiales (MM), Ventas y distribución (SD), Recursos Humanos (HR), Business Intelligence (BI), ABAP.

El módulo Finanzas satisface todas las necesidades internacionales que debe cumplir el departamento de gestión financiera de una empresa. Algunas de sus funcionalidades son las siguientes:

Gestión y representación de todos los datos de contabilidad, según el principio del registro por documentos. Disponibilidad de los datos en tiempo real y sincronización de las cuentas auxiliares con la contabilidad del libro Mayor. La integración del módulo finanzas con otros módulos asegura que se reflejen exactamente los movimientos logísticos de mercancías (tales como las entradas y salidas de mercaderías) en las actualizaciones de contabilidad basadas en valor. Entre sus subcomponentes se encuentra el de Activos fijos (FI-AA). (9)

El sistema permite gestionar todas las operaciones relacionadas con los activos fijos, entre las que se encuentran: Alta de Activos Fijos, Baja de Activos Fijos, Venta de Activos Fijos, Amortización de Activos, Traslados de Activos, entre otras. Abarcando toda la vida del activo desde el pedido o alta inicial hasta que se le da de baja de la entidad. (9)

1.2.2 Sistemas usados en Cuba

ASSETS NS

Es un Sistema de Gestión Integral estándar y parametrizado que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles y Herramientas y Recursos Humanos. Dispone, además, de métodos novedosos para administración y planificación de inventarios, así como una amplia gama de análisis y consultas. (10)

Características de ASSETS NS:

- Genera, automáticamente, los asientos de diario a la contabilidad por cada una de las transacciones contempladas en el sistema.
- Es una aplicación cliente-servidor programada en Visual Basic 6.0 y Microsoft SQL[1] Server 2000, utilizando adicionalmente Crystal Reports 7.0 para la generación de reportes de salidas.
- Todos los procesos están implementados con inicio y final de transacciones, lo que garantiza la integridad de la base de datos ante fallos de corriente, cambios de voltaje o cualquier otra eventualidad que provoque una falla en la operación del sistema.
- Al ser una aplicación cliente-servidor realiza una manipulación mínima de datos en el sitio del cliente, todos los procesos se ejecutan en el servidor viajando al usuario las respuestas a cada uno de los procesos, lo que garantiza por una parte, que la carga en la red no sea significativa. (10)

El sistema ASSETS permite dentro del módulo de activos fijos, el control y manejo tanto de los activos fijos tangibles como los intangibles. Entre los procesos que se manejan están la Apertura y cierre de apertura, Entrada de activos fijos (entre los que se encuentra la operación de Alta) y Salidas de activos fijos (entre los que se encuentra las operaciones de Baja y Amortización). Además, permite controlar la amortización acumulada de cada activo, y modificarla al cierre de cada mes. (10)

En el sistema ASSETS el proceso de Apertura y cierre de apertura se realiza durante la implantación del Sistema, permitiendo introducir los activos existentes en la entidad, con sus datos actualizados de valor, amortización acumulada y ubicación física. A partir de ese momento el activo está disponible para cualquier otra transacción. (10)

[1] **SQL**: Lenguaje de consulta estructurado, es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas.

El proceso de Entradas de activos fijos genera tantas entradas como activos fijos recibidos en la entidad. Según el tipo de entrada que se genere se definen datos a entrar en el sistema. El resto de los datos, deben ser completados antes de realizar la confirmación. La entrada se hace efectiva al ejecutarse el proceso de confirmación. El grupo al que pertenece el activo fijo determina la tasa de depreciación y amortización del mismo. (10)

Versat Sarasola

Versat Sarasola, es un sistema de gestión contable-financiero, representa un ejemplo de sustitución de importaciones en materia de aplicaciones informáticas. Diseñado por TEICO Soft (MINAZ) Filial Villa Clara. Es un paquete integrado para la gestión económica financiera que permite enviar información eficaz, de forma inmediata, desde lugares apartados, a la vez que ofrece mayor organización, control y disciplina en cada gestión. El software incluye los módulos: Contabilidad General, Costos y Procesos, Finanzas, Caja y Banco, Inventarios, Activos Fijos, Facturación, Nómina de Salarios, Planificación, Configuración y Complementos. Es una herramienta para la planificación económica, el control y el análisis de gestión. (11)

Diseñado para su empleo en cualquier tipo de entidad empresarial o presupuestada. Permite llevar el registro y control contable individual de todos los hechos económicos que se originan en las estructuras internas de las entidades, así como exponer el estado financiero y toda la información económica y contable en este universo. Se estructura en un grupo de subsistemas en los cuales se procesan y contabilizan los documentos primarios, donde se anotan los movimientos, los recursos materiales, laborales y financieros que se utilizan en una entidad. (11)

El módulo de Activos Fijos abarca a los tangibles e intangibles. Acerca de este último trabaja los procesos fundamentales entre los que se encuentran, la Apertura y cierre de apertura, Alta y Baja, también posibilita el cálculo diario de las amortizaciones, conceptualiza los diferentes movimientos y posibilita configurar la contabilización de los mismos. Ofrece variedad de reportes sobre las existencias y movimientos de los activos. (11)

Permite trabajar los procesos como documentos, especificándolos como operaciones asociadas a documentos. La información básica que almacena de los Documentos son: Número (identifica a un documento desde el momento que se creó), Descripción (comentario que se especificó en el documento), Tipo de documento (clasificación del documento según el tipo), Estado (como se encuentra el documento en un momento especifico. Los estado definidos son: proceso, confirmado y cancelado), Fecha procesamiento (fecha en que se procesó el documento) y Usuario (usuario que realizó el documento). Entre los principales datos que almacena de los

activos se encuentran: descripción del activo, grupo de activo a que pertenece, área de responsabilidad donde se encuentra, valor de adquisición, método de amortización, código del activo. (11)

1.2.3 Valoración de las soluciones existentes

Luego de haber realizado un análisis de varios sistemas que gestionan los procesos de los AFI, se concluye que estos sistemas son capaces de manejar los AFI y realizar los procesos principales relacionados con estos. Por otro lado, tienen la característica común de estar desarrollados sobre software propietario, implicando incrementos de gastos en licencias de uso y mantenimiento del software, en las entidades que recurran al uso de los de origen extranjero, en tanto, los sistemas ASSETS NS y Versat Sarasola son aplicaciones de escritorio, trayendo como desventaja una engorrosa gestión de actualizaciones, ya que cada actualización debe ser instalada en cada uno de las estaciones de trabajo.

El estudio de estos sistemas y la interacción con algunos de los procesos que se realizan sobre los AFI como la Alta, permitió conocer cuáles fueron los principales datos de los activos que se manejan en dicho proceso, con esto se logró tener una mayor visión acerca de cómo deberían quedar almacenados los datos de los AFI durante la gestión de los mismos.

Mediante las Entrevistas a técnicos y especialistas se obtuvo información acerca del sistema ASSETS NS, resaltando durante su estudio que este sistema trata a los activos tangibles e intangibles con cierta similitud y mediante los mismos documentos. Esta característica genero la necesidad de construir una gestión de documentos la cual permitiría controlar y registrar formalmente las actividades que se pueden realizar sobre los AFI. Por otro lado, el sistema Versat Sarasola aportó en gran medida la información básica a mostrar en las vistas de documentos y activos.

1.3 Modelo de desarrollo de software

Adscrito a la Facultad 3 de la UCI se encuentra el CEIGE. Dicho centro productivo dirige sus resultados hacia la esfera de la gestión de empresas y entidades. La producción se concentra en el desarrollo de proyectos generalmente de gran magnitud, por lo que se hace necesario contar con un modelo estandarizado, que establezca las distintas fases por las que se debe transitar y el conjunto de artefactos a generar en cada una de ellas. (12)

A continuación se muestran las diferentes disciplinas por las que transitará el producto a desarrollar (Figura 1). (12)

Figura 1: Ciclo de vida del proyecto

1.3.1 Descripción de las fases del ciclo de vida de los proyectos

- **Inicio o Estudio preliminar:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto a un alto nivel, la evaluación de la factibilidad del proyecto y el registro de este. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo, y decidir si se ejecuta o no el proyecto.
- **Desarrollo:** En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se refinan los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. En esta fase se ejecutan las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas y Pruebas de liberación.

Descripción general de la disciplina Modelado del negocio

Es la disciplina destinada a comprender los procesos de negocio de la organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Artefactos a generar: Modelo conceptual,

Glosario de términos (actualizado), Modelo de Procesos de Negocio, Mapa de procesos de Negocio, Descripción de procesos de negocio. (12)

Descripción general de la disciplina Requisitos

El esfuerzo principal en la disciplina de Requisitos es desarrollar un modelo del sistema que se va a construir. Incluye un conjunto de artefactos que describen todas las interacciones que tendrán los usuarios con el software y que responden a los requisitos funcionales del sistema. Se especifican los requisitos funcionales y no funcionales. Artefactos a generar: Descripción de requisitos. (12)

Descripción general de la disciplina Análisis y diseño

Durante esta disciplina es modelado el sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima disciplina. Los artefactos generados en esta disciplina son más formales y específicos de una implementación. En caso de llevarse a cabo la reutilización de componentes software ya desarrollados, durante esta disciplina se ajusta el modelado existente a los requisitos actuales. Artefactos a generar: Diseño de casos de prueba, Modelo de datos, Diagrama de componentes, Diagrama de clases del diseño, Diagramas de secuencia. (12)

Descripción general de la disciplina Implementación

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares. Al reutilizar componentes software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes. (12)

Descripción general de la disciplina pruebas internas

Durante esta disciplina se desarrollan las pruebas del grupo de calidad del centro verificando el resultado de la implementación. Permite identificar posibles errores en la documentación y el software, es decir requisitos que el producto debería cumplir y que aún no los cumple. Artefactos a generar: No conformidades, Acta de liberación. (12)

Descripción general de la disciplina pruebas de liberación

Se aplican pruebas diseñadas e implementadas por el Laboratorio Industrial de Pruebas de Software a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación. (12)

En la presente investigación no se aplican las pruebas de liberación, pues solamente se llegará a la disciplina pruebas internas.

1.4 Notación para el Modelado

BPMN [2] 2.0 es un estándar de Modelamiento Empresarial que proporciona una notación para la especificación de procesos de negocio. Está basada en diagramas de flujo, similar a los diagramas de actividades, del UML. El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio. BPMN ha sido desarrollado para cubrir la brecha entre el diseño y la implementación de un sistema. (13)

1.5 Lenguaje de Modelado

UML 2.1 es un lenguaje de modelado de sistemas de software. Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. UML (*Unified Modeling Language*) ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocios y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables. Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. (14)

1.6 Herramientas a utilizar en el desarrollo de la solución

1.6.1 Herramientas CASE

Se puede definir a las Herramientas CASE (Computer Aided Software Engineering, por sus siglas en inglés) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. (15)

Visual Paradigm 5.0

Es una herramienta de diseño de UML, diseñado para ayudar al desarrollo de software. Se caracteriza por su disponibilidad en múltiples plataformas (Windows, Linux) y su diseño centrado en casos de uso y enfocado al negocio. Ofrece un completo conjunto de herramientas

[2] **BPMN**: Business Process Modeling and Notation (Notación de Modelado de Procesos de Negocio).

a los equipos de desarrollo de software necesario para la captura de requisitos, software de planificación, la planificación de controles, la clase de modelado, modelado de datos, y etc. (15)

1.6.2 Navegador Web

Mozilla Firefox 3.6 o superior

Es un navegador de código abierto y multiplataforma, disponible para la mayoría de los sistemas operativos actuales. Incluye la navegación por pestañas, un administrador de tareas, un sistema de búsqueda integrado mediante el motor de búsqueda escogido por el usuario. Como una de sus principales características está la de aceptar complementos creados por terceros. Cuenta, además, con una protección antimalware e integración con el antivirus y utiliza el sistema SSL[3] para proteger la comunicación con los servidores web. (16)

Para los desarrolladores web posee un repertorio de herramientas incorporadas, como la Consola de errores, Scratchpad (para probar código JavaScript), editor HTML, el Inspector DOM, o incluyendo extensiones como Firebug. Este último es un paquete de utilidades con el que se puede analizar (revisar velocidad de carga, estructura DOM), editar, monitorizar y depurar el código fuente, CSS, HTML y JavaScript de una página web de manera instantánea. (16)

1.6.3 Ambiente de desarrollo integrado (IDE)

NetBeans 7.3.1

Es un entorno de desarrollo integrado libre, que permite a los programadores escribir, compilar, depurar y ejecutar programas. Posibilita la implementación de software utilizando otros lenguajes de programación. Cuenta con una enorme cantidad de extensiones y plugins, que pueden incorporarse al sistema para facilitar las tareas de programación. Además, tiene una intuitiva interfaz de usuario, un sistema de depuración de programas y las demás herramientas de ayuda, que facilitan el desarrollo de software. Es una aplicación de código abierto bajo licencias libres por lo que tiene una amplia comunidad de desarrollo en constante crecimiento. (17)

[3] **SSL**: Secure Sockets Layer (capa de conexión segura) protocolos criptográficos que proporcionan comunicaciones seguras por una red, comúnmente Internet.

1.6.4 Servidor de Aplicación Web

Apache 2.2.9

El servidor HTTP (Hypertext Transfer Protocol) Apache es un servidor web HTTP de código abierto, para plataformas Unix, Microsoft Windows, Macintosh y otras. Apache presenta entre otras características bases de datos de autenticación y negociado de contenido. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación el servidor Apache o que utilizarán características propias del mismo. Apache es el componente de servidor web en la popular plataforma de aplicaciones LAMP, junto a MySQL y los lenguajes de programación PHP, Perl, Python y ahora también Ruby. Se caracteriza por ser modular, multiplataforma, extensible y de código abierto. (18)

1.6.5 Herramientas de base de datos

PgAdmin 1.16

Es una herramienta de código abierto para la administración de bases de datos PostgreSQL y derivados. Está diseñado para responder a las necesidades de la mayoría de los usuarios, desde escribir simples consultas SQL hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y hace simple la administración. (19)

PostgreSQL 8.3

Es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (ORDBMS) de código abierto. Proporciona un gran número de características que solamente se encontraban en las bases de datos comerciales tales como DB2 u Oracle. PostgreSQL tiene soporte para lenguajes internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Además soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario. (20)

1.6.6 Control de versiones

El control de versiones es una combinación de tecnologías, permite controlar los cambios realizados en los ficheros del proyecto, en particular en el código fuente, en la documentación y en las páginas web.

Subversion 1.7.6

Es un controlador de versiones empleado en la administración de archivos utilizados en el desarrollo de software o contenido. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintas computadoras. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo conjunto de

datos desde sus respectivas ubicaciones fomenta la colaboración. Se puede progresar de modo más rápido sin un único conducto por el cual deban pasar todas las modificaciones. (21)

1.7 Tecnologías a utilizar en el desarrollo de la solución

En el centro CEIGE donde se desarrolla el Sistema Integral de Gestión CEDRUX, el Departamento de Tecnología definió las tecnologías que se usaron en la presente investigación, a continuación se hace una breve descripción de las mismas:

1.7.1 AJAX

Es una tecnología de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. (22)

Las tecnologías que forman a AJAX[4] son:

- XHTML[5] y CSS[6], para crear una presentación basada en estándares.
- XML[7], XSLT y JSON[8], para el intercambio y la manipulación de información.
- XMLHttpRequest[9], para el intercambio asíncrono de información.
- JavaScript, para unir todas las demás tecnologías.

XML

Es el estándar de Extensible Markup Language (Lenguaje de Etiquetado Extensible), que se encuentra conformado por un conjunto de reglas para definir etiquetas semánticas orientadas a organizar un documento en diferentes partes. Permite al usuario definir sus propios lenguajes de anotación adaptados a sus necesidades y contiene tres características muy importantes que son: extensibilidad, estructura y validación. Permite compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. (23)

---

[4] **AJAX:** Acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML).

[5] **XHTML:** eXtensible HyperText Markup Language. Es básicamente HTML expresado como XML válido.

[6] **CSS:** Hojas de Estilo en Cascada.

[7] **XML:** Lenguaje de Etiquetado Extensible.

[8] **JSON:** JavaScript Object Notation (Notación de Objetos JavaScript).

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

XHTML

XHTML, por sus siglas en inglés, Extensible Hypertext Markup Language (lenguaje extensible de marcado de hipertexto) es una versión más estricta y limpia de HTML pensado para sustituir a HTML como lenguaje estándar para las páginas Web XHTML es en sí mismo XML por lo que sigue las reglas del mismo que son: bien formado y válido. Que esté bien formado significa que debe contener un único elemento raíz y que todas las etiquetas que se abren deben tener una etiqueta de cierre. Que sea válido en cambio significa que debe seguir las reglas especificadas en un DTD (Definición de Tipo de Documento) o un schema[10] que son los que definen las reglas de cómo debe estar estructurado el documento XML a la hora de ser validado. (24)

JSON

JSON es un formato de datos muy ligero basado en un subconjunto de la sintaxis de JavaScript: literales de matrices y objetos. 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, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos. (25)

CSS

CSS en inglés Cascading Style Sheetses un mecanismo para añadir estilo (fuentes, colores, espaciados) a documentos web. Una característica fundamental de CSS es que más de una hoja de estilo puede influir en la presentación de un documento. Esta característica se conoce como cascada, dado que las hojas de estilos diferentes son consideradas como procedentes de una serie. (26)

1.7.2 Marco de trabajo Sauxe

Sauxe es un marco de trabajo desarrollado en el CEIGE para la construcción de aplicaciones web de gestión, que centra su desarrollo en los requisitos funcionales, las interfaces de usuario y la lógica del negocio. El marco de trabajo brinda funcionalidades de seguridad, auditoría, interoperabilidad, concurrencia, administración de transacciones, entre otras. Es una solución que sigue el principio de soberanía tecnológica, facilidad de mantenimiento, reusabilidad e integración. (27)

[10] **SCHEMA:** es un lenguaje de esquema escrito en XML.

Figura 2: Arquitectura de Sauxe

ExtJS 2.2

ExtJS es una librería JavaScript que permite construir aplicaciones complejas en internet además de flexibilizar el manejo de componentes de la página como el DOM, Peticiones AJAX, DHTML, tiene la gran funcionalidad de crear interfaces de usuario. Permite crear aplicaciones complejas utilizando componentes predefinidos. (28)

Zend Framework 1.7

Se trata de un framework para desarrollo de aplicaciones Web y servicios Web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es Open Source y trabaja con PHP 5. Este framework está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Entre los componentes que se encuentran tienen vital importancia: Zend_Config para temas de configuración de aplicaciones web, Zend_Db para tratar con bases de datos, Zend_Search o Zend_Feed entre otros. La instalación es sencilla, solamente se tendrá que añadir en el fichero de configuración php.ini, el path hasta la carpeta library del framework con la instrucción include_path. (29)

Doctrine Framework 1.2.1

Doctrine es un potente y completo sistema ORM (Object Relational Mapper por sus siglas en inglés) para PHP 5.2+ con un DBAL (Database Abstraction Layer por sus siglas en inglés, Capa de Abstracción a Base de datos) incorporado. Entre muchas otras cosas brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Por otro lado, como la librería es bastante grande ésta tiene un método para ser compilada al pasar a producción. (30)

1.8 Lenguajes de programación a utilizar en el desarrollo de la solución

Lenguaje de programación: es un idioma artificial diseñado para controlar el comportamiento de una máquina como las computadoras, para expresar algoritmos con precisión. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. (31)

1.8.1 Lenguaje de programación del lado del servidor

PHP5.3.10

En el desarrollo de la solución se usará el lenguaje del lado del servidor PHP[11], que es un lenguaje de programación interpretado de alto nivel embebido en páginas HTML. Es gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. Permite rápidamente a los desarrolladores la generación dinámica de páginas. Es uno de los lenguajes de programación más populares, debido a la gran fluidez y rapidez de sus scripts; siendo realmente fácil de utilizar por ventajas como su gratuidad y seguridad. Otra gran ventaja que presenta es que permita la conexión a las bases de datos más comunes como son MySQL, Oracle, PostgreSQL, Microsoft SQL Server, entre otras. (32)

1.8.2 Lenguaje de programación del lado del cliente

Java Script

Es un lenguaje interpretado, es decir, no requiere compilación. El navegador del usuario se encarga de interpretar las sentencias JavaScript contenidas en una página HTML y ejecutarlas adecuadamente. Es un lenguaje orientado a eventos. Cuando un usuario accede a un enlace o mueve el puntero sobre una imagen se produce un evento. Mediante JavaScript se pueden desarrollar scripts que ejecuten acciones en respuesta a estos eventos. Permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. (33)

1.9 Conclusiones parciales del capítulo.

- La definición de los conceptos fundamentales permitió conocer qué es un AFI y qué son los procesos de los AFI.

[11] **PHP**: Es el acrónimo de Hipertext Preprocesor.

- El estudio de los sistemas que gestionan los procesos de los AFI permitió obtener información acerca de estos, como son los datos que registran de los AFI, entre otros, lo cual fue utilizado en el desarrollo de la solución.
- El estudio del modelo de desarrollo de software del centro CEIGE permitió conocer las distintas fases a transitar y el conjunto de artefactos a generar en cada una de ellas durante la investigación.
- El estudio de las herramientas y tecnologías permitió obtener habilidades para enfrentarse al desarrollo de la solución.

CAPÍTULO 2. ANÁLISIS Y DISEÑO

Introducción

En este capítulo se representan las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño, así como los artefactos que en estas se generan. Entre ellos se describen los procesos del negocio relacionados a la investigación, además de los aspectos más importantes relacionados con la arquitectura y el diseño de la aplicación.

2.1 Modelo Conceptual

En la Figura 3 se muestra el modelo conceptual donde se representan las entidades que lo conforman y sus atributos, estableciendo una representación de los conceptos del dominio del problema dada las asociaciones y relaciones que se determinan entre ellas.

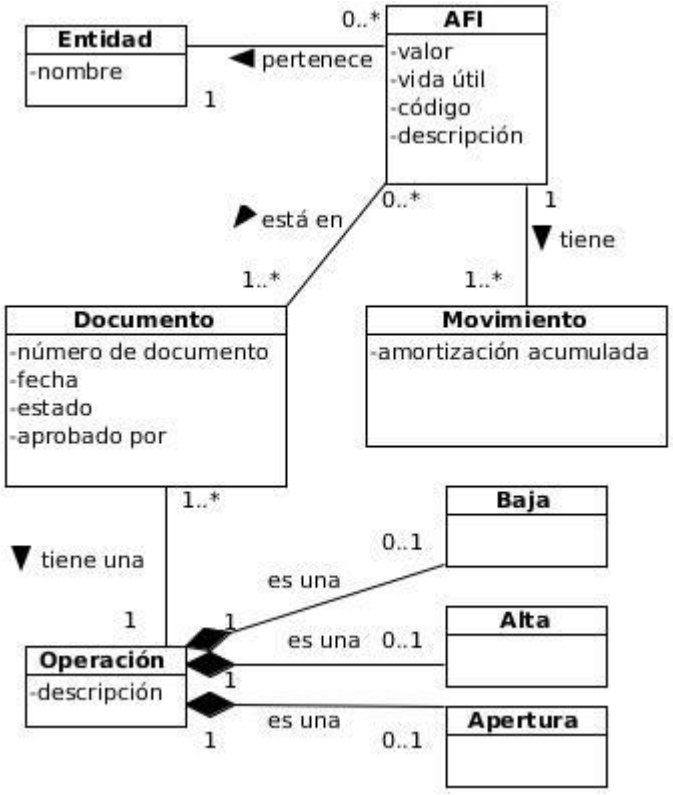

Figura 3: Modelo Conceptual

2.2 Procesos del negocio

Permite tener una visión del negocio mediante la definición, descripción y representación de los procesos. Además de comprender los procesos de negocio de la organización.

2.2.1 Descripción de proceso de negocio

Apertura y Cierre de Apertura

El proceso de apertura consiste en fijar un inventario inicial donde se registran los datos de los Activos Fijos Intangibles existentes en la entidad en el momento de comenzarse a trabajar con el subsistema. Se procede a cerrar la apertura, cuando el total de los valores de los Activos Fijos Intangibles captados y el de sus amortizaciones acumuladas coincida con los respectivos saldos que muestren los Submayores y el Mayor de la entidad. Se ejecuta por una sola vez y consiste en bloquear la utilización de la opción Apertura. A partir de ese momento solamente se podrán realizar altas para ingresar activos. (Ver Figura 4)

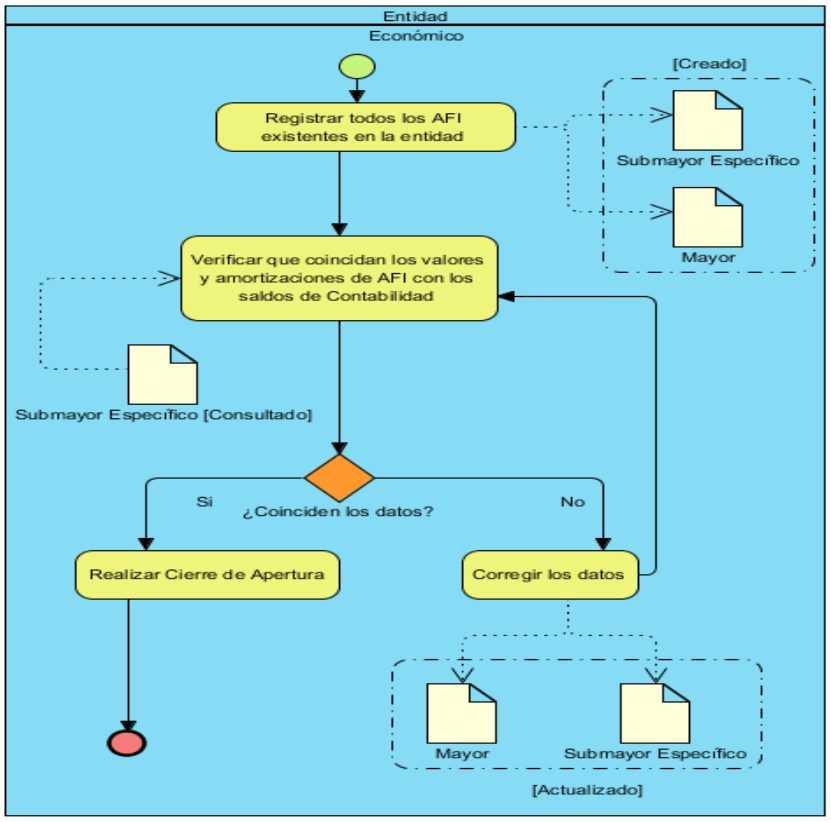

Figura 4: Diagrama del proceso: Apertura y Cierre de Apertura

Alta de Activos Fijos Intangibles

El proceso comienza con el registro de los activos que son adquiridos por la entidad, para ello se requiere del documento legal que asegura a la empresa el derecho sobre este activo pues de él se tomarán todos los datos necesarios para llevar a cabo el procedimiento para el alta. (Ver Figura 5)

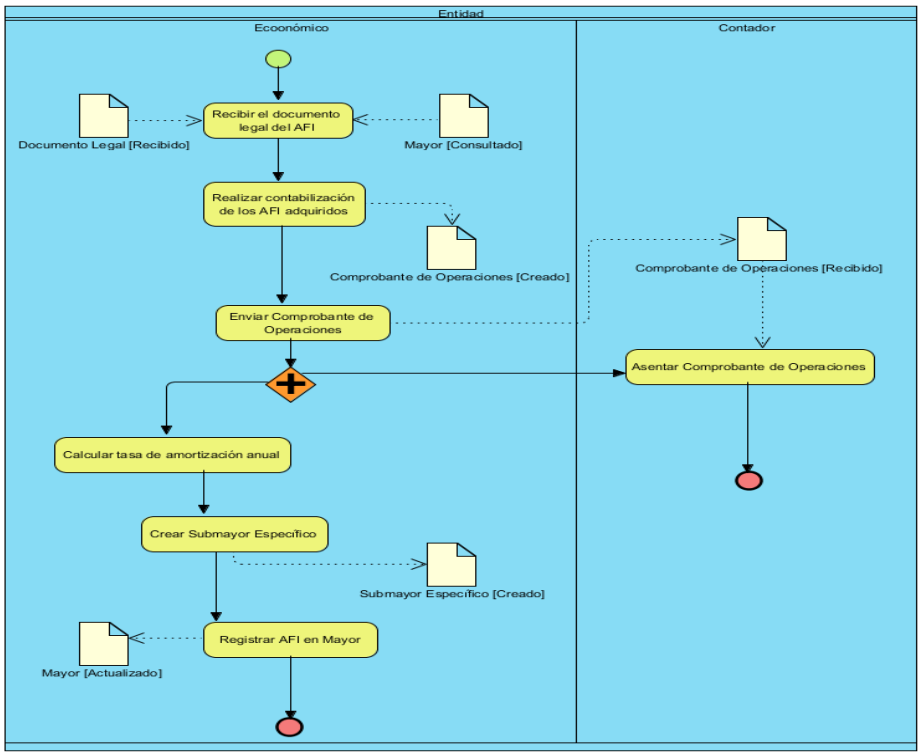

Figura 5: Diagrama del proceso: Alta de Activos Fijos Intangibles

Baja de Activos Fijos Intangibles

El proceso consiste en dar baja a un activo cuando ha llegado al límite de su vida útil o al término de su contrato para la entidad. (Ver Figura 6)

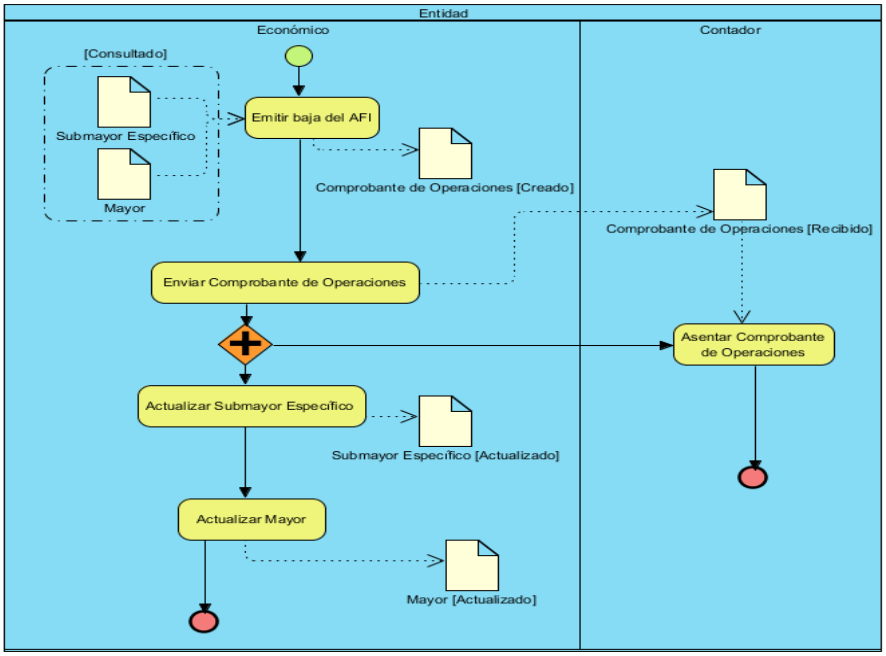

Figura 6: Diagrama del proceso: Baja de Activos Fijos Intangibles

2.2.2 Mapa de procesos del negocio

El mapa de proceso es una representación que evidencia de manera global el funcionamiento de los procesos existentes y cómo se entrelazan entre sí. El mismo abarca los tres procesos definidos anteriormente: Apertura y Cierre de Apertura, Alta de Activos Fijos Intangibles, Baja de Activos Fijos Intangibles.

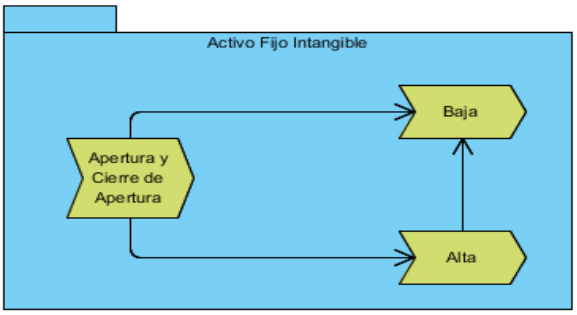

Figura 7: Mapa de procesos de Negocio

2.3 Requisitos del Software

La definición de las necesidades de un sistema es un proceso complejo principalmente si el entorno de trabajo es desconocido para el equipo de analistas, pues en él hay que identificar los requisitos que se deben cumplir para satisfacer las necesidades de los usuarios finales. (34)

2.3.1 Técnicas empleadas en la captura de requisitos

Para el levantamiento de requisitos se emplearon las técnicas de captura que se explican a continuación. Las cuales permitieron obtener la información necesaria para la realización de dicha tarea:

- **Entrevistas:** permitió obtener conocimiento del problema y comprender los objetivos de la solución buscada. Mediante esta técnica se le hicieron preguntas a personas conocedoras del tema a fin de establecer las posibles funcionalidades del sistema.
- **Brainstorming (Tormenta de ideas):** esta técnica se puso en práctica mediante los talleres y reuniones donde participaron los profesionales de la Línea. De esta forma todos expresaron sus ideas sobre el problema y su solución. Permitió revisar las descripciones de procesos de negocio para obtener otros requisitos.
- **Revisión documental:** esta técnica se aplicó realizando un análisis y revisión de documentaciones existentes que tratan a los procesos relacionados con los Activos

Fijos Intangibles, con el fin de concretar el conocimiento que permitió obtener los requisitos del sistema.

2.3.2 Requisitos funcionales

Después de haber realizado el proceso de captura de requisitos se determinó que la aplicación debe contar con los siguientes requisitos:

RF1. Gestionar Documentos

RF1.1 Adicionar documento.

RF1.2 Modificar documento.

RF1.3 Eliminar documento.

RF1.4 Buscar documento.

RF1.5 Listar documento.

RF1.6 Aprobar documento.

RF1.7 Confirmar documento.

RF1.8 Contabilizar documento.

RF1.9 Cancelar estado del documento.

RF1.10 Consultar documento

RF1.11 Realizar búsqueda avanzada de documento.

RF2. Gestionar AFI al Documento.

RF2.1 Adicionar AFI al documento.

RF2.2 Eliminar AFI del documento.

RF2.3 Buscar AFI en el documento.

RF2.4 Listar AFI en el documento.

RF2.5 Buscar Activos de entrada.

RF2.6 Buscar Activos de no entrada.

RF2.7 Listar Activos de entrada.

RF2.8 Listar Activos de no entrada.

RF3. Gestionar datos AFI al Documento.

RF3.1 Adicionar datos del AFI al documento.

RF3.2 Modificar datos del AFI al documento.

2.3.3 Descripción de requisitos

En el presente epígrafe se describe uno de los requisitos definidos anteriormente. El resto de las descripciones de requisitos pueden ser consultadas en el Anexo 1.

2.3.3.1 Descripción del requisito Gestionar datos AFI al Documento escenario Adicionar datos AFI al documento

Precondiciones	El cliente ha sido validado. Debe existir al menos un documento en el sistema. Debe existir al menos un AFI registrado en el sistema. Se ha seleccionado un documento de Apertura de AFI o Movimiento de AFI y la operación es Alta de AFI. El documento debe estar en estado de “Elaboración”.
Flujo de eventos	
Flujo básico Adicionar datos AFI	
1	Se listan los AFI del documento: Listar AFI del documento; en la agrupación Gestionar AFI del documento.
2	Se introducen los datos al AFI: -Valor del activo -Descripción -Código -Vida útil -Estado del activo -Amortización acumulada
3	El sistema valida (ver validación 1) los datos introducidos.
4	Si los datos son correctos el sistema los registra.
5	El sistema confirma el registro de los datos.
6	Concluye el requisito.
Pos-condiciones	
1	Se registró en el sistema los datos del AFI.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual de AFI

Conceptos	**Documento**	Número de documento, estado, fecha, entidad
	AFI	descripción, código, valor_inicial, vida_útil, amortización_acumulada, tasa_amortización,
Asuntos pendientes	N/A	

Tabla 1: Especificación del RF Gestionar datos AFI al Documento. Escenario Adicionar Datos AFI al documento

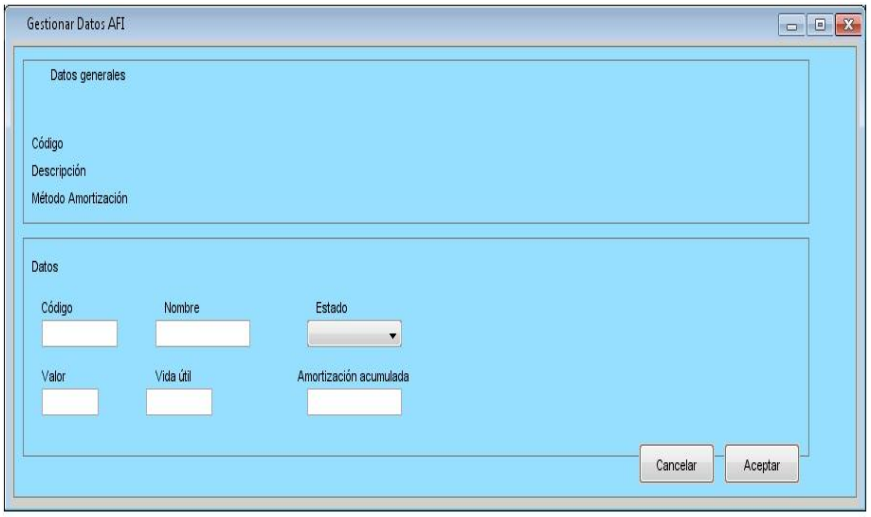

Figura 8: Prototipo de IU Gestionar datos AFI al Documento

2.3.4 Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Permiten que clientes y usuarios valoren las características no funcionales del producto, pues aunque cumpla con las funcionalidades requeridas, las características no funcionales pueden hacer una gran diferencia entre un producto que goce de la aceptación de todos y uno con poca aceptación. (35)

Portabilidad

La aplicación debe funcionar en varias plataformas, principalmente (Linux y Windows), siendo posible su acceso a través de un navegador Web.

Rendimiento

Las interfaces del sistema se muestran en un tiempo promedio de 0.1 a 1.5 segundos. El sistema deberá tener un tiempo de respuesta promedio de peticiones de 0.2 a 4 segundos.

Seguridad

La seguridad de las aplicaciones del Sistema Cedrux, perteneciente a CEIGE, se gestiona mediante el sistema Acaxia, el mismo está desarrollado sobre el marco de trabajo Sauxe. Acaxia gestiona la seguridad en un entorno de varias aplicaciones y garantiza temas tan importantes como: Autenticación, Autorización, Auditoria, Administración de perfiles, Administración de conexiones, Entornos multi-entidad, Entornos multi-lenguaje, Entornos multi-temas, Compartimentación de la información.

De aplicación o Software (SFT)

El sistema requiere como componentes de aplicación:

- Navegador Mozilla Firefox (recomendado versión 3.6 o superior).

Del lado del servidor de Aplicaciones requiere:

- Soporte para PHP5, un servidor web apache versión 2.2 o superior, Sistema Operativo Ubuntu Server (recomendado Ubuntu Server 10.04 LTS).

Del lado del servidor de BD requiere:

- Sistema Operativo Ubuntu Server (recomendado Ubuntu Server 10.04 LTS), PostgreSQL versión 8.3.

Dispositivo o Hardware (HDW)

Los dispositivos que se necesitan para el correcto funcionamiento de la aplicación son:

Por el lado cliente:

- Procesador: 1.40 GHZ, RAM: 256 MB (recomendado 512 Mb), Tarjeta de Red: 1.

Del lado del servidor de aplicaciones:

- Tarjeta de Red: 1, Procesador: 3.00 GHZ, RAM: 1GB, UPS: 1.

Del lado del servidor de Base de datos:

- Tarjeta de Red: 1, Procesador: 3.00 GHZ, RAM: 1GB, UPS: 1.

2.3.5 Validación de los requisitos funcionales

Es muy importante asegurar la validez de los requisitos previamente a comenzar un desarrollo de software. Para ello debe hacerse una comprobación de la correspondencia entre las descripciones iniciales y si el modelo es capaz de responder al planteamiento inicial. (36) Las técnicas que se aplicaron para validar los requisitos identificados y especificados fueron las siguientes:

Prototipo de interfaz de usuario: Es una técnica de representación aproximada de la interfaz de usuario de un sistema software que permite a clientes y usuarios entender más fácilmente la propuesta de los ingenieros de requisitos para resolver sus problemas de negocio. Una vez identificados y especificados los requisitos se obtuvieron los diseños de interfaz de usuario utilizando la herramienta CASE Visual Paradigm. (36)

Diseño de casos de prueba: Este método consiste en la definición de casos de prueba que permitan verificar el cumplimiento de los requisitos funcionales. Los casos de prueba se hacen teniendo en cuenta la especificación de los requisitos y para cada uno de ellos se elaboró casos de prueba. (36)

Mediante estas técnicas se obtuvieron 21 Diseños de casos de prueba, generándose un artefacto por cada requisito y 9 prototipos de Interfaz de Usuario.

2.4 Diseño del Sistema

El diseño se aplica de manera independiente al modelo de software que se utilice. Una vez que se analizan y especifican los requisitos, el diseño del software es la última acción de la ingeniería correspondiente dentro de la actividad del modelado, la cual establece una plataforma para la construcción del sistema.

2.4.1 Modelo de Datos

El modelo de datos permite describir los elementos de la realidad que intervienen en un problema dado y la forma en que se relacionan entre sí. El modelo de datos propuesto (figura 10) consta de 8 tablas para su obtención se tuvo en cuenta la reducción a la mínima expresión de los campos nulos.

De las 8 tablas definidas 3 son nomencladores encargados de gestionar conceptos específicos del negocio ya redefinidos, ejemplo: la tabla “nom_afi” gestiona los tipos de Activos Fijos Intangibles y su vínculo con el grupo al que pertenece.

Las tablas restantes se encargan de gestionar los datos que son necesarios tener registrados, por ejemplo, la tabla “dat_doc_afi” almacena los principales datos relacionados con los documentos, como son el número que lo identifica y el estado en que se encuentra.

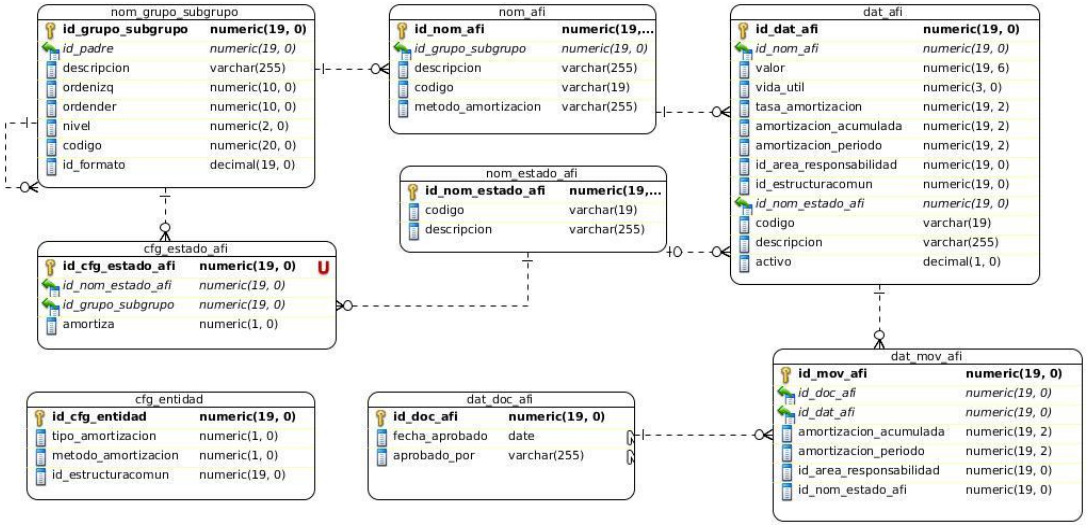

Figura 9: Modelo de Datos del Subsistema de AFI

2.4.2 Diagrama de componentes

Tomando como referencia la definición adoptada y publicada por el Software Engineering Institute (SEI): “Un componente es un fragmento de un sistema de software que puede ser ensamblado con otros fragmentos para formar piezas más grande o aplicaciones completas.” (37)

Un diagrama de componentes representa cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos, a partir de los servicios que brindan. A continuación se muestra el diagrama de componentes de la propuesta de solución y las relaciones que se establecen.

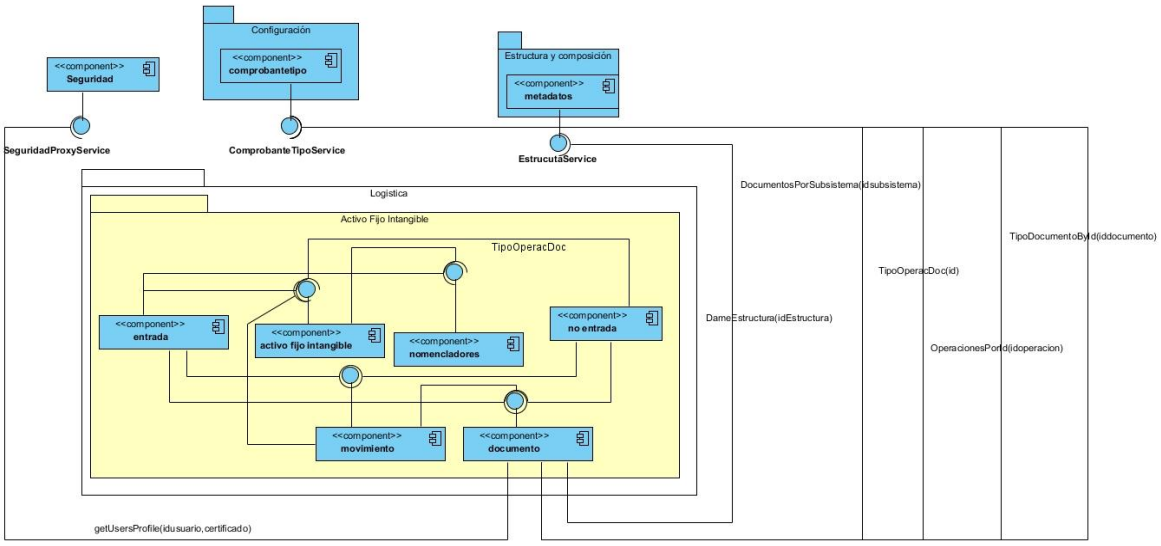

Figura 10: Diagrama de componentes

2.4.3 Diagramas de Clases del Diseño

El diagrama de clases del diseño especifica la estructura de clases del sistema con relaciones entre clases y estructuras de herencia. Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. (38)

En el diseño de la solución se elaboró un diagrama de clases por cada agrupación de requisito. En la Figura 12 se muestra el diagrama de clases del diseño correspondiente a la agrupación de requisitos Gestionar datos AFI al Documento, los restantes diagramas se encuentran en el Anexo 2. Las páginas clientes están compuestas por archivos java script los cuales representan las interfaces de usuario de la aplicación. Por otra parte estas interfaces están compuestas por formularios, los cuales registran los datos que serán enviados al servidor para su manipulación. Las clase controladora (las clases que terminan con el nombre Controller) es la encargada de seleccionar qué clase modelo (las que terminan con el nombre Model) es la adecuada para realizar la acción encomendada por el usuario, o sea Insertar, Actualizar (modificar), Eliminar, Listar, Buscar, entre otras. Las clases modelo le piden los datos a las clases del dominio (clases encargadas del acceso a datos) de forma directa o a través de servicios. Por último las clases del dominio proveen la información solicitada por la controladora y esta se encarga de que las páginas clientes reciban la misma.

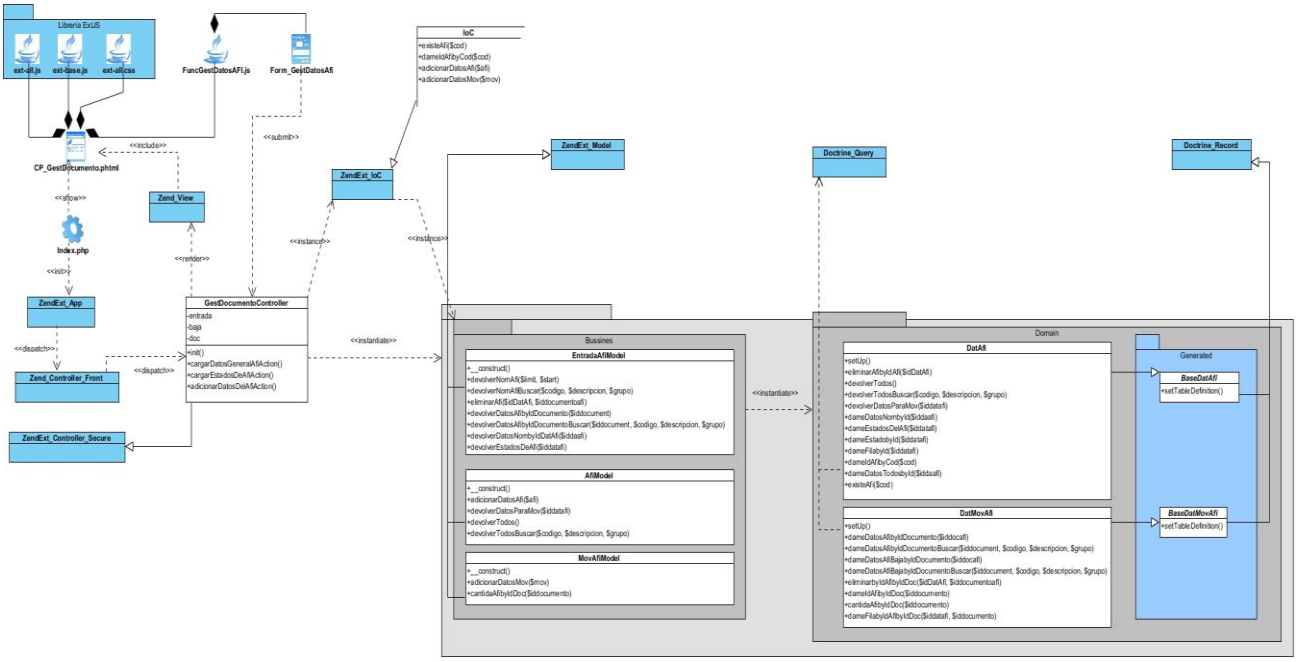

Figura 11: Agrupación de requisito Gestionar datos AFI al Documento

2.4.4 Diagrama de Secuencia

Los modelos de secuencia son modelos dinámicos que documentan, para cada modo de interacción, la secuencia de interacciones que tienen lugar entre los objetos. En un modelo de secuencia los objetos involucrados en la interacción están ordenados horizontalmente con una línea vertical a cada objeto. Las interacciones entre los objetos se representan por flechas etiquetadas que vinculan a las líneas verticales. Estos no son datos que fluyen, sino que representan mensajes o eventos que son fundamentales para la interacción. (39) En la Figura 13 se muestra el diagrama de secuencia correspondiente al requisito Adicionar datos del AFI al documento, en el Anexo 3 se muestran los diagramas correspondientes a las restantes agrupaciones de requisitos.

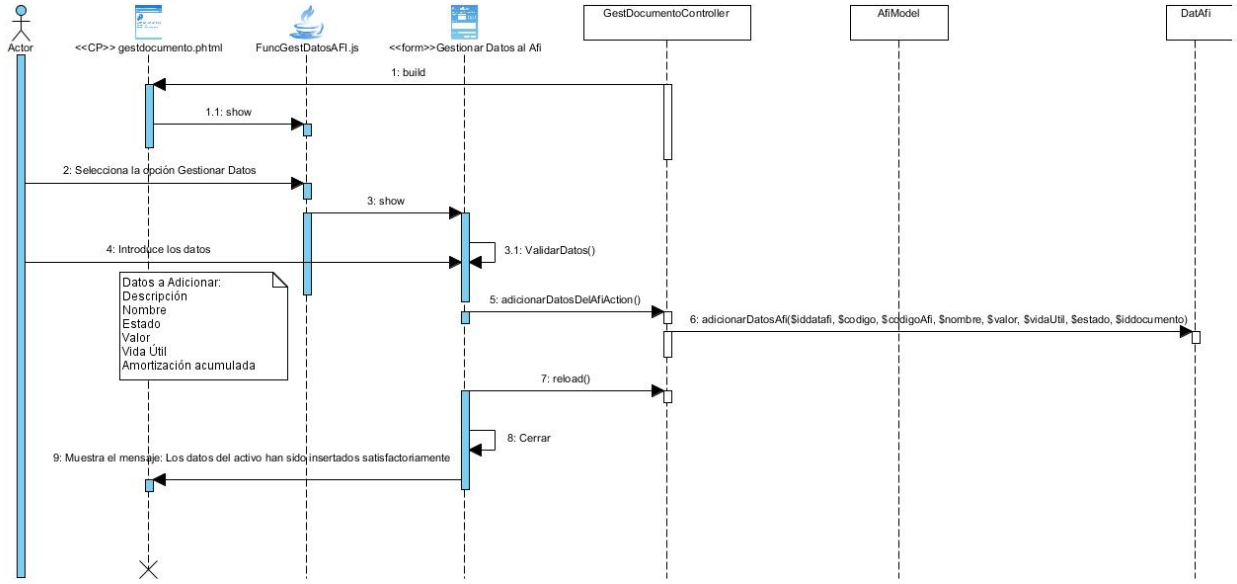

Figura 12: Diagrama de secuencias del requisito Adicionar datos del AFI al documento

2.4.5 Estilos y patrones arquitectónicos utilizados

2.4.5.1 Arquitectura basada en componentes

El marco de trabajo Sauxe, sobre el cual fue desarrollado el sistema en cuestión, se rige por una arquitectura basada en componentes de forma horizontal, que tiene como objetivo hacer un uso correcto del software reutilizable.

Un componente es un fragmento reemplazable de un sistema de software, que satisface una o varias funcionalidades dentro del contexto de una arquitectura bien definida y puede ser ensamblado con otros fragmentos por medio de una interfaz. (40)

El sistema Cedrux está formado por varios subsistemas los cuales cuentan con una serie de componentes que trabajan de forma independiente y pueden ser reutilizados por otros subsistemas.

2.4.5.2 Modelo Vista Controlador (MVC)

El Modelo Vista Controlador (Model View Controller, MVC por sus siglas en inglés) es un patrón de arquitectura de software que organiza el código de acuerdo a la función que realiza. Es el patrón que será usado en la construcción del sistema, ya que ofrece muchas ventajas tanto a la hora de construir la aplicación como a la hora de agregarle nuevas funcionalidades. Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos (Modelo, Vista y Controlador). (41)

Para comprender cómo funciona el patrón Modelo vista controlador, se debe entender la división a través del conjunto de estos tres elementos y como estos componentes se comunican unos con los otros y con otras vistas y controladores externos al modelo principal. Para ello, es importante conocer que el controlador interpreta las entradas del usuario (tanto teclado como el ratón), enviando el mensaje de acción al modelo y a la vista para que se proceda con los cambios que se consideren adecuados. (41)

La aplicación de este patrón se evidencia en los diagramas de clases del diseño. En este se muestra la clase del modelo (AfiModel) que representa la lógica de negocio. La vista (las páginas .phtml donde se incluyen las clases .js) permite al usuario interactuar con el sistema. El controlador (GestDocumentoController) realiza las peticiones del usuario, notificando al modelo la acción que debe realizar, este último devuelve los resultados para que así el controlador le envíe la información al usuario mediante la vista. Las clases mencionadas en este ejemplo se podrán observar en la Figura 12.

2.4.5.3 Patrones GRASP

Los patrones GRASP[12] describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. Son una serie de buenas prácticas de aplicación recomendable en el diseño de software. (42)

A continuación se describen los patrones GRASP que se aplicaron en el diseño de la solución:

Patrón Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Su función es asignarle a una determinada clase la responsabilidad de crear una instancia de otra clase, relacionada con la creación de objetos. (42)

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

Patrón Controlador

12 **GRASP:** acrónimo que significa **G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns (Patrones de asignación de responsabilidades)

Este patrón garantiza que los procesos de dominio sean manejados por la capa de los objetos del dominio y no por la de interfaz. Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase determinada. (42)

La clase controladora definida: GestDocumentoController, es un ejemplo de la aplicación de este patrón, la misma tendrá a cargo la responsabilidad de manejar los eventos dentro del sistema.

Patrón Alta Cohesión

La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme debido a que una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. (42)

Este patrón fue utilizado en el diseño del sistema de manera general; agrupando las clases en dependencia de los requisitos, según la premisa de que la información almacenada en una clase debe ser coherente y estar relacionada con ésta en mayor medida, enfocada en sus responsabilidades.

Patrón Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. El bajo acoplamiento significa que una clase no depende de muchas clases. (42)

Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, las clases se encuentran lo menos relacionadas entre sí, para que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión en las demás, potenciando la reutilización y disminuyendo la dependencia entre las clases.

Patrón Experto

Sugiere asignar la responsabilidad al objeto que posea la información necesaria para desempeñarla. El procedimiento se distribuye entre las clases que cuentan con la información requerida, incitando con ello definiciones de clases sencillas y altamente cohesivas que son más fáciles de comprender y mantener. (42)

Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El sistema cuenta con clases

controladoras, modelos y de dominio que poseen funciones concretas de acuerdo con la información que gestionan.

2.4.5.4 Patrones GOF

Los patrones GOF[13] describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Están clasificados en patrones creacionales, estructurales y de comportamiento. Los creacionales son los encargados de la creación de instancias de los objetos, los estructurales plantean las relaciones entre clases y los de comportamiento plantean la interacción y cooperación entre las clases. (43)

A continuación se describen los patrones GOF que serán utilizados en el diseño de la solución:

Chain of Responsibility (Cadena de responsabilidad)

Es un patrón de comportamiento que evita acoplar el emisor de un mensaje a su receptor dándole a más de un objeto la posibilidad de manejar la solicitud. Se define una cadena de objetos, de modo que un objeto pasa la solicitud al siguiente en la cadena hasta que uno la maneja. (43)

Se evidencia al distribuir las responsabilidades, puesto que ante la ocurrencia de un error al realizar una determinada consulta a la base de datos el mismo es manejado por el Modelo, creando una nueva excepción de tipo ZendExt_Exception. Dicha excepción debe ser propagada al Controlador, el cual será el encargado de capturarla y enviarla a la Vista ya traducida, esta última por su parte mostrará un mensaje al usuario en un lenguaje entendible notificando el error.

Patrón Facade (del español Fachada)

Es un patrón estructural que proporciona una única interfaz a un conjunto de clases de un subsistema. Define una interfaz de más alto nivel que facilita el uso de un subsistema. Las fachadas permiten estructurar el sistema en capas y reduce las dependencias de compilación. (43)

Este patrón se evidencia en los servicios que consumen los componentes a través del mecanismo de Inversión de Control (IoC: Inversion of Control). Donde se puede acceder a métodos de otros componentes que se encuentran tanto dentro como fuera del subsistema de Activos Fijos Intangibles.

13 GOF: Gang of Four (La banda de Cuatro)

Singleton

Es un patrón creacional, es utilizado para garantizar que una clase solamente tenga una instancia y proporcione un punto de acceso global a ella. (43)

Una de las clases donde se observa su utilización en el sistema es GestDocumentoController, la cual crea instancias únicas de las modelos a las que puede acceder directamente. Estas instancias serán utilizadas a lo largo de toda la clase controladora.

2.5 Validación del modelo de diseño propuesto

2.5.1 Métricas orientadas a clases para evaluar el diseño

El diseño propuesto fue validado a partir de las métricas orientadas a clases, concebidas como instrumentos para evaluar la calidad del diseño, definidas por Lorenz y Kidd, Tamaño operacional de las clases (TOC) y las de Chidamber y Kemerer, Relaciones entre clases (RC), que proponen validar el diseño mediante la evaluación de los siguientes atributos de calidad:

- **Tamaño operacional de las clases:**

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

- **Relación entre clases:**

Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta pero fuertemente en los costes y la planificación del proyecto.

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad del producto diseñado.

2.5.1.1 Resultados del instrumento de evaluación de la métrica Tamaño operacional de clase (TOC)

Al aplicar la métrica se tuvieron en cuenta las clases Controladoras, Modelos y del Domain reflejadas en el diseño de acuerdo a la propuesta planteada, así como la cantidad de procedimientos (métodos) por cada una de ellas.

Para determinar el valor de los atributos, se calcula el promedio de la columna cantidad de procedimientos y este promedio es el que se emplea en la columna criterio (El instrumento y la tabla de resultados, se muestran en el Anexo 4).

Resultados en % de los atributos de calidad afectados:

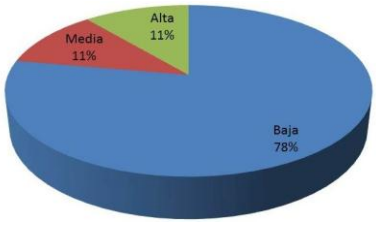

Figura 13: Representación del valor en % del atributo responsabilidad

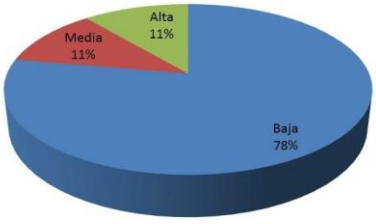

Figura 14: Representación del valor en % del atributo complejidad

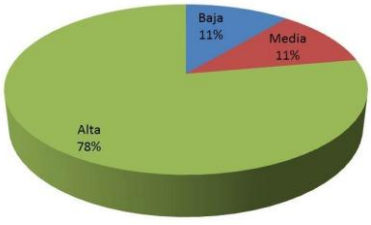

Figura 15: Representación del valor en % del atributo reutilización

Análisis de los resultados obtenidos en la evaluación de la métrica TOC:

Considerando que el 78% de las clases contienen un número de funcionalidades inferior a la media de procedimientos por clases (cuyo valor asciende a 13.7), dando como resultado una elevada reutilización, baja responsabilidad y complejidad de implementación en el diseño propuesto. Solamente un 11% de las clases tienen pocas posibilidades de reutilización y una

gran complejidad de implementación. Estos valores demuestran que los indicadores de reutilización, complejidad y responsabilidad son aceptables.

2.5.1.2 Resultados del instrumento de evaluación de la métrica Relaciones entre Clases (RC)

La métrica Relaciones entre Clases está dada por el número de relaciones de uso de una clase con otra(s). Se tuvieron en cuenta las clases Controladoras, Modelos y del Domain reflejadas en el diseño de acuerdo a la propuesta planteada. (El instrumento y la tabla de resultados, se muestran en el Anexo 5)

Resultados en % de los atributos de calidad afectados:

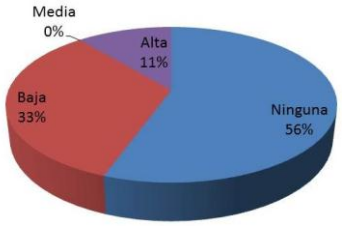

Figura 16: Representación del valor en % del atributo Acoplamiento

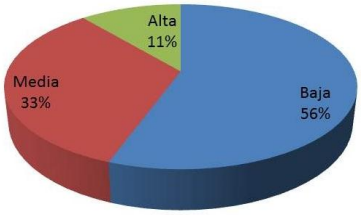

Figura 17: Representación del valor en % del atributo Complejidad de mantenimiento

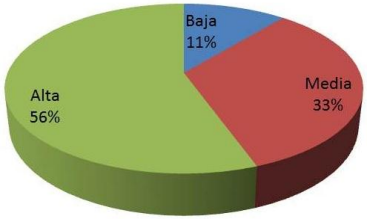

Figura 18: Representación del valor en % del atributo Reutilización

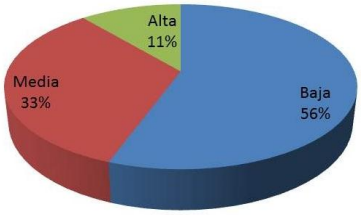

Figura 19: Representación del valor en % del atributo Cantidad de pruebas

Los resultados obtenidos durante la evaluación de las clases y sus respectivas relaciones, mediante los atributos de la métrica RC, demuestran que las clases del diseño poseen un bajo acoplamiento, ya que para este atributo las categorías ninguno y bajo sumaron un 89% del total, mostrando igual por ciento en las categorías alta y media del atributo reutilización. Los atributos complejidad de mantenimiento y cantidad de pruebas, sumaron un 89 % igualmente en las categorías baja y media, lo que demuestra que no es necesario un elevado esfuerzo en el momento de realizar cambios, rectificaciones y pruebas al software.

2.5.1.3 Matriz de inferencia de indicadores de calidad

La matriz de inferencia permite evaluar positiva o negativamente los resultados obtenidos de las relaciones atributo/métrica en una escala numérica, donde el valor 1 representa un resultado "positivo" y el valor 0 "negativo". Si la métrica no influye en el atributo de calidad la relación es considerada como nula y es representada con un guion simple (-). Al promediar por cada atributo las relaciones no nulas, se obtiene un resultado general que al ubicarse en el rango de evaluación (valor <=0.4: "Mala", valor >0.4 y valor<0.7: "Regular", valor>=0.7: "Buena") permite arribar a conclusiones sobre la calidad del diseño propuesto. Teniendo en cuenta que los resultados arrojados con la aplicación de las métricas fueron positivos para todos los atributos, la matriz de la inferencia queda elaborada de la siguiente manera:

Atributos/Métricas	TOC	RC	Promedio
Responsabilidad	1	(-)	1
Complejidad de implementación	1	(-)	1
Reutilización	1	1	1
Acoplamiento	(-)	1	1
Complejidad de Mantenimiento	(-)	1	1
Cantidad de pruebas	(-)	1	1

Tabla 2: Matriz de Inferencia de indicadores de calidad

El promedio general es 1 y teniendo en cuenta el rango de evaluación se concluye que los componentes Documento, Movimiento, AFI, Entrada y No Entrada presenta un diseño aceptable.

2.6 Conclusiones parciales del capítulo.

- La elaboración del Modelo Conceptual permitió establecer los conceptos fundamentales relacionados con los procesos de los AFI y la relación que existe entre ellos.
- La definición de los procesos de negocio permitió comprender cómo funciona el negocio que se desea informatizar.
- La definición de los artefactos de la disciplina Requisitos permitió conocer las interacciones que tendrán los usuarios con el sistema y se identificaron un total de 21 requisitos funcionales, de ellos 5 de complejidad alta, 11 de media y 5 de baja complejidad.
- La elaboración y validación del diseño del sistema a través de las métricas TOC y RC, las cuales arrojaron como resultado que el diseño es simple y presenta una calidad aceptable, ya que el mayor porcentaje de las clases que se implementaron poseen un bajo acoplamiento, baja responsabilidad y complejidad (de esta última tanto de mantenimiento como de implementación), además de ser altamente reutilizables.

CAPÍTULO 3. IMPLEMENTACIÓN Y PRUEBAS

Introducción

En este capítulo se realiza una breve descripción de los estándares a utilizar durante la implementación de la herramienta. También se describen las clases y funcionalidades de dicha aplicación. Se presentan algunas pantallas de la aplicación y algunos de los diseños de casos de pruebas que apoyarán las pruebas de caja negra y los resultados obtenidos de estas.

3.1 Estándar de Codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Los estándares de codificación en el marco del ERP van a permitir una mejor integración entre las líneas de producción y se establecerán las pautas que conlleven a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su mantenibilidad a lo largo del tiempo. (44)

Dentro de este estándar se utilizan las siguientes notaciones:

Notación PascalCasing: Esta notación define que los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula. Ejemplo: NotacionPascalCasing.

Notación CamelCasing: Establece que los identificadores y nombres de variables, métodos y funciones están compuestos por múltiples palabras juntas iniciando cada palabra con letra mayúscula excepto la primera palabra que debe iniciar con minúscula. Ejemplo: notacionCamelCasing.

3.1.1 Nomenclatura de las clases

Los nombres de las clases deben comenzar 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.

Nomenclatura según el tipo de clases

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

Clases de los modelos:

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

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

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

3.1.2 Nomenclatura de las funciones

La nomenclatura 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.

En caso de ser una acción de la clase controladora se especifica el nombre de la acción y seguido el sufijo “Action”. Ejemplo: adicionarDatosDelAfiAction.

3.2 Estructura física del sistema

Cumpliendo con las decisiones arquitectónicas de la dirección del proyecto ERP-CUBA y del equipo de arquitectura se define una estructura contenedora donde van a estar ubicados cada uno de los subsistemas implementados dentro de la aplicación, de manera que facilite la organización y claridad durante el desarrollo. Situadas en la carpeta raíz del sistema Cedrux se en encuentran definidas las carpetas apps y web. La especificación que a continuación se presenta está enfocada al Subsistema de AFI.

Figura 20: Contenido de las carpetas apps y web para AFI

apps: Es la carpeta donde se almacenarán los controladores y el modelo de cada uno de los componentes que le corresponden a los subsistemas.

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

- **ioc:** En él están publicados los servicios que brindan cada uno de los componentes del Subsistema.
- **exepction:** Se define el tipo, idioma y la descripción del mensaje que va a ser mostrado cuando se lance una excepción en el servidor.

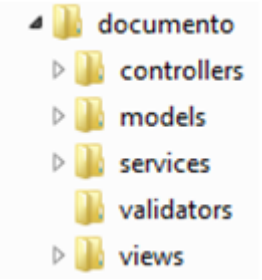

Figura 21: Estructura del Subsistema de AFI dentro de la carpeta apps

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

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

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

services: Esta carpeta contiene las clases que se utilizan para interactuar con los servicios que brinda el módulo, permitiendo el acceso a las funcionalidades desde otros componentes.

validators: Esta carpeta posee las clases de tipo .php que van a realizar acciones de validación en el módulo, como las precondiciones que se deben cumplir antes de que un determinado método sea ejecutado.

views: En esta carpeta se recopilan los ficheros que van a gestionar la capa de presentación, estos ficheros se agrupan en dos carpetas:

- idioma: Contiene ficheros de tipo json que recopilan etiquetas para la gestión de los mensajes vinculados a la presentación.
- scripts: En este directorio se incluyen todas las vistas, para ello se crea una carpeta para cada clase controladora y dentro se incluye la vista o script, archivos de extensión phtml donde se especifica el título de la página que se gestiona y se carga el archivo js que mostrará la presentación.

Figura 22: Estructura del Subsistema de AFI dentro de la carpeta web

css: En este directorio se encuentran las plantillas y estilos para el diseño del módulo.

js: Es la carpeta donde se incluyen las clases javascript, ficheros con la extensión .js donde se encuentra el código correspondiente a la capa de presentación (interfaces de usuario).

3.3 Funcionalidades implementadas

El punto de partida para realizar la gestión de los procesos Apertura y cierre de Apertura, Altas y Bajas de Activos Fijos Intangibles, comienza con la Gestión de Documentos, el cual ofrece las opciones de Adicionar, Modificar, Eliminar, Confirmar, Aprobar, Cancelar y Buscar. La interfaz principal del mismo puede observarse en la Figura 23.

El sistema también permite adicionarle y eliminarle activos fijos intangibles al documento seleccionado mientras se encuentre en estado de “Elaboración” (ver figura 24). Después de adicionarle activos se proceda a gestionarle los datos, donde se definen datos generales y los entrados por el usuario (ver figura 25). Luego se prosigue a confirmar el documento, la cual

pasa a estado de “Preparado”, en caso de ser necesario se puede cancelar el estado de la misma.

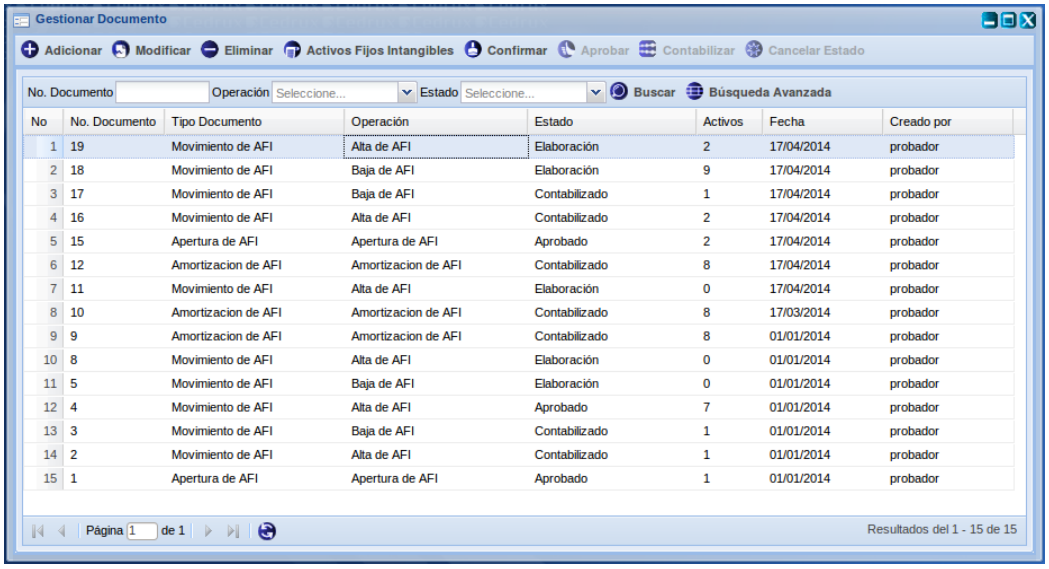

Figura 23: Interfaz principal Gestionar Documento

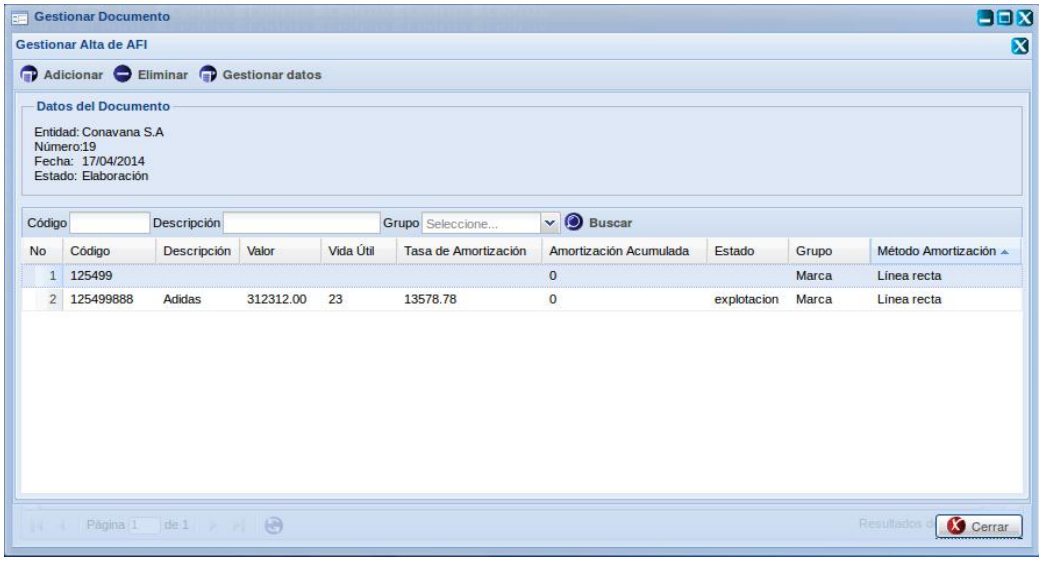

Figura 24: Interfaz Gestionar AFI al Documento de Alta

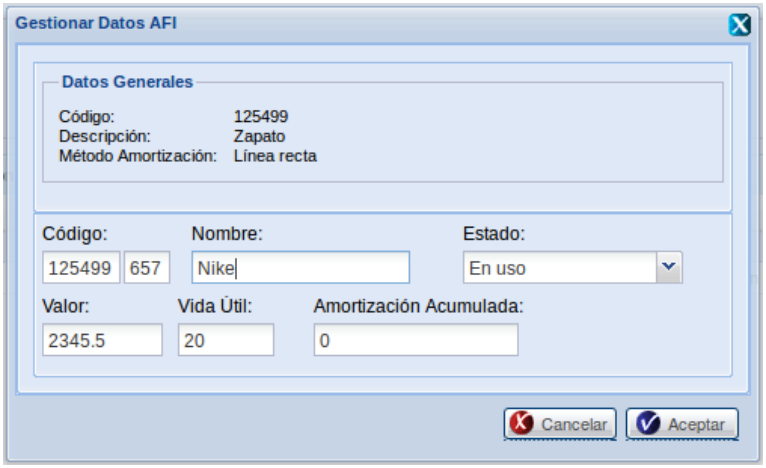

Figura 25: Interfaz Gestionar Datos AFI

3.4 Pruebas de software

3.4.1 Pruebas de Caja blanca

La prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Como es de suponer, es poco práctico realizar una prueba exhaustiva de todos los caminos de un programa. Por ello existen ciertas técnicas para el diseño de este tipo de pruebas que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Entre las más usadas se encuentran las siguientes: (45)

- **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 seleccionan 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 se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe y representa un límite inferior para el número de casos de pruebas que se deben realizar para asegurar que se ejecuta cada camino del programa.

Dentro de la prueba de caja blanca, la técnica que se utilizó fue Camino básico. Para aplicar esta técnica se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de flujo en el cual: (45)

- 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.

Un grafo de flujo está formado por tres componentes fundamentales que ayudan a su elaboración y comprensión, estos brindan información para confirmar que el trabajo se está haciendo adecuadamente. (45)

Componentes del grafo de flujo:

- **Nodo:** Son los círculos representados en el grafo, el cual contiene 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 del Camino básico es preciso calcular la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se muestra el código del método **eliminarDocumentoAfi()** encargado de eliminar un documento seleccionado así como los activos que tiene registrados.

```php
public function eliminarDocumentoAfi($iddocumento, $operacion) {
    $idDatAfi = $this->pIntegrator->movimientoAFI->dameIdAfibyIdDoc($iddocumento);//1
    if ($operacion == 90000000030 || $operacion == 90000000031) {//2
        foreach ($idDatAfi as $value) {//3
            $this->pIntegrator->AFI->eliminarAfibyIdAfi($value['id_dat_afi']);//4
            $this->pIntegrator->movimientoAFI->eliminarbyIdAfibyIdDoc($value['id_dat_afi'], $iddocumento);//4
        }
    } elseif ($operacion == 90000000032 || $operacion == 90000000033) {//5
        foreach ($idDatAfi as $value) {//6
            $this->pIntegrator->AFI->cambiarValorActivobyIdDatAfi($value['id_dat_afi'], 3);//7
            $this->pIntegrator->movimientoAFI->eliminarbyIdAfibyIdDoc($value['id_dat_afi'], $iddocumento);//7
        }
    }
    DatDocAfi::eliminarDoc($iddocumento);//8
    return true;//8
}
```

Figura 26: Método para eliminar documentos

Para el cálculo de la complejidad ciclomática es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la figura 24.

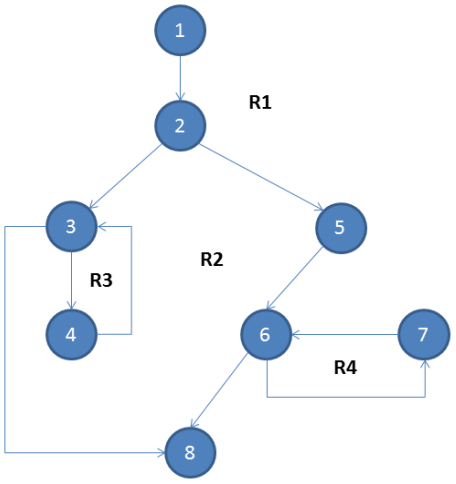

Figura 27: Método para eliminar un documento

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres fórmulas, se debe utilizar el mismo grafo en cada caso:

Fórmula 1: V (G) = (A – N) + 2

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

Resultado:

V (G) = (10-8) + 2

V (G) = 4

Fórmula 2: V (G) = P + 1

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

Resultado:

V (G) = 3+1

V (G) = 4

Fórmula 3: V (G) = R

Donde “R” es la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado:

V (G) = 4

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución:

Camino básico # 1: 1, 2, 3, 8.

Camino básico # 2: 1, 2, 3, 4, 3, 8.

Camino básico # 3: 1, 2, 5, 6, 8.

Camino básico # 4: 1, 2, 5, 6, 7, 6, 8.

Es necesario aclarar que existen otros caminos en el grafo que no son considerados independientes ya que son combinaciones de los anteriormente especificados.

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 el resultado que se espera que devuelva el procedimiento.
- **Resultados:** Se muestra el resultado obtenido.
- **Salida:** Se presenta el valor final.

Camino básico # 1: 1, 2, 3, 8.	
Descripción	Dado un documento eliminar el documento incluyendo los activos que contiene.
Condición de ejecución	Se debe tener el id del documento que se desea eliminar.
Entrada	$iddocumento, $operacion
Resultados esperados	Se elimina el documento satisfactoriamente.
Resultados	Se elimina un documento de entrada satisfactoriamente.
Salida	1

Tabla 3: Caso de prueba para el camino básico #1

Camino básico # 2: 1, 2, 3, 4, 3, 8.	
Descripción	Dado un documento eliminar el documento incluyendo los activos que contiene.
Condición de ejecución	Se debe tener el id del documento que se desea eliminar.
Entrada	$iddocumento, $operacion
Resultados esperados	Se elimina el documento satisfactoriamente.
Resultados	Se elimina el documento de entrada y los activos que contiene.
Salida	1

Tabla 4: Caso de prueba para el camino básico #2

Camino básico # 3: 1, 2, 5, 6, 8.	
Descripción	Dado un documento eliminar el documento incluyendo los activos que contiene.
Condición de ejecución	Se debe tener el id del documento que se desea eliminar.
Entrada	$iddocumento, $operacion
Resultados esperados	Se elimina el documento satisfactoriamente.
Resultados	Se elimina el documento de no entrada satisfactoriamente.
Salida	1

Tabla 5: Caso de prueba para el camino básico #3

Camino básico # 4: 1, 2, 5, 6, 7, 6, 8.	
Descripción	Dado un documento eliminar el documento incluyendo los activos que contiene.
Condición de ejecución	Se debe tener el id del documento que se desea eliminar.
Entrada	$iddocumento, $operacion
Resultados esperados	Se elimina el documento satisfactoriamente.
Resultados	Se elimina el documento de no entrada y los activos que contiene.
Salida	1

Tabla 6: Caso de prueba para el camino básico #4

3.4.1.1 Resultado de la ejecución de las pruebas de Caja blanca

Luego de haber ejecutado el método de Camino básico, se pudo comprobar que el flujo de trabajo de las funcionalidades es correcto, ya que se probó que cada sentencia es ejecutada al

menos una vez con los parámetros de entrada y las salidas esperadas. Con las pruebas de caja blanca se comprobó que la implementación se realizó de manera correcta, se examinó el estado de la aplicación en varios puntos de la misma determinando que el estado real coincidía con el esperado.

3.4.2 Pruebas de Caja negra

Se denominan pruebas funcionales o Functional Testing, a las pruebas de software que tienen por objetivo probar que los sistemas desarrollados, cumplan con las funciones específicas para los cuales han sido creados, es común que este tipo de pruebas sean desarrolladas por analistas de pruebas con apoyo de algunos usuarios finales, esta etapa suele ser la última etapa de pruebas y al dar conformidad sobre esta el paso siguiente es el pase a producción. (46)

A este tipo de pruebas se les denomina también pruebas de comportamiento o pruebas de caja negra, ya que los analistas de pruebas, no enfocan su atención a como se generan las respuestas del sistema, básicamente el enfoque de este tipo de prueba se basa en el análisis de los datos de entrada y en los de salida, esto generalmente se define en los casos de prueba preparados antes del inicio de las pruebas. (46)

Técnica Partición de equivalencia: divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Para verificar que la aplicación se comporta según los requisitos establecidos, se realizan método de caja negra utilizando casos de prueba. En las tablas 7, 8 se especifica el caso de prueba para el requisito Adicionar Datos AFI, los restantes Diseños de casos de prueba pueden ser consultados en el Anexo 6.

Condiciones de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema de Logística.
- Se debe seleccionar la opción Logística/AFI.

- Debe existir un documento en estado de “Elaboración”.
- Debe existir al menos un AFI registrado en el sistema.

Requisitos a probar: Adicionar Datos AFI.

Escenario	Descripción	Descripción	Código	Valor del activo	Estado del activo	Amortización acumulada	Vida útil	Respuesta del sistema	Flujo central
EC 1.1 Adicionar datos del AFI correctamente.	El sistema debe permitir adicionar los datos del AFI.	V (havanos)	V 321	V 200	V (En uso)	V 10	V 10	El sistema muestra el mensaje de información: “Los datos fueron insertados satisfactoriamente.”	Se selecciona el AFI al cual se desea insertar los datos. Se introducen los datos del AFI correctamente. Se presiona el botón Aceptar. Se muestra un mensaje de información.
EC 1.2 Adicionar datos del AFI incorrectamente.	El sistema debe permitir adicionar los datos del AFI.	I vacío	I (havanos)	I (hii)	I (vacío)	I (vsvs)	I hoaaa	El sistema no permite insertar datos incorrectos.	Se selecciona el AFI al cual se desea insertar los datos. Se introducen los datos del AFI con algunos valores incorrectos.
EC 1.3	Se	I	I	I	I	I	I	El sistema	Se

Adicionar datos del AFI dejando campos obligatorios vacío.	introducen campos vacíos.	(Vacío)	(Vacío)	(Vacío)	(Vacío)	(Vacío)	(Vacío)	muestra el mensaje de información: "Existen campos vacíos."	selecciona el AFI al cual se desea insertar los datos. Se introducen los datos del AFI dejando campos vacíos. Se presiona el botón Aceptar. Se muestra un mensaje de información.
EC 1.4 Adicionar código del AFI ya existente.	Se introduce un código ya existente en el sistema.	V popular	V 321	V 200	V en uso	V 0	V 5	El sistema muestra el mensaje de información: "Ya existe un activo con este código".	Se selecciona el AFI al cual se desea insertar los datos. Se introduce el código de un AFI ya existente. Se presiona el botón Aceptar. Se muestra un mensaje de información.
EC 1.5 Cancelar	Se cancela la operación	N/A	N/A	N/A	N/A	N/A	N/A	El sistema cierra la ventana y se cancela la operación.	Se presiona el botón Cancelar.

Tabla 7: Requisitos a probar correspondiente al DCP Adicionar Datos AFI

Descripción de variables

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Descripción	Campo de texto	No	Letras y número (no permite Caracteres especiales)
2	Código	Campo de texto	No	Números
3	Amortización acumulada	Campo de texto	No	Números
4	Valor del activo	Campo de texto	No	Números
5	Estado del activo	Lista desplegable	No	Objetos de la lista
6	Vida útil	Campo de texto	No	Números

Tabla 8: Descripción de las variables correspondiente al DCP Adicionar Documento

3.4.2.1 Resultados de las pruebas de caja negra

Las pruebas de caja negra que se le realizaron a la solución mediante los DCP y utilizando la técnica Partición de equivalencia fueron desarrolladas por los Especialistas de la Línea de Logística de CEIGE. Donde se obtuvieron en la primera iteración un total de 26 No conformidades, quedando finalmente resueltas para la segunda iteración, donde se detectaron 7 No conformidades, las cuales fueron resueltas y en la tercera iteración se comprobó que la aplicación estaba libre de No conformidades. Todo lo anterior dio paso a la liberación de la aplicación, donde se firmó el acta de aceptación por parte del equipo de desarrollo y los Especialistas de la Línea Logística de CEIGE. (Ver Anexo 7)

3.5 Conclusiones parciales del capítulo.

- La definición de los estándares de codificación facilitó la lectura y comprensión del código.
- La estructura física del sistema permitió la organización durante el desarrollo.
- Se aplicaron las pruebas de caja blanca mediante la técnica de Camino Básico para de esta manera revisar el código, y verificar que el sistema detecte la mayor cantidad de errores que existan.
- De igual manera se ejecutaron pruebas de caja negra mediante la técnica de partición de equivalencia donde se validó que el sistema funciona de forma correcta, respondiendo a los requisitos funcionales aprobados.

Conclusiones

La realización del presente trabajo y los resultados obtenidos en el mismo permitieron arribar a las siguientes conclusiones:

- El análisis de los principales conceptos relacionados con los procesos de los AFI, propició conocer qué es un AFI y qué son los procesos de los AFI, además de la identificación de los elementos teóricos necesarios para guiar el desarrollo de la investigación.
- La definición de los procesos de negocio y los artefactos de la disciplina Requisitos permitieron conocer cómo funciona el negocio a informatizar y las interacciones que tendrán los usuarios con el sistema.
- La elaboración del diseño propuesto, el cual fue comprobado a través de las métricas de validación del diseño TOC y RC permitió la correcta implementación de las funcionalidades del sistema, evidenciado en la presencia de valores positivos en los indicadores de calidad medidos por cada métrica.
- La realización de las pruebas permitió probar que el sistema cumpla con las funcionalidades específicas para las que fue creado, respondiendo a los requisitos funcionales aprobados.

Recomendaciones

- Continuar con la realización de pruebas funcionales que permitan garantizar una mayor calidad del producto.
- Integrar el subsistema de AFI con el de Contabilidad.
- Realizar el resto de los procesos de los AFI, para completar el Subsistema de AFI del Sistema Cedrux.

Referencias bibliográficas

1. La importancia del capital intangible en las sociedades de la información. [En línea] [Citado el: 20 de marzo de 2013.] http://agenciaescrow.com/es/agencia-escrow.

2. Normas Internacionales. NIC 38. [En línea] [Citado el: 21 de octubre de 2013.] http://www.normasinternacionalesdecontabilidad.es/nic/pdf/NIC38.pdf.

3. F.A.C.P.C.E. Resolución Técnica Nro. 9. [En línea] [Citado el: 21 de octubre de 2013.] http://www.economicas-online.com/docs/rt09.htm.

4. Definición de activo fijo intangible. [En línea] [Citado el: 21 de octubre de 2013.] http://www.eumed.net/cursecon/dic/ctc/A.htm.

5. Activos Fijos Intangibles. [En línea] [Citado el: 21 de octubre de 2013.] http://www.monografias.com/trabajos29/activos-intangibles/activos-intangibles.shtml.

6. La Real Academia Española. [En línea] [Citado el: 27 de mayo de 2014.] http://lema.rae.es/drae/srv/search?id=FRMW2SMo32x5Y3FtnZh).

7. Openbravo. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.slideshare.net/mikelrubio16/open-bravo-15704538.

8. Openbravo ERP. [En línea] [Citado el: 10 de febrero de 2014.] http://www.golivepyme.com/openbravo-modulo-finanzas-contabilidad.aspx.

9. SAP España - SAP - Sistema de software de gestión de pedidos de cliente. [En línea] [Citado el: 29 de enero de 2014.] http://global.sap.com/spain/software/gestion_de_pedidos.epx.

10. Sistema ASSET. [En línea] [Citado el: 22 de octubre de 2013.] http://www.assets.co.cu/.

11. Versat Sarasola. [En línea] [Citado el: 21 de octubre de 2013.] http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc.

12. Ing. William González Obregón. Modelo de desarrollo de software v1.2. La Habana : s.n., 2012.

13. Modelando Procesos. [En línea] [Citado el: 15 de noviembre de 2013.] http://www.softein.com/publico/Modelando_procesos_01.pdf.

14. Lenguaje de Modelado Unificado "UML". [En línea] [Citado el: 2 de febrero de 2014.] http://www.uml.org/.

15. **López, Patricia.** *Ingeniería de Software, Herramienta CASE Visual Paradigm.* Cantabria : s.n.

16. Mozilla Firefox. [En línea] [Citado el: 4 de noviembre de 2013.] https://www.mozilla.org/es-CL/firefox/features/.

17. NetBeans IDE Features. [En línea] [Citado el: 15 de diciembre de 2013.] http://netbeans.org/features/index.html.

18. Servidor de Aplicación, Apache. [En línea] [Citado el: 4 de noviembre de 2013.] http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/.

19. PgAdmin. [En línea] [Citado el: 20 de diciembre de 2013.] http://www.pgadmin.org/visualtour16.php.

20. PostgreSQL. [En línea] [Citado el: 2 de noviembre de 2013.] http://www.postgresql.org.es/sobre_postgresql.

21. Subversion. [En línea] [Citado el: 2 de noviembre de 2013.] http://www.lyx.org/WebEs.HowToUseSVN.

22. ¿Qué es AJAX? [En línea] [Citado el: 20 de noviembre de 2013.] http://elmasterdelaweb.wikispaces.com/file/view/QUE+ES+AJAX.pdf.

23. Jeremías Morales. SlideShare.net. [En línea] [Citado el: 7 de noviembre de 2013.] http://www.slideshare.net/JeremiasMorales.

24. Manual XHTML. [En línea] [Citado el: 27 de noviembre de 2013.] http://manual-xhtml.blogspot.com/2006/05/primer-documento-xhtml.html#L3154.

25. Desarrollo Web. [En línea] [Citado el: 15 de noviembre de 2013.] http://www.desarrolloweb.com/wiki/json.html.

26. Guía Breve de CSS. [En línea] [Citado el: 27 de noviembre de 2013.] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo.

27. **Pupo, Yanisleydi Cañete.** *Libro de ayuda del marco de Trabajo Sauxe, en su versión 2.0.* Universidad de las Ciencias Informática, Ciudad de la Habana : s.n., 2010.

28. Ext JS. [En línea] [Citado el: 20 de noviembre de 2013.] http://es.scribd.com/doc/100415898/SenchaExt-JS.

29. Zend Framework. [En línea] [Citado el: 27 de noviembre de 2013.] http://www.maestrosdelweb.com/editorial/guia-zend/.

30. Doctrine. [En línea] [Citado el: 27 de noviembre de 2013.] http://www.doctrine-project.org/projects/orm.html.

31. **ARIAS PATIÑO, J. D. y ISAZA, LOZANO.** Desarrollo de un sistema computacional para el aprendizaje de programación estructurada" SCAPE". 2011.

32. PHP. [En línea] [Citado el: 15 de noviembre de 2013.] http://ivancamayo.files.wordpress.com/2010/09/php1.pdf.

33. **ABAD, M. P. G.-M. y COLLADO, D. L.**, et al. *An Open Cloud Computing Interface (OCCI) management console for the OpenNebula Toolkit.* 2010.

34. **Méndez, Gonzalo.** *Ingeniería de Requisito.* Madrid : s.n., 2009.

35. Pressman, Roger. Ingeniería de Software, un enfoque práctico. s.l. : McGraw-Hill Companies, 2002. Quinta edición.

36. *M.J. Escalona y N. Koch. Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo.* España y Alemania : s.n. TIC2000-1673C06-03.

37. Software Engineering Institute. [En línea] [Citado el: 10 de mayo de 2014.] http://www.sei.cmu.edu/architecture/.

38. **Larman, C.** *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Madrid : Pearson Educación S.A 2da edición, 2003.

39. Sommerville, Ian. Ingenieria de Software 7ma Edición.

40. **Guerrero, Carlos Miguel: Carlos Vingut.** *Diseño e implementación de una herramienta para la gestión de credenciales en el subsistema capital humano del sistema integral de gestión Cedrux.* La Habana : s.n., 2011.

41. Estructura de las Aplicaciones Orientadas a Objetos. [En línea] [Citado el: 27 de noviembre de 2013.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf.

42. **GROSSO, A.** Algo de patrones GRASP... Patrones GRASP . [En línea] 2011. [Citado el: 27 de noviembre de 2013.] http://www.practicasdesoftware.com.ar/2011/03/patrones-grasp/.

43. Patrones de diseño. Patrones GOF. [En línea] [Citado el: 27 de noviembre de 2013.] http://infow.wordpress.com/2013/01/30/gof-patrones-de-diseno-ii-que-son/.

44. Normas y estándares de Codificación del ERP. La Habana : s.n., 2008.

45. Pressman, Roger. Ingeniería de Software, un enfoque práctico. s.l. : McGraw-Hill Companies, 2005, Sexta edición.

46. **B, Ing. Alexander Oré.** Calidad y software. [En línea] [Citado el: 25 de abril de 2014.] http://www.calidadysoftware.com/testing/pruebas_funcionales.php.

Bibliografía consultada

1. Palmero, Miguel Angel Sanchez. Modelado de Negocio y Levantamiento de Requisitos del subsistema Activos Fijos Intangibles del sistema Cedrux. Ciudad de la Habana: s.n., junio 2009.

2. Definición de activo fijo intangible. [En línea] [Citado el: 21 de octubre de 2013.] http://www.eumed.net/cursecon/dic/ctc/A.htm.

3. Activos Fijos Intangibles. [En línea] [Citado el: 21 de octubre de 2013.] http://www.monografias.com/trabajos29/activos-intangibles/activos-intangibles.shtml.

4. F.A.C.P.C.E. Resolución Técnica Nro. 9. [En línea] [Citado el: 21 de octubre de 2013.] http://www.economicas-online.com/docs/rt09.htm.

5. Openbravo. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.slide share.net/mikelrubio16/open-bravo-15704538.

6. Sistema ISIS. [En línea] [Citado el: 25 de noviembre de 2013.] http://www.sistemaisis.com/gestion-contador.html.

7. Sistema ASSET. [En línea] [Citado el: 22 de octubre de 2013.] http://www.assets.co.cu/.

8. Versat Sarasola. [En línea] [Citado el: 21 de octubre de 2013.] http://www.forum.villaclara.cu/UserFiles/forum/PonenciasWORD/0500691.doc.

9. Ing. William González Obregón. Modelo de desarrollo de software v1.2. La Habana : s.n., 2012.

10. G., Dr. Guillermo. *ACTIVOS INTANGIBLES.* Argentina : s.n., 2005.

11. Torre, Anibal de la. Lenguajes del lado servidor o cliente. [En línea] [Citado el: 23 de noviembre de 2013.] http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

12. Los Activos Intangibles dentro del Contexto de la Sociedad del Conocimiento: El Reto. Citraro, Leonidas Torres. 13, Venezuela: Redalyc, 2010

13. Garcia, Yanira Ivhet Perez. Valoración de activos intangibles. 2011.

14. Autores, Colectivo de. Valor económico agregado e intangibles. 2001.

15. Español, Guillermo G. Activos Intangibles Informe Nro 23 A-.

16. Una aproximación empírica al efecto de los activos intangibles sobre la estructura financiera en la empresa española. Garcia-Meca, Emma. s.l.: Revista Europea de Dirección y Economía de la Empresa, 2009, Vol. 18.

17. Sitio oficial del Ministerio de Finanzas y Precios. [En línea] [Citado el: 18 de noviembre de 2013.] www.mfp.cu/html_mfp/legt.php.

18. SAP-R3 [En línea] [Citado el: 20 de febrero de 2014.]http://www.oocities.org/espanol/emoly188/que_es_sap_r3.htm

19. Colectivo de autores. Rodas XXI. Rodas XXI. [En línea] [Citado el: 21 de octubre de 2013.] http://www.rodasxxi.cu/rodasxxi.php.

20. Sommerville, Ian. Ingenieria de Software 7ma Edición.

21. Fundamentos de Ingeniería de Software. [En línea] [Citado el: 25 de noviembre de 2013.] http://fundamentosoftware.blogspot.com/p/introduccion-al-modelado.html.

22. Agut, Raúl Monferrer. Especificación de Requisitos de Software según el estándar IEEE 830. 2001

23. ¿Qué es PHP? [En línea] [Citado el: 25 de noviembre de 2013.] http://php.net/manual/es/intro-whatis.php.

24. PRESSMAN, R. S. Ingeniería de Software. vol. Quinta Edición.

25. Pérez, Javier Eguíluz. Introducción a JavaScript.

26. El servidor de web Apache: Introducción práctica. [En línea] [Citado el: 25 de noviembre de 2013.] http://acsblog.es/articulos/trunk/LinuxActual/Apache/html/index.html.

27. Tutorial de PostgreSQL. [En línea] [Citado el: 25 de noviembre de 2013.] http://palomo.usach.cl/Docs/postgres/Postgres-Tutorial.pdf.

28. NetBeans. [En línea] [Citado el: 17 de noviembre de 2013.] http://netbeans.org.

29. Visual-paradigm. [En línea] [Citado el: 18 de diciembre de 2013.] http://www.visual-paradigm.com.

30. Torre, Anibal de la. Lenguajes del lado servidor o cliente. [Online] [Cited: noviembre 23, 2013.]http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html.

31. G., Dr. Guillermo. ACTIVOS INTANGIBLES. Argentina: s.n., 2005.

32. **Rolando Alfredo Hernández León, Sayda Coello González.** *EL PARADIGMA CUANTITATIVO DE LA INVESTIGACIÓN CIENTIFICA.* Ciudad de la Habana : EDUNIV, 2002. ISBN: 959-16-0343-6.