Universidad de las Ciencias Informáticas

Facultad 3

Centro de Informatización de Entidades (CEIGE)

Desarrollo de los componentes nomenclador, configuración y amortización para el módulo Activo Fijo Intangible del Sistema Integral de Gestión Xedro-ERP

Autores:
Yeni Veliz Alvarez
Geidis Arévalo Quintana

Tutores:
Ing. Annia Verdecia Boza
Ing. Carlos Alberto Giralt
Ing. Adrián Sánchez

La Habana, Junio del 2014

Año 56 de la Revolución

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.

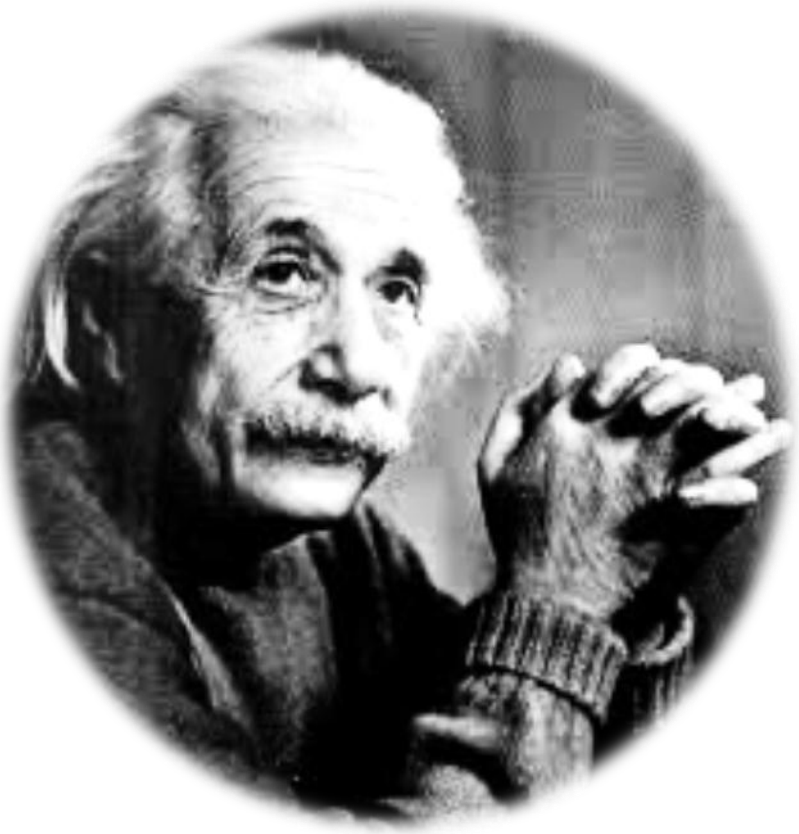
Para que así conste, firmamos la presente a los ____ días del mes de ___ del año _____.

Yeni Veliz Álvarez
(Autora)

Geidis Arévalo Quintana
(Autor)

Ing. Annia Verdecia Boza

Ing. Carlos Alberto Giralt Torriente



Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

Albert Einstein

Agradecimientos

Yeni Veliz Alvarez

A nuestro Comandante en Jefe Fidel Castro Ruz, por darnos la oportunidad de estudiar en la Universidad de las Ciencias Informáticas.

A mis padres por esfuerzo y dedicación.

A mi prima Baniueska por ser mi principal motivo a seguir adelante.

A los profesores que han contribuido a nuestra formación en todos estos años.

A nuestros compañeros por su invaluable ayuda y las siempre bien recibidas enseñanzas.

A todas aquellas personas que con su esfuerzo han hecho posible la realización del presente trabajo, en especialmente a mi compañero de tesis.

A nuestros tutores Annia y Carlos, guiándonos en todo momento.

Geider Arevalo Quintana

Les agradezco a todos mis amigos de la universidad que siempre estuvieron conmigo en los buenos y malos momento.

A los profesores que a lo lago de la carrera fueron mis guías.

A mi familia por siempre darme su apoyo.

Y por último a las personas más importantes en mi vida mi papá, mi hermano y mi mamá.

¡Muchas Gracias!

Dedicatoria

Yeni Veliz Alvarez

Los sueño son una de las mejores experiencia que han experimentado el ser humano, por ello le quiero dedicar esta este trabajo íntegramente a mi prima Baniuska, a ella soñaba con graduarse y ser una buena profesional pero lamentablemente la vida no se lo permitió.

A mis padres y a mi hermanito que han sido mi fuerza, mi espíritu y mis ganas de seguir adelante en la vida.

A mi novio que ha ocupado un papel fundamental en los últimos meses de mi carrera.

A mis amigaos Danay, Yoandris, la china, Maricet y Anabel que a lo largo de la carrera siempre me han apoyado incondiconalmente, especialmente a mi compañero de tesis.

Geider Arevalo Quinatana

Dedico este trabajo a todas las personas que siempre creyeron en mí, en especial: A toda mi familia, mi mamá, mi hermano y mi papá por su apoyo incondicional y su preocupación constante…

A todos mis amigos… A mis tutores y profesores que me han guiado durante estos cinco años…

A quien lo lea…

Resumen

En los últimos años los Activo Fijos Intangibles han tomado gran auge a escala mundial. Estos en Cuba son un tema relativamente nuevo donde muchas entidades los subvaloran tratándolos erróneamente como gastos y carentes de beneficios. Actualmente la Universidad de las Ciencias Informáticas trabaja en la creación de un sistema integral de gestión denominado Xedro-ERP.

Esta solución aún no abarca todas las áreas de las empresas. Siendo los Activos Fijos Intangibles (AFI) una tarea pendiente en la informatización. Por todo esto el presente trabajo tiene como objetivo desarrollar los componentes nomenclador, configuración y amortización para el módulo AFI del Sistema Integral de Gestión Xedro-ERP.

Para guiar el desarrollo de la solución se utiliza el modelo de desarrollo establecido por el Centro de Informatización para la Gestión de Entidades de la UCI. Con el fin de lograr un mejor entendimiento de los procesos de los AFI, se realiza un estudio de algunas de las herramientas utilizadas para el control de estos a escala nacional y global.

A partir de este estudio se llevó a cabo el modelado del negocio. Se definieron los requisitos funcionales y no funcionales de la solución. Se realizó el diseño e implementación de los componentes nomenclador, configuración y amortización permitiendo la gestión y control de los AFI en el sistema integral Xedro-ERP. Por último, se aplicaron pruebas de caja blanca y caja negra para comprobar el correcto funcionamiento de los componentes implementados así como el ajuste a las características y necesidades de las entidades cubanas.

Palabras claves: activo fijo intangible, amortización.

Índice

Índice de figuras

Índice de tablas

Introducción

En los últimos años los Activos Fijos Intangibles (AFI) se han convertido en un elemento esencial para las empresas, debido a que estas enfrentan la necesidad de avanzar hacia modelos basados en la innovación, las nuevas tecnologías y el conocimiento. Como consecuencia del nuevo desarrollo económico regido por la competitividad, el fuerte protagonismo del mercado y el creciente interés por el conocimiento, se le ha dado mayor importancia a estos recursos.

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.”(1).

En Cuba los activos fijos intangibles son un tema relativamente nuevo, debido a que muchas entidades aún se rigen por la contabilidad tradicional, la cual no proporciona la información suficiente en relación con su reconocimiento y valoración, tratándolos erróneamente como gastos y carentes de beneficios.

La economía cubana ha evolucionado paulatinamente al estar inmersa en un proceso de perfeccionamiento. Para dar cumplimiento a este objetivo, el país se ha planteado la necesidad de informatizar los procesos de gestión de las entidades presupuestadas y empresariales, utilizando sistemas que se ajusten a las necesidades de la sociedad. En tal sentido la Universidad de las Ciencias Informáticas (UCI) juega un papel fundamental en el desarrollo de soluciones informáticas. El Centro de Informatización para la Gestión de Entidades (CEIGE) es uno de los centros de la UCI que se encarga del desarrollo de software para la gestión empresarial, este se ha trazado la meta de informatizar dentro de su producto Xedro-ERP, los procesos de negocio de las empresas cubanas.

Los procesos empresariales que contiene Xedro-ERP se encuentran agrupados en varios subsistemas, entre los que se encuentra el subsistema Logística, el cual permite la gestión de los activos fijos tangibles, los inventarios y la facturación. Actualmente, este subsistema no garantiza la gestión de los procesos relacionados con los activos fijos intangibles, provocando que no se pueda nomenclar, configurar, ni gestionar la amortización sufrida por estos activos, una vez que hayan perdido su valor. Lo antes planteado trae como consecuencia que las entidades que utilizan este sistema para guiar sus procesos empresariales no puedan alcanzar una correcta financiación,

reducir los costos económicos, ni utilizar nuevas vías de creación de ingresos. A su vez provoca la pérdida de tiempo pues el control de estos activos se realiza manualmente. Esta lentitud en el proceso de gestión, se debe a la existencia de un gran cúmulo de información. La no existencia de control sobre el registro de estos AFI, propicia la falta de conocimiento de la existencia real de estos y del beneficio que pudieran brindar.

Partiendo de lo antes expuesto, se propone como **problema a resolver**: ¿Cómo incorporar los componentes nomenclador, configuración y amortización en el módulo Activos Fijos Intangibles del Sistema Integral de Gestión Xedro-ERP?

Se define como **objeto de estudio** la gestión de los procesos de activos fijos intangibles, enmarcado en el campo de acción, la gestión de los procesos de activos fijos intangibles en las empresas cubanas.

Se presenta como **objetivo general**, desarrollar los componentes nomenclador, configuración y amortización para el módulo Activos Fijos Intangibles del Sistema Integral de Gestión Xedro-ERP.

Para darle cumplimiento al objetivo general se trazaron los siguientes **objetivos específicos**:

- Elaborar la fundamentación teórica a partir del estudio de las leyes y procesos de los activos fijos intangibles.
- Desarrollar la disciplina Modelado del negocio y Requisitos para los componentes nomenclador, configuración y amortización del módulo activos fijos intangibles.
- Realizar la disciplina Análisis y diseño de los componentes nomenclador, configuración y amortización del módulo activos fijos intangibles.
- Implementar los componentes nomenclador, configuración y amortización del módulo activos fijos intangibles.
- Validar la solución propuesta.

Se propone como **idea a defender** que si se desarrollan los componentes nomenclador, configuración y amortización para el módulo Activos Fijos Intangibles, se contribuirá a la integración de estos componentes al Sistema Integral de Gestión Xedro-ERP.

Para cumplimentar los objetivos específicos, se define las siguientes **tareas de la investigación**:

1. Entrevista a los especialistas de la línea con el objetivo de comprender los procesos de negocio de los activos intangibles.
2. Definición del marco teórico para definir los principales conceptos asociados al campo de acción y al objetivo de investigación.
3. Análisis de sistemas ERP relacionado con los activo fijo intangibles a nivel internacional y nacional.
4. Desarrollo del artefacto Modelo conceptual.
5. Descripción de los procesos de negocio identificados.
6. Elaboración del artefacto Mapa de procesos.
7. Identificación de los requisitos funcionales a partir de los procesos de negocio modelados y técnicas para la captura de requisitos.
8. Desarrollo del artefacto Descripción de requisitos.
9. Desarrollo de los Prototipos de interfaz de usuario para cada requisito.
10. Validación de los requisitos funcionales.
11. Desarrollo del artefacto diagrama de clases del diseño.
12. Desarrollo del artefacto Modelo de dato.
13. Desarrollo del artefacto Diagrama de componentes.
14. Desarrollo del artefacto Diseño de casos de prueba.
15. Validación del diseño.
16. Implementación del componente configuración para el módulo Activo Fijo Intangible.
17. Implementación del componente nomenclador para el módulo Activo Fijo Intangible.
18. Implementación del componente amortización para el módulo Activo Fijo Intangible.
19. Realización de pruebas internas del proyecto mediante las pruebas de caja negra.
20. Realización de pruebas de caja blanca para validación del código.

Métodos de investigación:

Métodos Teóricos:

Histórico–lógico: se utilizó para comprobar teóricamente como se ha comportado los procesos relacionados con los AFI durante su desarrollo y evolución hasta la actualidad, destacando sus principales ventajas y desventajas.

Analítico–sintético: se utilizó para realizar un análisis de los elementos más importantes relacionados con los AFI en las entidades cubanas en función de lograr un mayor entendimiento del negocio.

Modelación: es empleado en la fase de análisis y diseño para representar de forma visual las características estructurales de la solución mediante todos los artefactos establecidas por el modelo de desarrollo.

Métodos Empíricos:

Monitoreo de proyectos: este método permitió observar el comportamiento del sistema Assets utilizado en el departamento de Contabilidad de la UCI.

El presente trabajo está estructurado en 3 capítulos, a continuación se realiza una breve explicación de cada uno de ellos:

Capítulo 1. Fundamentación teórica. En este capítulo se hace un análisis sobre los conceptos más relevantes con el objetivo de lograr un mayor entendimiento del negocio. Se realiza un estudio de sistemas nacionales y extranjeros que tengan relación con los activos intangibles, así como las diferentes herramientas, tecnologías y lenguajes a utilizar en el desarrollo de la solución.

Capítulo 2. Disciplina de negocio, requisitos, análisis y diseño. En este capítulo se describe la propuesta de solución del sistema a desarrollar y los procesos que serán informatizados, así como los requisitos funcionales y no funcionales con los que debe cumplir el sistema. Se describen las técnicas empleadas tanto en la captura como en la validación de los requisitos y los diferentes patrones arquitectónicos y de diseño utilizados en la aplicación. Además se realiza el modelo de datos del sistema y se elaboran los artefactos y diagramas de los procesos.

Capítulo 3. Disciplina de implementación y prueba. En este capítulo se muestra la situación física de los distintos componentes lógicos desarrollados a través del modelo de despliegue y la organización del sistema mediante el modelo de componentes. Se explican los estilos de programación y estándares de codificación empleados. Y por último se valida el sistema desarrollado aplicándole las pruebas de caja negra y de caja blanca para validar los requisitos del cliente.

Capítulo 1. Fundamentación teórica

En el presente capítulo se abordan los conceptos y las características relacionadas con los AFI, conjuntamente con un marco legal que describe los principios de cada una de las leyes, legislaciones y normas estudiadas. Para un mayor entendimiento del negocio se realiza un estudio de sistemas nacionales y extranjeros que comprendan algunas de las funcionalidades de los activos intangibles, además de las diferentes tecnologías, lenguajes y herramientas para sentar las bases de la implementación.

1.1 Elementos asociados al dominio del problema

Sistema de Planificación de Recursos Empresariales (ERP, por sus siglas en inglés): es un paquete de software que permite administrar todos los procesos operativos de una empresa, integrando varias funciones de gestión en un único sistema, representa la "columna vertebral" de una empresa (2).

La **logística** es una función operativa que comprende todas las actividades y procesos necesarios para la administración estratégica del flujo y almacenamiento de materias primas, componentes y activos fijos, existencias en proceso y productos terminados; de tal manera, que éstos estén en la cantidad adecuada, en el lugar correcto y en el momento apropiado (3).

Disímiles son los conceptos que definen a los **AFI**, tales como:

- Los AFI son aquellos que no tienen una naturaleza corpórea, es decir, no se pueden ver ni tocar y se caracterizan por ser un activo no monetario (4).
- 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 (1).
- Se definen los AFI como el conjunto de bienes inmateriales, representados en derechos, privilegios o ventajas de competencia que son valiosos porque contribuyen a un aumento en ingresos o utilidades por medio de su empleo en el ente económico; estos derechos se compran o se desarrollan en el curso normal de los negocios (5).

Con el propósito de utilizar un concepto claro y preciso, se toma el concepto propuesto por la Ley 38 de la Normal Internacional de la contabilidad que define a un AFI como: "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" (1).

De este concepto se distinguen tres condiciones para el reconocimiento de un intangible: identificabilidad, control y beneficios económicos futuros. Todo activo que se pueda alquilar, vender, disponer, etc. por separado es identificable, pero no todo activo **identificable** es separable. Un activo no es separable si genera beneficios económicos solo en combinación con otros activos. Sin embargo, la empresa puede distinguir los beneficios económicos que proceden del activo en cuestión, en cuyo caso el activo será identificable. El **control** del activo se refiere al poder de obtener los beneficios económicos futuros que procedan de ese activo y la posibilidad de restringir a terceras personas ese beneficio. Esto puede darse por disposiciones legales o no. Los **beneficios económicos futuros** están referidos tanto a ingresos como a ahorro de costos (6).

Configuración

Según la Real Academia Española (RAE) la configuración es la disposición de las partes que componen una cosa y le dan su peculiar forma y propiedades anejas (7). En el presente trabajo investigativo la configuración se utiliza para guiar el elemento general de la entidad, específicamente la amortización donde debe de permitir el tipo y método de amortización de la misma.

Nomencladores

El nomenclador o clasificador constituye un conjunto de conceptos generales definidos que permiten distinguir y formar agrupaciones de conceptos de orden en particular, buscando establecer una clasificación flexible y ordenada (8). En el presente trabajo se definen los nomencladores con el objetivo de crear la base para ejecutar los diferentes procesos de los AFI.

Amortización

La amortización es una reducción en el valor de los activos o pasivos para reflejar en el sistema de contabilidad cambios en el precio del mercado u otras reducciones de valor (9). Para determinar a qué tipo de activos intangibles se les debe aplicar el proceso de amortización es preciso establecer si tienen vida útil limitada o no. Los activos intangibles, cuya vida útil es ilimitada o indefinida se mantienen intactos hasta que pierdan su valor. En este caso, el costo de los intangibles debe amortizarse de una manera sistemática durante su vida útil estimada.

Existen varios métodos para efectuar el cálculo de la amortización tales como: método lineal o constante, método del porcentaje constante sobre valores contables decrecientes, método de los números dígitos, métodos basados en la actividad y método basado en las tablas fiscales (10).

El presente trabajo solo empleará el método lineal para calcular la tasa de amortización, como su propio nombre indica las cuotas de amortizaciones son constantes. A continuación se muestra la fórmula para calcular la tasa de amortización aplicando el método de línea recta:

Tasa de amortización = Valor inicial / Vida útil

Valor inicial: Es el costo de adquisición del activo.

Vida útil: Los activos intangibles pueden tener una vida útil limitada e ilimitada. Se dice que es limitada cuando toma el valor máximo del activo e ilimitada con valor igual cero.

1.2 Marco legal

A continuación se muestra un listado con todas las leyes y legislaciones estudiadas que permitieron un mayor entendimiento del negocio:

Declaraciones legales en las creaciones UCI (IPL-4020:2009): tienen como objetivo establecer las normas generales para realizar y ubicar las declaraciones de carácter legal en las creaciones intelectuales de la UCI, haciendo referencia a: titularidad y año de liberación, reservación de derechos del titular, utilización de productos de terceros y legal (11).

Registro de la Propiedad Intelectual (IPP-4000:2008): tiene como objetivo registrar legalmente las creaciones intelectuales que forman parte del patrimonio de la UCI, entre las que se encuentran: productos de software (soluciones, componentes, librerías, marcos de trabajo, funcionalidades similares a los anteriores), bases de datos, marcas, diseños de comunicación visual, autorías DVD, obras fotográficas, obras literarias, obras científicas (informes de investigaciones, metodologías, modelación de negocios, programas de formación, entre otros), audiovisuales, obras de arte y entre otras obras (12).

Decreto-Ley 203 De Marcas y Otros Signos Distintivos: tiene por objeto la protección de las marcas, los nombres comerciales, los emblemas empresariales, los rótulos de establecimiento y los lemas comerciales en Cuba, a través de la concesión de derechos de propiedad industrial (13).

Norma Internacional de Contabilidad 38: el objetivo de esta norma es recomendar el tratamiento contable para los activos intangibles que no estén contemplados específicamente en otra norma.

Esta norma requiere que una entidad reconozca un activo intangible si, y sólo si, se cumplen ciertos criterios. La norma también especifica cómo determinar el valor en libros de los activos intangibles, y exige la revelación de información específica sobre activos intangibles (1).

Norma Internacional de Contabilidad 36: el objetivo de esta norma consiste en establecer los procedimientos que una entidad aplicará para asegurarse de que sus activos están contabilizados por un importe que no sea superior a su importe recuperable. Un activo estará contabilizado por encima de su importe recuperable cuando su importe en libros exceda del importe que se pueda recuperar del mismo a través de su utilización o de su venta. Si este fuera el caso, el activo se presentaría como deteriorado, y la norma exige que la entidad reconozca una pérdida por deterioro del valor de ese activo. En la norma también se especifica cuándo la entidad revertirá la pérdida por deterioro del valor, así como la información a revelar (14).

1.3 Estudio de sistemas existentes

En la actualidad existen varios sistemas de gestión que ayudan a realizar con éxito todos los procedimientos que propician el control de los activos fijos intangibles en las empresas. A continuación se describen algunos de los sistemas más utilizados a escala global y nacional para el trabajo con los AFI.

1.3.1 Sistemas Internacionales

ASSETS es un sistema de gestión integral que permite el control de los procesos de Compras, Ventas, Producción, Taller, Inventario, Finanzas, Contabilidad, Presupuesto, Activos Fijos, Útiles Herramientas y Recursos Humanos. El Módulo de Activos Fijos le permite actualizar al cierre del mes, la depreciación acumulada y la amortización respectivamente, generando el comprobante de operaciones. Se pueden realizar además, gestiones comerciales sobre los activos fijos como son Alquileres, Ventas de activos y servicios asociados a ellos. Este sistema se caracteriza por ser multiplataforma, aplicación web privativa y su mantenimiento se efectúa mediante pago (15). En este sistemas se crean categorías para diferenciar los activos intangibles de los tangibles, es decir, que los trata como un subsistema (16).

OpenERP permite la gestión completa de los recursos empresariales (ERP/CRM) de la información de la empresa, organización o departamento, vía web y/o aplicación de escritorio. Este sistema cuenta con un módulo para la gestión de los activos fijos. Permite crear categorías de activos para diferenciar los activos tangibles, intangibles y diferidos, así como crear los activos con su jerarquía,

su cuadro de depreciación (o amortización) y aplicar los asientos contables (17). Tanto el cliente como el servidor de OpenERP pueden ejecutarse sobre sistemas operativos *GNU/Linux*, *Windows* y *MacOS X*, es decir que es multiplataforma. Es un *software* de código abierto y de libre distribución, por lo que no tiene coste de licencias, ni mantenimiento (18). Puede ponerlo en marcha con su propio personal técnico, o subcontratar el trabajo a una empresa consultora o implantadora, la cual le cobrará únicamente por sus servicios. Ofrece dos tipos de interfaz (aplicación de escritorio y web), menús simplificados o extendidos, acceso basado en roles y un sistema de búsquedas integrado (19).

1.3.2 Sistemas Nacionales

Versat Sarasola es el primer sistema de contabilidad cubano certificado. Está constituido por 10 módulos o subsistemas que incluyen configuración y seguridad, contabilidad general y de gastos, costos y procesos, finanzas y caja. El módulo de los Activos Fijos abarca tanto los tangibles e intangibles. Posibilita a este último realizar el cálculo diario de las amortizaciones, el control de los activos en diferentes monedas, configurar la contabilización, conceptualizar los diferentes movimientos, así como ofrecer variedad de reportes sobre las existencias y movimientos de los activos. A pesar de ser un sistema cubano es privativo y no diferencia los AFI de los Activos Fijos Tangibles (AFT), es decir que son tratados como un mismo subsistema (20).

Rodas XXI es el sistema integral económico administrativo que posibilita automatizar el funcionamiento de cualquier empresa o unidad presupuestada. Contiene diferentes módulos que pueden usarse integrados o independientes como: finanzas, contabilidad, activos fijos, nóminas, inventario, facturación, recursos humanos y tele-cobranzas. En la actualidad estos son difundidos especialmente en el sector de la ciencia. El módulo de Activos Fijos permite tener un control detallado de los activos fijos de la entidad, realizando en el mismo momento que se registra un movimiento, su contabilización. Se pueden realizar todo tipo de operaciones de activos fijos con facilidad en el momento que se desee, generando el documento asociado al movimiento de forma automática previa a la configuración del sistema. Cuando se realizan los cierres si no se ha calculado la amortización, no permite cerrar el período y cuenta con una opción para visualizar información correspondiente a períodos anteriores, tan sólo con cambiar de período contable a períodos anteriores ya cerrados, aunque en dichos períodos no podrá realizar ninguna operación y todos sus módulos cuentan con una ayuda en línea y un manual de usuarios detallado. Entre sus desventajas se encuentra que no es un sistema multiplataforma que solo puede ser ejecutado en el

sistema Windows. Está desarrollado sobre plataformas privativas y no diferencia los AFI de los AFT (21).

1.3.3 Análisis de las soluciones

Durante el análisis de los sistemas se definieron las siguientes pautas como criterio de evaluación:

1. Módulo AFI, permite identificar si los sistemas en cuestión realizan gestión de los activos intangibles.
2. Funcionalidad AFI, es el criterio encargado de determinar si algunos de los sistemas cuenta con al menos una funcionalidad de los activos intangibles.
3. Cálculo de la amortización.
4. Multiplataforma.
5. Software libre.
6. Subsistema, determina si diferencia a los AFT de los AFI.

Tabla 14 Comparación entre los sistemas

Sistemas	1	2	3	4	5	6
Assets	-	X	X	X	-	-
OpenERP	-	X	X	X	X	-
Versat-Sarasola	-	X	X	X	-	-
Rodas XXI	-	X	X	-	-	-

Posteriormente al análisis realizado de los sistemas estudiados, se llegó a la conclusión de que estos no son integrables con el producto Xedro-ERP debido a que son aplicaciones privativas, lo que implica gastos de licencia y mantenimiento, excepto OpenERP. Aunque estas soluciones son utilizadas por algunas entidades en Cuba, se caracterizan por no diferenciar los activos tangibles de los intangibles, es decir, tiene comportamiento unificado. O son aplicaciones desarrolladas sobre plataformas privativas como los sistemas nacionales Versat-Sarasola y Rodas XXI, que además, son soluciones de escritorio. También, agregar que la universidad y el país están inmersos en la migración a software libre, por lo que estos sistemas no cumplen con el principio de independencia tecnológica.

A pesar de no tomar estas soluciones para dar respuesta al problema planteado su estudio permitió un mayor entendimiento del proceso de amortización, así como la configuración de la misma.

1.4 Modelo de desarrollo

El modelo de desarrollo a utilizar es el propuesto por el Centro de Informatización de la Gestión de Entidades (CEIGE) en su versión 1.2. Detalla el ciclo de vida de sus proyectos con la incorporación de los distintos subprocesos dictados por CMMI (Capability Maturity Model Integration, por sus siglas en inglés) para su nivel II, certificación obtenida por el centro en julio de 2011 y reconocida por el Instituto de Ingeniería de Software (SEI, por sus siglas en inglés) como aval de la calidad de su proceso de desarrollo de software (22).

A continuación se muestran las fases del modelo de desarrollo del centro:

Fase de 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 (22). En el presente trabajo no se desarrolla la disciplina de Inicio, debido que en la introducción del trabajo para el diseño metodológico se realizó un estudio del problema.

Fase de 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. Tiene como objetivo obtener un sistema que satisfaga las necesidades de los clientes y usuarios finales. Esta fase está compuesta por las siguientes disciplinas:

- **Disciplina Modelado del negocio**: Esta disciplina está enfocada a comprender los procesos de negocio de la organización, así como identificar su funcionamiento y los procesos que se desean informatizar (22). Los artefactos a generar son el modelo conceptual, glosario de términos, mapa y descripción del proceso de negocio.
- **Disciplina de Requisitos:** Su objetivo principal es desarrollar un modelo con las características de la solución que se va a construir (22). En esta disciplina se identifican y

describen los requisitos funcionales y no funcionales del sistema, así como los artefactos Descripción de requisitos, Prototipos de interfaz de usuario.

- **Disciplina de Análisis y diseño**: Se modela el sistema para que soporte todos los requisitos, lo que contribuye a una arquitectura sólida y estable que se convierte en un plano para la próxima disciplina (22). Entre los artefactos generados en esta etapa están el Diagrama de clases del diseño, Diagrama de secuencia, Modelo de datos, Casos de pruebas, así como el diagrama de despliegue.
- **Disciplina de 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 de software ya implementados se lleva a cabo el desarrollo necesario para ajustar a los requisitos actuales y posteriormente realizar la integración de los componentes (22). En esta disciplina se genera el diagrama de despliegue y el diagrama de componentes.
- **Disciplina de Pruebas internas:** En esta disciplina se realizan pruebas a nivel de proyecto, verificando el resultado de la implementación, permitiendo la identificación de posibles errores en la documentación y el software (22). El tipo de prueba que se realizarán son las pruebas de caja negra utilizando la técnica de partición de equivalencia y de caja blanca utilizando la técnica camino básico.

En el presente trabajo no se tuvieron en cuenta la elaboración de todos los artefactos especificados en el Modelo de Desarrollo del Centro por decisión del proyecto, debido a que la investigación de la solución informática no está respaldada por un funcional capacitado en el tema. Por tanto, se decidió solo confeccionar los artefactos más importantes para el entendimiento de la solución en una posterior versión.

1.5 Ambiente de desarrollo

El Departamento de Tecnologías del CEIGE, en el que se está desarrollando el Sistema de Gestión Integral Xedro-ERP definió las herramientas y tecnologías que se usarán para el desarrollo de la aplicación.

1.5.1 Lenguaje de Modelado Unificado (UML)

Para el desarrollo de la aplicación se utiliza UML 2.2 como el lenguaje con el que se modelaron los artefactos creados en el proceso de desarrollo del software.

UML (Unified Modeling Language): Es un lenguaje que permite visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. Es un conjunto de herramientas, que permite modelar, analizar y diseñar sistemas orientados a objetos (23).

La utilización de UML está dada por las bondades que brinda a los desarrolladores, al proveer un vocabulario y reglas para permitir una comunicación estándar y robusta. Se utiliza dicha versión de UML debido a que es la utilizada por Visual Paradigm 8.0 (24).

UML representa para los desarrolladores de aplicaciones y sistemas una serie de ventajas, al igual que para las organizaciones (24), entre estos beneficios destacan:

- Produce un aumento en la calidad del desarrollo.
- Mejora en un 50% o más los tiempos totales de desarrollo.
- Brinda la posibilidad de obtener un "plano" del sistema.
- Ofrece un mejor soporte a la planificación y control del proyecto.
- Constituye un lenguaje de modelado entendido tanto por humanos como por máquinas.
- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y viceversa.

1.5.2 Visual Paradigm 8.0

Se seleccionó por parte del proyecto el Visual Paradigm para el modelado UML por ser un producto de calidad que soporta aplicaciones web en varios idiomas, siendo fácil de instalar y actualizar, además de tener compatibilidad entre todas sus ediciones. Otra de sus ventajas es la disponibilidad en múltiples sistemas operativos como Windows, Linux, Unix.

Visual Paradigm es una herramienta profesional para el modelado de software que soporta el lenguaje UML para el ciclo de vida completo del desarrollo: análisis y diseño, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción. Además, incluye una herramienta llamada Visual Architect que permite la generación de código para el manejo de la base de datos y puede generar códigos para lenguajes como PHP, Java y gestores de base de datos PostgreSQL (25).

1.5.3 *Apache 2.2*

Apache es un servidor web que utilizando el Protocolo de Transferencia de Hipertexto (por sus siglas en inglés HTTP), es capaz de recibir peticiones de información de un programa cliente (navegador), recuperar la información solicitada y enviarla al programa cliente para su visualización por el usuario (26).

Entre sus principales características se encuentran:

- Multiplataforma.
- Presenta una alta configuración en la creación y gestión de logs. Apache permite la creación de ficheros de log a medida del administrador, de este modo se puede tener un mayor control sobre lo que sucede en el servidor.
- Se desarrolla de forma abierta.
- Extensible: gracias a que es modular se han desarrollado diversas extensiones entre las que destaca PHP.

Es un sistema de gestión de base de datos relacional orientada a objetos. Es software libre y tiene una arquitectura robusta, por lo que ha ganado una sólida reputación para la fiabilidad, integridad y exactitud de los datos. Funciona en todos los principales sistemas operativos. Soporta el almacenamiento de grandes objetos binarios, incluyendo imágenes, sonidos o vídeo. Tiene interfaces de programación en los lenguajes C/C++, Java, Net, Perl, Python, Ruby entre otros. Cuenta con versiones para una amplia gama de sistemas operativos (27).

1.5.4 Netbeans 6.9

El Entorno Integrado de Desarrollo (IDE por sus siglas en inglés) Netbeans es una herramienta para escribir, compilar, depurar y ejecutar programas. Consta de una gran comunidad de usuarios en constante crecimiento, lo que le ha permitido el progreso paulatino de sus prestaciones y la eliminación de errores que pudiesen existir. Ofrece todas las herramientas necesarias para crear aplicaciones web y de escritorio con el lenguaje Java, C/C++ y lenguajes dinámicos como PHP y JavaScript. Es fácil de instalar y se puede ejecutar tanto en Windows como en Linux. Netbeans permite que las aplicaciones se desarrollen a partir de un conjunto de módulos o componentes de software. Brinda una barra de navegación para el acceso rápido a funciones en una clase muy extensa, detecta errores de sintaxis en tiempo real, además de un completamiento de código fuente

eficiente y seguro. En su versión 6.9 destacan las características de soporte para Zend Framework y un nuevo formateador con muchas más reglas (28).

1.5.5 Mozilla Firefox 3.6

Mozilla Firefox es un navegador web con interfaz gráfica de usuario desarrollado por la Corporación Mozilla. El programa es multiplataforma y el código fuente está disponible libremente bajo la triple licencia de Mozilla como un programa libre y de código abierto. Firefox incorpora bloqueo de ventanas emergentes, navegación por pestañas, marcadores dinámicos, compatibilidad con estándares abiertos y un mecanismo para añadir funciones mediante extensiones (29).

Presenta compatibilidad para múltiples extensiones y utiliza el sistema Capa de Conexión Segura (por sus siglas en inglés SSL) para proteger la comunicación con los servidores web, utilizando fuerte criptografía cuando se utiliza el protocolo HTTP. Permite la integración con el antivirus y realiza la navegación por pestañas (30).

1.5.6 Subversión 1.6.6

Subversión es un sistema centralizado de control de versiones que permite llevar el control de los cambios realizados por una o más personas a una serie de documentos a través del tiempo. El mismo es de código abierto y distribuido bajo la licencia Apache. Trabaja replicando el modelo cliente/servidor, uno o más clientes se conectan a un servidor central que tiene la última copia del proyecto como también copias de sus versiones anteriores, dentro de lo que llamaremos un repositorio (31).

1.5.7 Marco de trabajo Sauxe 1.5.4

El desarrollo de aplicaciones informáticas requiere de una serie de pasos para organizar dicho desarrollo y hacerlo estructuradamente. Los desarrollos en entornos web, debido a los procesos que deben manejar, hacen necesario contar con un software capaz de administrar, organizar y manejar ciertos procesos en la etapa de desarrollo. Este software es lo que se conoce como marco de trabajo, que es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado (32).

El marco de trabajo Sauxe fue creado por el departamento de Tecnología del CEIGE de la Universidad de las Ciencias Informáticas. Contiene un conjunto de componentes reutilizables que

provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo (32).

El marco de trabajo define las siguientes herramientas y tecnologías a utilizar para su trabajo:

ExtJS 2.2

Es una librería JavaScript que permite con pocas líneas de código realizar interfaces sencillas y fáciles de usar por los usuarios para construir aplicaciones complejas. Es una de las bibliotecas más avanzadas para el desarrollo rápido de aplicaciones con una apariencia totalmente novedosa y una arquitectura flexible (33).

Sus principales ventajas son: (34)

- Permite crear aplicaciones complejas utilizando componentes predefinidos.
- Evita tener que validar el código para el correcto funcionamiento en cada uno de los navegadores (Firefox, Internet Explorer, Safari, Opera).
- El funcionamiento de las ventanas flotantes lo pone por encima de cualquier otro.
- Relación balanceada entre Cliente-Servidor: Se distribuye la carga de procesamiento, permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- Eficiencia de la red: Disminuye el tráfico en la red ya que las aplicaciones cuentan con la posibilidad de elegir qué datos desean transmitir al servidor y viceversa.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información logrando una abstracción del cliente en el proceso.

Su utilización como eje principal en el desarrollo, se deriva de las numerosas ventajas y funcionalidades que aporta la librería, sobre todo del trabajo con peticiones asincrónicas al servidor. Sus características lo convierten en un potente marco de trabajo, las cuales mejoran considerablemente la experiencia del usuario final de la aplicación, sin dejar de mencionar las herramientas que brinda para la creación de ventanas, formularios y validadores.

Zend Framework 1.9.7

Se trata de un marco de trabajo para el desarrollo de aplicaciones web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. Además es de código abierto. Está formado por una serie de métodos estáticos y componentes que usarán estos métodos. Entre los componentes tienen vital importancia: Zend_Config para temas de configuración de aplicaciones web y Zend_Db para tratar con bases de datos. Trabaja con el patrón Modelo Vista Controlador (MVC, por sus siglas en inglés) e incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultarlas, sin tener que escribir ninguna consulta SQL y solución para el acceso a base de datos que balancea el Mapeo de Objetos Relacionales (ORM, por sus siglas en inglés) con eficiencia (32).

Doctrine Framework 1.2.2

Es un potente y completo sistema para el ORM en PHP, con una capa de abstracción a base de datos incorporado. Brinda la posibilidad de exportar una base de datos existente, 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. Tiene como ventajas que facilitan tareas comunes y de mantenimiento como son: la reutilización, encapsulación, portabilidad, seguridad y mantenimiento del código (32).

JavaScript 1.6

JavaScript es un lenguaje de programación que es interpretado similar en su sintaxis al lenguaje C y también incorpora algunas convenciones del lenguaje Java, orientado a objetos y basado en prototipos. Es una tecnología del lado del cliente por lo que permite interactuar con casi todos los navegadores de manera dinámica y eficaz. Es importante conocer que el lenguaje es muy sensible al uso indebido de mayúsculas y permite dinamismo en las páginas HTML (33).

PHP 5.2

Es un lenguaje de código abierto interpretado de alto nivel, especialmente pensado para desarrollo web y el cual puede ser incrustado en páginas HTML. Permite la conexión a diferentes tipos de servidores de bases de datos entre ellos PostgreSQL. Aplica técnicas de programación orientada a objetos. La meta de este lenguaje es permitir a los creadores de páginas web escribir páginas dinámicas de una manera rápida y fácil. No obliga a quien lo usa a seguir una determinada metodología a la hora de programar, aún estando dirigido a alguna en particular. El programador

puede aplicar en su trabajo cualquier técnica de programación y/o desarrollo que le permita escribir código ordenado, estructurado y manejable (35).

Servidor de BD: PostgreSQL 8.3

Es un sistema de base de datos relacional perteneciente al ámbito del software libre que destaca por su robustez, escalabilidad y cumplimiento de los estándares SQL. Este servidor tiene optimizador de consultas, crea y borra índices en la base de datos sin necesidad de reescribir el código y es el mismo gestor el encargado del mantenimiento de los mismos, alta concurrencia, que evita tener que bloquear una tabla cuando se está escribiendo en ella y -copias de seguridad en línea. El mismo es multiplataforma y extensible, además fue diseñado para ambientes de alto volumen de información con satisfactorios resultados y está disponible totalmente sin costos alguno (36).

1.6 Conclusiones parciales

- Con el estudio de diferentes leyes y procesos de los AFI, se definieron los conceptos principales ERP, Logística, AFI, Nomenclador, Configuración y Amortización referentes al dominio del problema.
- Los sistemas analizados no se adaptan a las necesidades de independencia tecnológica de la universidad y a la diferenciación de los procesos de los activos tangibles e intangibles.
- En este trabajo se tendrá presente para el ciclo de vida de la solución informática el Modelo de Desarrollo del Centro CEIGE, aplicando la disciplina de la fase de desarrollo: negocio, requisitos, análisis y diseño, implementación y pruebas internas, excluyendo la fase de inicio.
- La solución informática será desarrollada mediante el marco de trabajo Sauxe en su versión 1.5.4, como base tecnológica del producto Xedro-ERP, que cuenta con varias tecnologías de carácter obligatorio: ExtJS 2.2, Zend Framework 1.9.7, Doctrine Framework 1.2.2, PostgreSQL 8.3. Para el modelado de diagramas se utilizará Visual Paradigm 8.0, como servidor web Apache 2.2, como entorno de desarrollo Netbeans 6.9, como lenguaje de programación PHP 5.2, JavaScript 1.6 y como control de versiones *Subversión 1.6.6*.

Capítulo 2. Disciplina de negocio, requisitos, análisis y diseño

En este capítulo se representa la disciplina Modelado del negocio y los artefactos que en esta se generan. Se realiza una descripción detallada de los principales procesos relacionados a los activos fijos intangibles y se definen los requisitos funcionales y no funcionales precisos para el desarrollo de la aplicación. Después de un análisis de los requerimientos del sistema se procede al diseño de los componentes en cuestión, para así tener una visión más clara de la implementación del sistema, teniendo en cuenta la utilización de patrones de diseño.

2.1 Disciplina de modelado del negocio

El modelado del negocio logra crear una visión más profunda de la organización donde se realiza la automatización, permitiendo definir los procesos, roles y responsabilidades en los modelos de casos de uso del negocio. Permite comprender la estructura y la dinámica de la organización en la cual se va a implantar el sistema, además de los problemas actuales de dicha organización e identificar así las mejoras potenciales. Realizar un modelado del negocio tiene como ventaja que se puedan derivar los requerimientos del sistema que va a soportar la organización (37).

2.1.1 Modelo conceptual

Un modelo conceptual es una representación de conceptos significativos en el dominio del problema. Se realiza cuando no se logra determinar el proceso del negocio con fronteras bien establecidas y donde los flujos de información son difusos (múltiples orígenes, sólo eventos, sucesos), cuando existe solapamiento de responsabilidades, así como múltiples responsabilidades. Además es difícil establecer reglas de funcionamiento (38).

A continuación se muestra el modelo conceptual del presente trabajo.

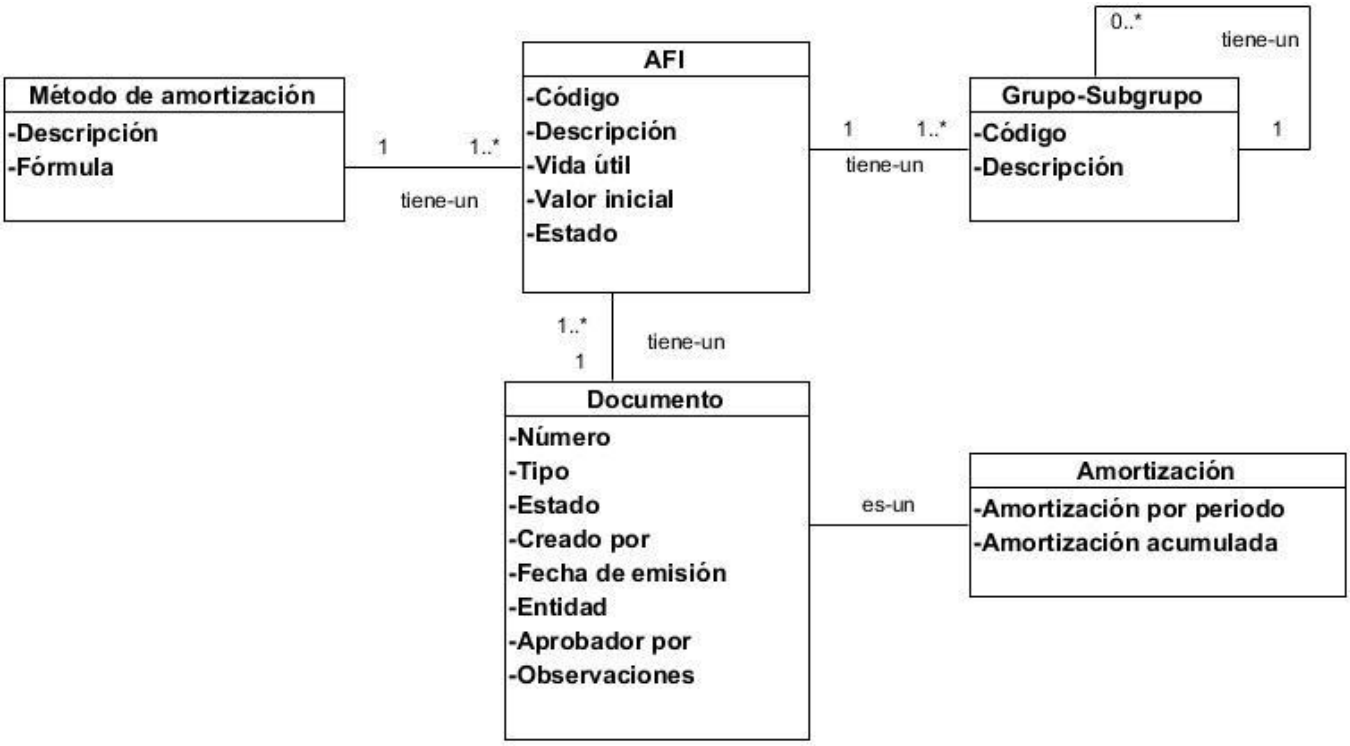


Figura 20. Modelo conceptual

A continuación se muestra la descripción de cada uno de los conceptos que conforman el modelo conceptual.

Tabla 15. Descripción de conceptos del modelo conceptual

Concepto	Descripción
AFI	Son aquellos que representan el nomenclador AFI y se caracterizan por ser activos o elementos que no se pueden palpar, identificables, de carácter no monetario y sin apariencia física, se construyen y acumulan a lo largo del tiempo.
Amortización	Es la merma en el valor contable de un bien que se utiliza para distribuir su costo de origen durante la vida útil del mismo.
Grupo-Subgrupo	Se crea con el objetivo de agrupar los activos fijos intangibles, unidos por lo general estructuralmente o porque desempeñan una función común.

Documento	Es un escrito digital que permite guardar las operaciones que se realizan.
Método de Amortización	Es una forma que me permita calcular cuánto amortiza un activo intangible, respecto al tiempo. Existen diversos métodos a utilizar pero en el presente trabajo se utilizará el de línea recta, donde: tasa de amortización anual = costo / número de años a amortizar. Los activos se amortizarán según el término de su vigencia, para los casos en que no está determinado el plazo será de 20 años y para el caso del software de 2 años.

2.1.2 Mapa de proceso de negocio

Un proceso de negocio es un grupo de tareas que en conjunto crean un resultado de valor para el consumidor. Permite realizar un análisis del problema existente a fin de comprender la estructura y la dinámica de la entidad en la cual se va a implantar el sistema, entender los problemas actuales de la organización e identificar las mejoras potenciales. Asegura que los consumidores, usuarios finales y desarrolladores tengan un entendimiento común de la organización y deriva los requerimientos del sistema que va a soportar la organización. El proceso identificado es: **Amortización**.

El mapa de procesos es el artefacto mediante el cual se puede explicar de manera global el funcionamiento del proceso existente. A continuación se muestra el mapa de proceso correspondiente al proceso Amortización y su relación con otros procesos del módulo de AFI.

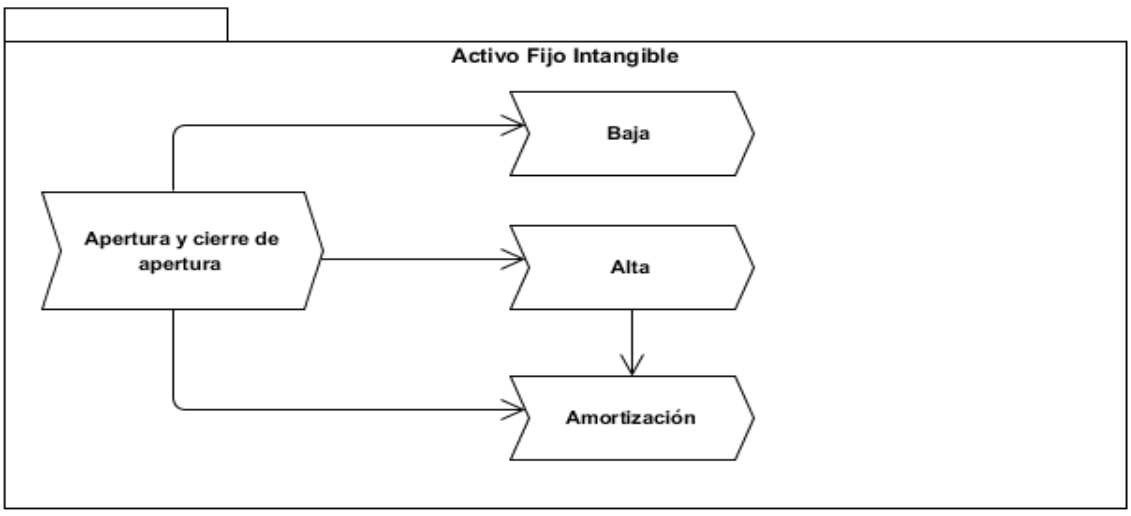


Figura 21. Mapa de procesos

2.1.3 Descripción del proceso Amortización

La amortización es la pérdida de valor de un producto a lo largo del tiempo. En la descripción de este proceso intervienen dos actores fundamentales el económico y el contador. A continuación se muestra la descripción del proceso puesto en cuestión:

- Se calcula la amortización para el período actual. (Económico)
- Calcular la amortización acumulada a partir de la sumatoria de los gastos de la amortización mensual de cada activo. Contabiliza cada activo generando un comprobante de operaciones con las amortizaciones mensuales. (Económico)
- Envía el comprobante de operaciones a Contabilidad General. (Económico)
- Se actualiza el Submayor Específico del activo (Económico) y al mismo tiempo se asienta el comprobante de operaciones. (Contador)

A continuación se muestra el diagrama del proceso de negocio correspondiente a lo explicado anteriormente (ver **figura 3**).

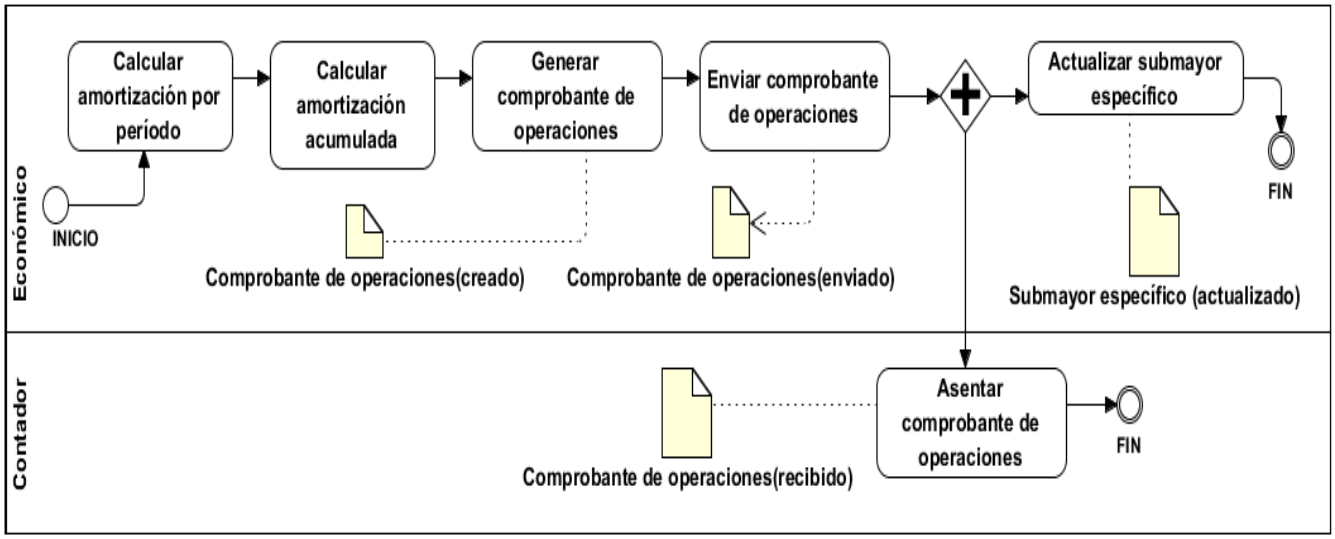


Figura 22. Descripción del proceso de negocio de amortización

2.2 Disciplina de requisitos

Los requisitos cumplen un papel primordial en el proceso de desarrollo de software. Los requisitos de software se clasifican en funcionales y no funcionales. Los requisitos funcionales escriben qué es lo que el sistema debe hacer para dar soporte a las funciones y objetivos del usuario. Los requisitos

no funcionales imponen restricciones de cómo los requisitos funcionales deben ser implementados (39).

2.2.1 Requisitos funcionales

Los requisitos funcionales establecen el comportamiento del sistema y son condiciones o capacidades que el sistema debe cumplir (39).

2.2.2 Técnicas utilizadas para la captura de requisitos

Tormenta de ideas: Se utilizó la técnica tormenta de ideas donde participaron un grupo de estudiantes y trabajadores de la línea logística, con el objetivo de que los participantes aportaran ideas que contribuyan al desarrollo de la solución planteada. Estas ideas fueron recogidas por escrito para su posterior análisis.

Soluciones existentes: esta técnica fue utilizada durante el estudio de sistemas que tengan relación con alguna de las funcionalidades de los AFI, específicamente el tratamiento de los componentes nomenclador, configuración y amortización.

Luego de la captura de requisitos, se identificaron 34 requisitos funcionales en 6 agrupaciones. A continuación se muestra un listado de los requisitos y su prioridad.

Tabla 16. Requisitos funcionales y prioridad

Nro	Requisitos Funcionales(RF)	Prioridad
	RF1 Gestionar nomenclador grupo-subgrupo	
1	RF1.1 Adicionar Grupo-Subgrupo	Crítico
2	RF1.2 Modificar Grupo-Subgrupo	Crítico
3	RF. 1.3 Listar Grupo-Subgrupo	No Crítico
4	RF1.3 Eliminar Grupo-Subgrupo	Crítico
	RF2 Gestionar nomenclador AFI	
5	RF2.1 Adicionar nomenclador AFI	Crítico
6	RF2.2 Modificar nomenclador AFI	Crítico
7	RF2.3 Eliminar nomenclador AFI	Crítico
8	RF2.4 Listar nomencladores AFI	No Crítico
9	RF2.5 Buscar nomenclador AFI	No Crítico
	RF3 Gestionar nomenclador estado AFI	
10	RF3.1 Adicionar estado AFI	Crítico

11	RF3.2 Modificar estado AFI	Crítico
12	RF3.3 Eliminar estado AFI	Crítico
13	RF3.4 Listar estado AFI	No Crítico
14	RF3.5 Buscar estado AFI	No Crítico
15	RF4 Configurar amortización	No Crítico
	RF5 Gestionar documento de amortización	
16	RF5.1 Adicionar documento de amortización	Crítico
17	RF5.2 Modificar documento de amortización	Crítico
18	RF5.3 Eliminar documento de amortización	Crítico
19	RF5.4 Listar documento de amortización	No Crítico
20	RF5.5 Consultar documento de amortización	No Crítico
21	RF5.6 Buscar documento de amortización	No Crítico
22	RF5.7 Realizar búsqueda avanzada de documento de amortización	No Crítico
23	RF5.8 Imprimir modelo de amortización	No Crítico
24	RF5.9 Imprimir listado	No Crítico
	RF6 Gestionar AFI de amortización	
25	RF6.1 Adicionar AFI de amortización	Crítico
26	RF6.2 Calcular amortización del periodo	Crítico
27	RF6.3 Calcular amortización acumulada	Crítico
28	RF6.4 Listar AFI de documento de amortización	No Crítico
29	RF6.5 Buscar AFI de documento de amortización	No Critico
30	RF7 Confirmar documento de amortización	Crítico
31	RF8 Cancelar estado del documento de amortización	No Crítico
32	RF9 Contabilizar documento de amortización	No Crítico
33	RF10 Cancelar documento de amortización	No Crítico
34	RF11 Listar AFI amortizables de la entidad	Crítico

Crítico: Si al quitar el requisito puesto en cuestión afecta la lógica del negocio.

No Crítico: Si al quitar el requisito puesto en cuestión no afecta la lógica del negocio.

2.2.3 Descripción de requisito

Como ejemplo ilustrativo se presenta la descripción del requisito funcional Adicionar documento de amortización de la agrupación Gestionar documento de amortización. Las descripciones de los requisitos funcionales restantes se pueden encontrar en el expediente del proyecto.

Tabla 17. Descripción de requisito Adicionar documento de amortización

Precondiciones	El usuario se ha autenticado ante el sistema. No existe un documento de amortización para el periodo.	
Flujo de eventos		
Flujo básico: Adicionar documento de amortización		
1. El sistema debe cargar los siguientes datos del documento: entidad, número, fecha de emisión, estado. 2. Se introducen los datos del documento de amortización: observaciones. 3. El sistema valida los datos introducidos. 4. Si los datos son correctos el sistema los registra. 5. El sistema confirma el registro de los datos. El sistema guardará la fecha actual. 6. El sistema valida que los datos introducidos sean correctos. 7. Concluye el requisito.		
Pos-condiciones	Se registró en el sistema un nuevo documento de amortización para el período. El documento toma el estado de “Elaboración”.	
Flujos alternativos		
Flujo alternativo 3.a Información incompleta		
1. El sistema señala los datos vacíos y permite corregirlos. 2. El usuario corrige los datos. **3.** Volver al paso 2 del flujo básico.		
Pos-condiciones	N/A	
Flujo alternativo 3.b Información errónea		
1. El sistema señala los datos erróneos y permite corregirlos. 2. El usuario corrige los datos. **3.** Volver al paso 2 del flujo básico.		
Pos-condiciones	N/A	
Flujo alternativo *. El usuario cancela la acción		
1. El sistema no registra los datos del documento. 2. Concluye el requisito.		
Pos-condiciones	No se registran los datos.	
Validaciones	Se valida que el atributo observación va admitir 255 caracteres.	
Conceptos	**Documento**	Visibles en la interfaz: entidad, número, fecha de emisión, estado, observaciones. Utilizados internamente: entidad, número, fecha de emisión, estado, observaciones.

Requisitos especiales	Son los requisitos no funcionales específicos para el requisito.
Asuntos pendientes	Posibles mejoras al requisito.

2.2.4 Validación de requisitos funcionales

El proceso de validación de requisitos comprende actividades que generalmente se realizan una vez que se ha obtenido una primera versión de la documentación de requisitos. Según Somerville la validación de requisitos "trata de mostrar que éstos realmente definen el sistema que el cliente desea. Su importancia radica en que los errores en el documento de requisitos pueden conducir a importantes costes al repetir el trabajo cuando son descubiertos durante el desarrollo o después que el sistema esté en uso" (39).

Para la validación de requisitos se aplicaron las siguientes técnicas:

Revisión técnica formal a los documentos y artefactos: Esta técnica se aplicó mediante la revisión de documentos y artefactos generados por parte de los especialistas y el equipo de desarrollo de la línea de logística. Con la misma se pudo comprobar que la información brindada por los especialistas permitió realizar una correcta descripción de los requisitos. Mediante la aplicación de esta técnica se obtuvo como resultado un documento legal aprobado y firmado por la jefa de la línea y los tutores (ver **Anexo 2**).

Prototipo de interfaz de usuario: La aplicación de esta técnica permite ver cómo quedarán las funcionalidades en una futura implementación, en ésta se obtiene como salida un prototipo de interfaz de usuario por cada uno de los requisitos funcionales identificados. A continuación se muestra un ejemplo de la salida de la aplicación de esta técnica como es el prototipo de gestionar documento de amortización, específicamente adicionar documento de amortización.

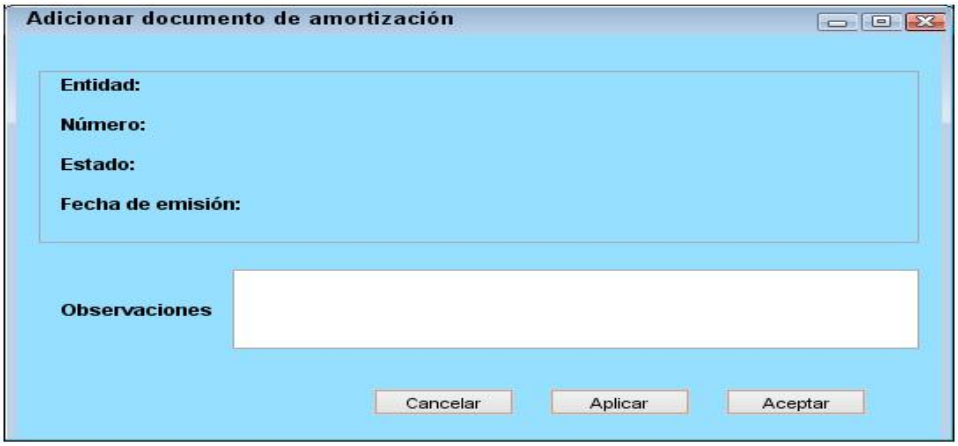


Figura 23. Adicionar documento de amortización

2.2.5 Requisitos no funcionales

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos no funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer (39).

Los requisitos no funcionales por los que se guiará este trabajo, son los ya definidos en la línea base de logística. A continuación se muestra un listado con los requisitos no funcionales en la **tabla 5**.

Tabla 18. Requisitos no funcionales

RNF	Descripción
Portabilidad (POR)	El sistema debe ser multiplataforma.
Restricciones de diseño e implementación (RED)	Se empleará como lenguaje del lado del servidor PHP 5.2 y del lado del cliente JavaScript 1.6. Además se utilizará como servidor de bases de datos PostgreSQL 8.3.
Seguridad (SEG)	El usuario debe autenticarse antes de entrar al sistema. La información que se maneje en el sistema estará protegida de acceso no autorizado y divulgación, a partir de los diferentes roles de los usuarios que empleen el sistema. Se especificarán niveles de acceso al sistema en dependencia del rol que desempeñe el usuario. La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Se les garantizará el acceso a la información solo a los usuarios autorizados

	evitando que los dispositivos o mecanismos utilizados para lograr la seguridad oculten o retrasen a los usuarios en la obtención de los datos deseados en un momento dado.
Software (SFT)	Para el servidor: Sistema operativo Linux en cualquiera de sus distribuciones, PHP v.5.2 (estable), Doctrine v.1.2.1 (estable), Zend Framework v.1.9.7 (estable) ZendExt v.2.5 (, estable). Para el cliente: Sistema operativo Windows XP o GNU/ Linux. Navegador Mozilla Firefox v.3.6 o superior (estable).
Soporte (SOP)	Se necesita un servidor de bases de datos que soporte grandes volúmenes de datos. Debe elaborarse un paquete de instalación que abarque verificación de componentes ya instalados y la instalación de los nuevos.
Usabilidad (USB)	El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras.
Hardware (HDW)	Por el lado cliente: Procesador 1.40 GHZ, RAM 256 MB (recomendado 512 MB), Tarjeta de Red. Del lado del servidor de aplicaciones: Tarjeta de Red, Procesador 3.00 GHZ, RAM 1GB, Disco duro 160 GB, 1 UPS, Lector de CD. Del lado del servidor de Base de datos: Tarjeta de Red, Procesador 3.00 GHZ, RA1GB, Disco duro 160 GB, 1 UPS, Lector de CD.
Interfaz (INU)	El sistema debe contar con una interfaz fácil, sencilla, permitiendo que los usuarios finales del mismo sean capaces de interactuar con este aun teniendo conocimientos básicos. Empleo de imágenes y colores identificados con el negocio donde se implantará el sistema.

2.3 Disciplina de análisis y diseño

El diseño del software se encuentra en el núcleo técnico de la ingeniería de software. Una vez que se analizan y especifican los requisitos del producto se realizan las pruebas que se requieren para construir y verificar el software. El diseño tiene como objetivo determinar una solución a los requisitos del sistema definidos anteriormente. En este flujo de trabajo se generan artefactos como el modelo de diseño a través de diagramas de clases del diseño con estereotipos web y se detallan los patrones empleados en el diseño de la aplicación.

2.3.1 Arquitectura del sistema

La arquitectura de software es considerada como el diseño de más alto nivel de la estructura de un sistema, lo cual abarca componentes de software, las propiedades visibles externamente de esos componentes, y las relaciones entre ellas.

La definición oficial ofrecida por la IEEE[1] plantea que: “La Arquitectura del Software es la organización fundamental de un sistema formado por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución”.

2.3.2 Estilos arquitectónicos

Los estilos expresan la arquitectura en el sentido más formal y teórico, describen entonces una clase de arquitectura, o piezas identificables de las arquitecturas empíricamente dadas. Esas piezas se encuentran repetidamente en la práctica, trasuntando la existencia de decisiones estructurales coherentes. Una vez que se han identificado los estilos, es lógico y natural pensar en reutilizarlos en situaciones semejantes que se presenten en el futuro (40).

Para el desarrollo del marco de trabajo Sauxe, en el Departamento de Tecnologías del CEIGE se definió el estilo N capas, con cinco capas fundamentales que se describen a continuación:

Capa de presentación: Esta es la capa encargada de elaborar y mostrar las interfaces a los usuarios. La capa de presentación está compuesta principalmente por el marco de trabajo ExtJS y centra su desarrollo en tres componentes fundamentales archivos JS, archivos CSS, páginas clientes y páginas servidoras.

Capa de control o negocio: En esta capa se encuentra ubicado el marco de trabajo Zend y se implementa el patrón Modelo-Vista-Controlador.

Capa de acceso a datos: Está ubicado el marco de trabajo Doctrine, como ORM para la comunicación con el servidor de datos mediante el protocolo PDO[2].

Capa datos: En esta capa se encuentra ubicado el servidor de base de datos y los archivos XML donde se almacena la información de la aplicación.

Capa de servicio: En esta última capa se encuentran los subsistemas que prestan y consumen servicios entre sí.

En la capa de control o negocio de la arquitectura descrita anteriormente para Sauxe, se emplea el patrón arquitectónico MVC[3], el cual se encarga de separar los datos de la aplicación, la interfaz de usuario y la lógica de control en tres componentes diferentes:

---

[1] **IEEE**: por las siglas en inglés ***I**nstitute of **E**lectrical and **E**lectronics **E**ngineers*. Traducción: Insituto de Ingenieros Eléctricos y Electrónicos

[2] **PDO:** Protocolo de conexión a base de datos basado en objetos

Modelo: Es la representación específica de la información con la cual el sistema trabaja. Es responsable de la lógica de negocio de una aplicación. Se encapsula el acceso a los datos y proporciona una biblioteca de clases reutilizables (40).

Vista: Éste presenta al modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Es quien controla la apariencia de los datos y proporcionan funcionalidades para recoger los de los usuarios (40).

Controlador: Responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista. Es el encargado de unir la capa de las vistas con la del modelo, así como de tratar la información recibida por la vista y manejar el flujo de ejecución de la aplicación, es la encargada de llamar a las funcionalidades del modelo y retornar estos datos a una vista (40).

2.3.3 Patrones de diseño

Los patrones de diseño tienen como objetivo proporcionar catálogos de elementos reusables en el diseño de sistemas de software. Evita la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente. Formaliza un vocabulario común entre diseñadores, estandariza el modo en que se realiza el diseño y facilita el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente (41).

En la implementación del sistema se utilizan patrones de diseño que pertenecen a las dos vertientes de estos: Patrones de los principios generales para asignar responsabilidades (GRASP, por sus siglas en inglés) y GOF[4]. A continuación se describen brevemente estas dos clasificaciones y los patrones utilizados para el sistema.

GRASP:

Los GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresado en forma de patrones (41). A continuación se describen los patrones GRASP que serán utilizados en el diseño de la solución, con el fin de contribuir al desarrollo de un sistema más robusto y flexible:

Experto: Sugiere asignar la responsabilidad al objeto que posea la información necesaria para desempeñarla. Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos

[3] **MVC**: patrón arquitectónico ***M**odelo **V**ista **C**ontrolador*

[4] **GoF:** acrónimo de ***G**ang **o**f **F**our.* Traducción: Banda de los Cuatro.

se valen de su propia información para hacer lo que se les pide. 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. Este patrón se evidencia en la clase DatDocAfi la cual es la encargada de realizar las acciones directamente a la base de datos.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. Su fundamental propósito es elegir los objetos que agreguen, contengan, registren, utilicen más estrechamente o posean los datos de inicialización de las instancias. Este patrón se evidencia en la clase GesAmortizacionController, la cual contiene las acciones y es la encargada de ejecutarlas. Dentro de esta existen varias funcionalidades en las cuales se crean varios objetos o instancias de las clases que representan cada una de las tablas existentes en la base de datos.

Controlador: Este patrón plantea asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Donde un evento del sistema es un evento de alto nivel generado por un actor externo; es decir, un evento de entrada externa. La clase controladora definida, la cual se denomina GesAmortizacionController es un ejemplo de la aplicación de este patrón, y se encargará de manejar los eventos dentro del componente en cuestión.

Alta Cohesión: Este patrón Responde al "**cómo**" mantener la complejidad dentro de límites manejables. Mejora la claridad y la facilidad con que se entiende el diseño, además simplifica el mantenimiento y las mejoras en funcionalidad. Fue utilizado en el diseño de los componentes de manera general; donde se agruparon las clases en dependencia de los requerimientos a los que se les debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

Bajo Acoplamiento: Este patrón responde al **"cómo"** dar soporte a una dependencia escasa y a un aumento de la reutilización. En el modelo de datos se definieron un conjunto de clases persistentes, entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables para reducir el impacto de los cambios y acrecentar la oportunidad de una mayor productividad.

GOF:

A continuación se presenta una descripción de los patrones GOF empleados por el marco de trabajo Sauxe, el cual usa varios marcos de trabajo de desarrollo y estos a su vez implementan independientemente sus patrones de diseño.

Cadena de Responsabilidad: Es un patrón de comportamiento que permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada (42). Este patrón es utilizado en el tratamiento de excepciones. Un ejemplo del mismo es cuando se produce en error al adicionar un documento de amortización en la base de datos este es captado por las capas superiores (en este caso se evidencia en las clases controladoras y las del modelo) y estas reenvían la excepción hasta la capa de aplicación donde traduce al lenguaje del usuario y se le informa.

Facade (del español Fachada): en la vista se implementa el patrón estructural Fachada el cual proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de utilizar. En el desarrollo de esta solución la utilización del patrón estructural Fachada se ve en que cada componente tiene un paquete denominado servicios donde se encuentran las clases php que contiene los servicios que brindan estos componentes a otros.

Singleton (del español Instancia Única): garantiza que exista una instancia única para una clase y proporciona un punto de acceso global a ella (42). Este patrón se evidencia en las conexiones a la base de datos, donde, si ya el objeto de conexión está creado no es necesario volver a crear uno nuevo, sino que se establece la conexión desde un mismo punto de acceso.

Decorador (del español envoltorio): Es otro patrón estructural que añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad. Se implementa en la vista, en la clase Zend_View, encargada de asignarle responsabilidades a objetos de manera dinámica y configurarlos con nuevos atributos.

2.3.4 Diagrama de clases de diseño

Un diagrama de clases es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. 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 (43).

A continuación se muestra el diagrama de clases del diseño del requisito gestionar documento de amortización del presente trabajo:

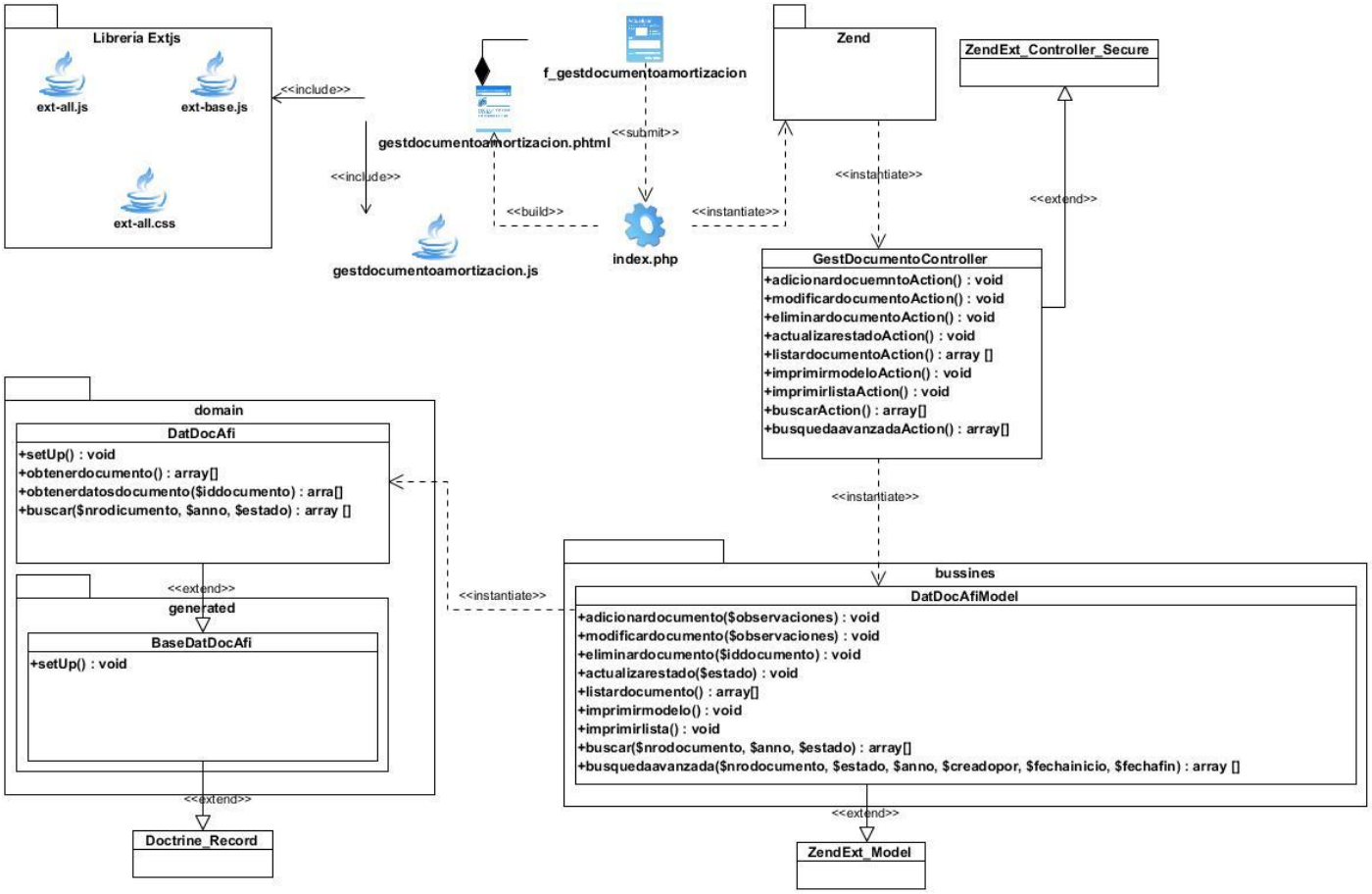


Figura 24. Diagrama de clase del diseño Gestionar documento de amortización

A continuación se muestra la descripción de la clase DatDocAfiModel, así como sus respectivos atributos.

Tabla 19. Descripción de la clase DatDocAfiModel

Nombre de la clase: DatDocAfiModel	
Tipo de clase: Modelo	
Nombre	function DatDocAfiModel ().
Descripción	Constructor de la clase DadDocAfiModel.
Nombre	function CargarDocumentos ().
Descripción	Carga los documentos del módulo AFI.
Nombre	function addActivoDocumento($arreglo, $iddocumento)
Descripción	Adicionar los activos que amortizan al documento
Nombre	function adicionarDocumento($doc)
Descripción	Adiciona un nuevo documento
Nombre	function modificarDocumento($doc)
Descripción	Modifica el documento seleccionado
Nombre	function eliminarDocumento($iddocumento, $operacion, $version)
Descripción	Elimina el documento

2.3.5 Diagrama de secuencia

Un diagrama de secuencia es una forma de diagrama de interacción que muestra los objetos como líneas de vida a lo largo de la página y con sus interacciones en el tiempo representadas como mensajes dibujados como flechas desde la línea de vida origen hasta la línea de vida destino. Los diagramas de secuencia son buenos para mostrar qué objetos se comunican con qué otros objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos (44).

A continuación se muestra el diagrama de secuencia correspondiente al requisito de adicionar documento de amortización.

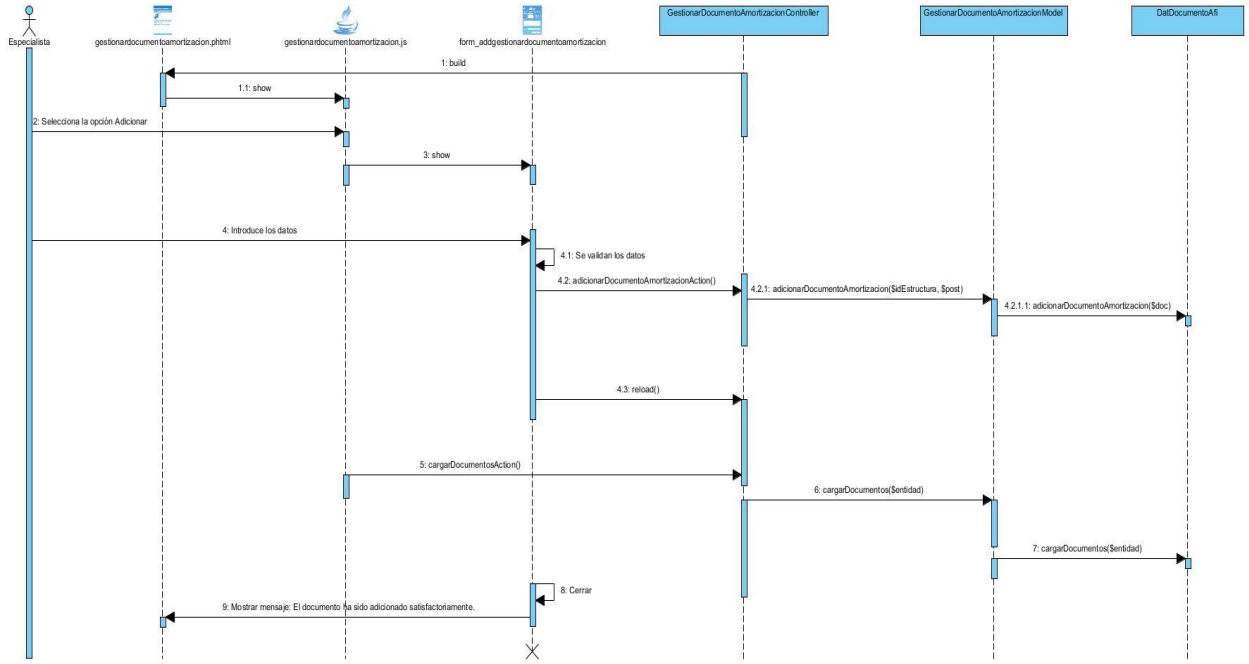


Figura 25. Diagrama de secuencia del requisito Adicionar documento de amortización

2.3.6 Modelo de Datos

Un modelo de datos es la combinación de una colección de estructuras de datos, operadores o reglas de inferencia y de reglas de integridad, las cuales definen un conjunto de estados consistentes. El cual puede ser usado como una herramienta para especificar los tipos de datos y la organización de los mismos. Además para la manipulación de consultas y datos, así mismo es el elemento clave en el diseño de la arquitectura de un manejador de Base de Datos (45). En la **figura 7** se muestra el modelo de datos correspondientes al presente trabajo.

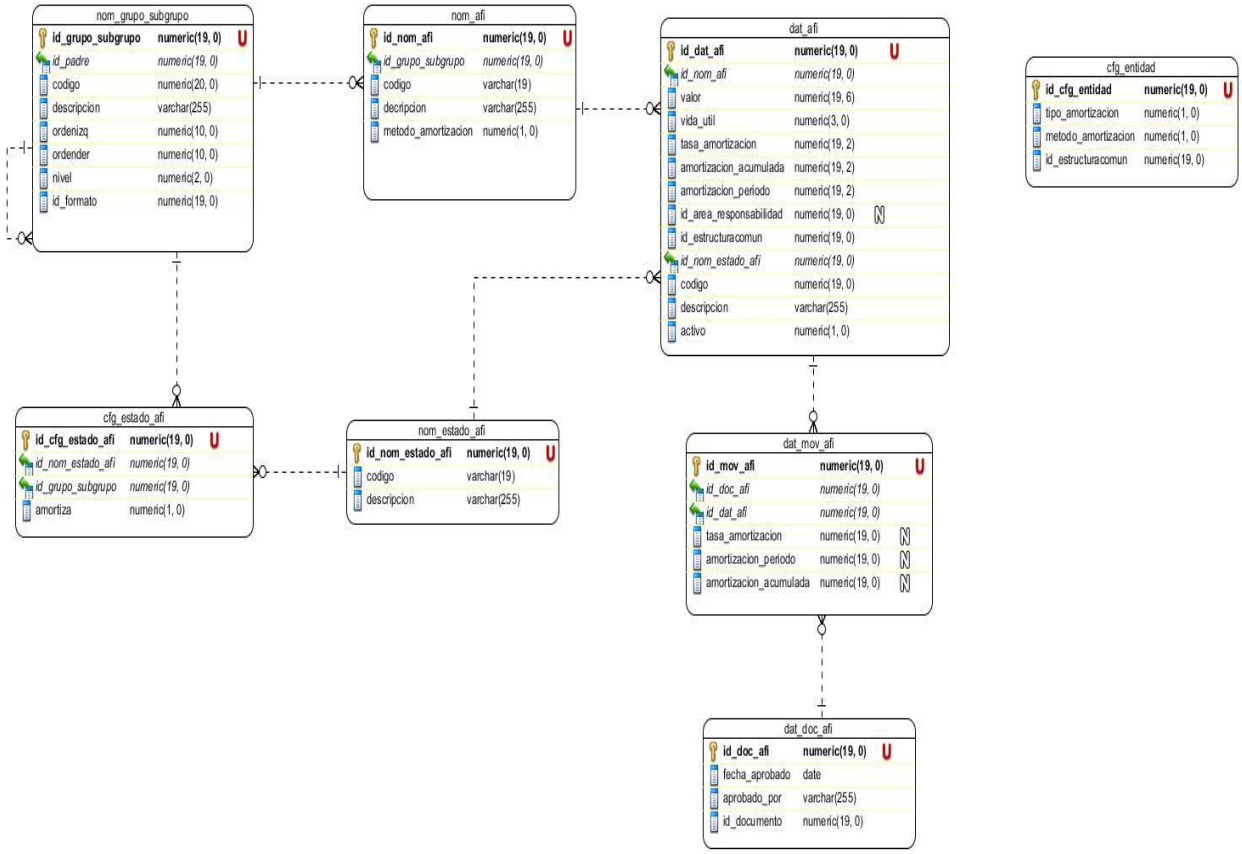


Figura 26. Modelo de datos

2.3.7 Validación del diseño

Las métricas basadas en clases propuestas por Lorenz y Kidd dividen las métricas basadas en clases en cuatro categorías: tamaño, herencia, valores internos y valores externos. Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones, para luego promediar los valores para el sistema en su totalidad. Las métricas basadas en herencia se centran en la forma en que se reutilizan las operaciones en la jerarquía de clases. Las métricas para valores internos de clase examinan la cohesión y asuntos relacionados con el código así como las métricas orientadas a valores externos examinan el acoplamiento y la reutilización (46).

Para la validación del diseño se utilizan un conjunto de métricas de software que permiten medir de forma cuantitativa la calidad de los atributos internos del software. Se seleccionaron las métricas Tamaño Operacional de Clase (TOC) y Relaciones entre Clases (RC) para validar el diseño de los componentes nomenclador, configuración y amortización.

Tamaño Operacional de Clase (TOC)

Esta métrica se determina por el número total de operaciones que están encapsuladas dentro de la clase. Cada una de las clases puede tener demasiada responsabilidad, lo cual reducirá la reusabilidad de la misma y complicará su implementación. Además cuanto menor sea el valor medio para el tamaño más probable es que las clases tengan menos responsabilidad y complejidad y más nivel de reutilización.

Los criterios y categorías definidos para la evaluación de los atributos de calidad que anteriores se presentan se visualizan en la siguiente tabla:

Tabla 20. Criterios de evaluación para la métrica TOC

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio.
Reutilización	Baja	>2*Promedio.
	Media	Entre Promedio y 2*Promedio.
	Alta	<=Promedio.

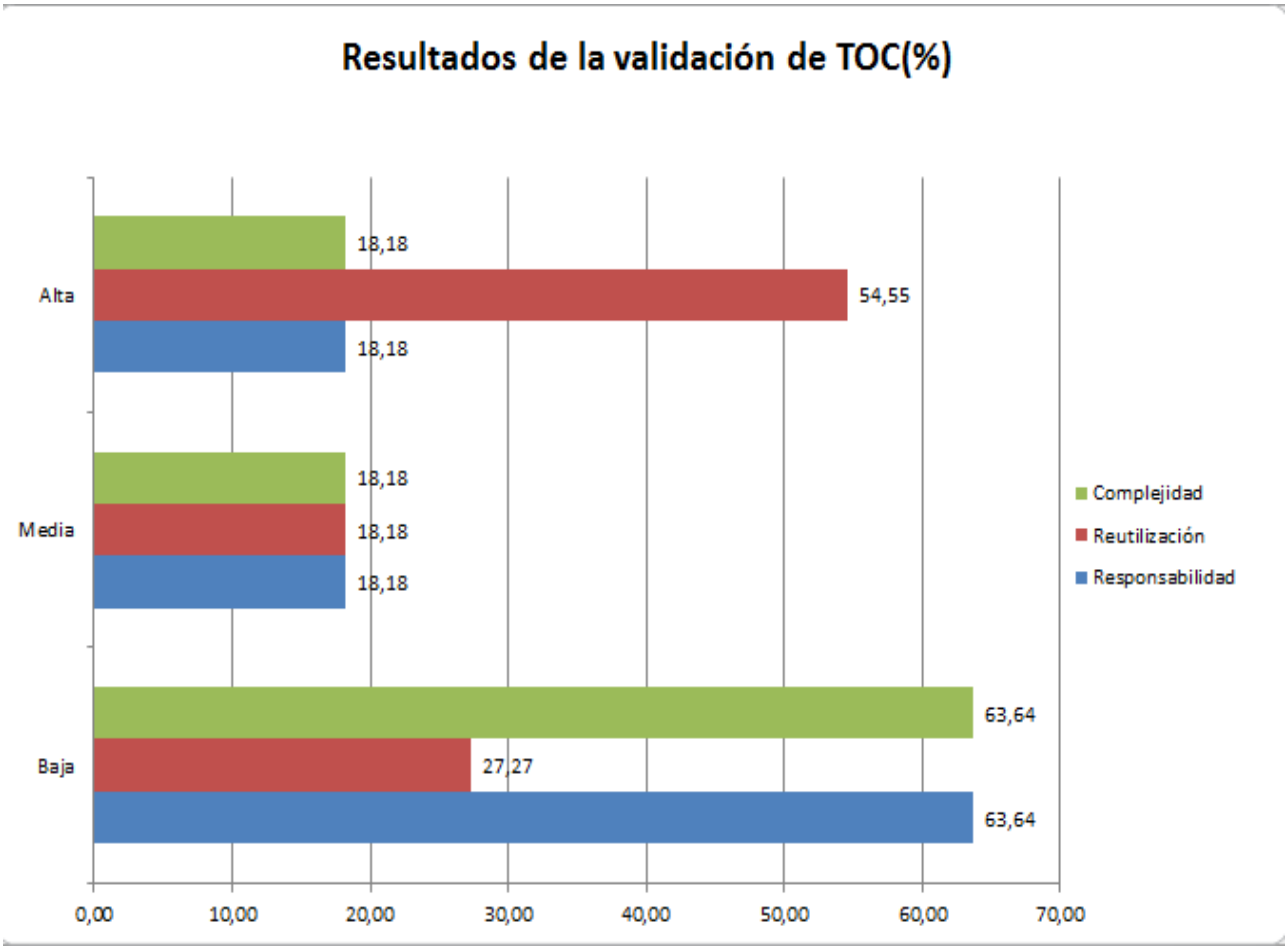


Figura 27. Resultados de la métrica TOC

Haciendo un análisis de los resultados obtenidos para los atributos de la métrica TOC en la evaluación del instrumento, se puede observar que la mayoría de las clases que conforman las funcionalidades de los componentes nomenclador, configuración y amortización para las atributos responsabilidad y complejidad están dentro de la categoría baja para un 63.64% del total, mientras que el atributo reutilización cuenta con igual por ciento en la categoría alta, mostrando así que el sistema cuenta con una elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

Relación entre clases (RC)

La métrica RC viene dada por el número de relaciones de uso de una clase con otra. Los criterios y categorías definidos para la evaluación de los atributos de calidad anteriores se presentan en la siguiente tabla:

Tabla 21. Criterios de evaluación para la métrica RC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Promedio.
	Media	Entre Promedio y 2*Promedio.
	Alta	>2*Promedio.
Reutilización	Baja	>2*Promedio.
	Media	Entre Promedio y 2*Promedio.
	Alta	<=Promedio.
Cantidad de pruebas	Baja	<=Promedio.
	Media	Entre Promedio y 2*Promedio.
	Alta	>2*Promedio.

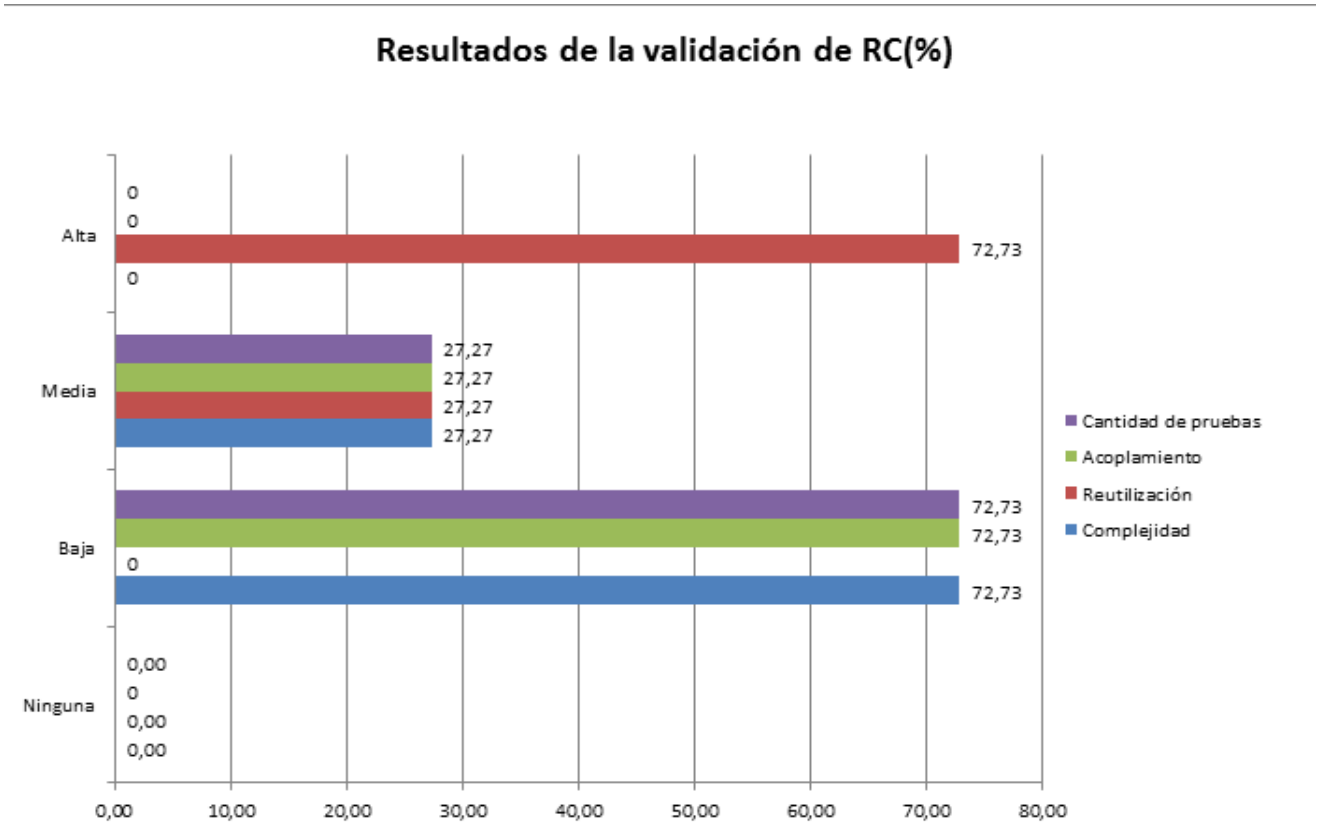


Figura 28. Resultado de la métrica RC

Los resultados obtenidos durante la evaluación del instrumento de medición de la métrica RC demuestran que las clases del diseño poseen un bajo acoplamiento, cantidad de pruebas y complejidad, debido a que para estos atributos la categoría baja es el 72.73 % del total, mostrando igual por ciento en la categoría alta del atributo reutilización. Lo que demuestra que no es necesario un elevado esfuerzo en el momento de realizar cambios, rectificaciones y pruebas a las funcionalidades de los componentes nomenclador, configuración y amortización.

Matriz de inferencia de indicadores de calidad

La matriz de inferencia permite evaluar positiva o negativamente los resultados obtenidos de las relaciones atributo, 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 guión 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. (Ver **tabla 9**).

Tabla 22: Matriz de inferencia de indicadores de calidad

Atributos	TOC	RC	Promedio
Responsabilidad	1	1	1
Complejidad de implementación	1	1	1
Reutilización	1	1	1
Acoplamiento	-	1	1
Complejidad de mantenimiento	-	1	1
Cantidad de prueba	-	1	1

El promedio general es 1 y teniendo en cuenta el rango de evaluación se concluye que los componentes nomenclador, configuración y amortización tienen presenta un diseño aceptable.

2.4 Conclusiones parciales

- En la disciplina de negocio se identificó y describió el proceso de negocio Amortización.
- En la disciplina de requisitos se identificaron 34 requisitos funcionales organizados en 6 grupos de requisitos y 8 no funcionales con los que debe de cumplir la aplicación, este último basado en las características de Xedro-ERP.
- Se validaron los requisitos mediante la Revisión técnica formal donde se obtuvo una carta de aceptación del proyecto, los cuales son los clientes; y los Prototipos de interfaz de usuario, donde se obtuvieron por cada requisito un prototipo evaluado por el cliente.
- En la disciplina de análisis y diseño se obtuvieron 6 diagramas de clases del diseño con estereotipos web, 33 diagramas de secuencias y el modelo de datos.
- Se valida que el diseño es aceptable evidenciándose una cierta posibilidad de reutilización de las clases.

CAPÍTULO 3. Disciplina de implementación y pruebas internas

En el presente capítulo se abordan los elementos relacionados con la implementación y lo referente a las pruebas realizadas al sistema, específicamente a los componentes nomenclador, configuración y amortización con el objetivo de validar su total funcionamiento partiendo del diseño del capítulo anterior. También se expone el diagrama de despliegue y diagrama de componentes, conjuntamente con una breve descripción de cada uno de los elementos que los componen.

3.1 Disciplina de implementación

La implementación es la etapa donde quedan desarrolladas las clases que dan solución al problema planteado. Estas se desarrollaron siguiendo los estándares de codificación definidos para el sistema Xedro-ERP para las descripciones de requisitos descritas en el capítulo anterior.

3.1.1 Diagrama de componentes

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestran la organización y las dependencias lógicas entre un conjunto de componentes de software, sean componentes de código fuente, librerías, binarios o ejecutables. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. Cada diagrama describe un apartado del sistema. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (46).

A continuación se muestra el diagrama de componentes de la propuesta de solución y las relaciones que se establecen entre los componentes, así como una breve explicación del mismo:

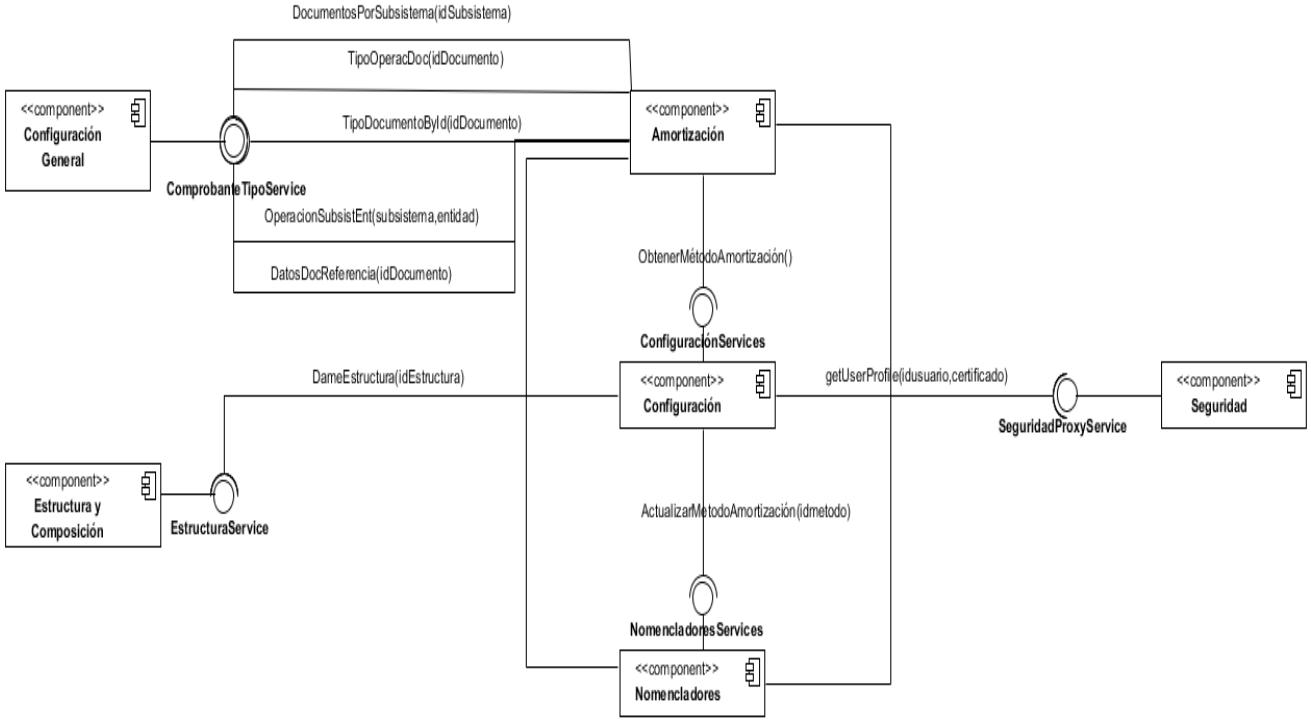


Figura 29. Diagrama de componentes

A continuación se representa la descripción del componente puesto en cuestión y el resto de los componentes externos:

Tabla 23. Descripción del diagrama de componente

Componente externo	Relación	Servicio	Descripción
Seguridad	Componentes nomenclador, configuración y amortización.	getUserProfile(idusuario, certificado)	Permite verificar la autenticación de un usuario; además consumen el servicio.
Estructura y composición	Componentes nomenclador, configuración y amortización.	DameEstructura(idEstructura)	Posibilita constatar la entidad.
Documento de amortización	Componente amortización.	DocumentosPorSubsistema(Idsubsistema)	Permite obtener todos los documentos de un subsistema.
		TipoOperacDoc(idDocumento)	Posibilita todas las

			operaciones de un documento.
		TipoDocumentoByid(idDocumento)	Obtiene tipo de documento.
		OperacionSubsistemaEnti(subsistema, entidad)	Muestra operaciones que se pueden realizar en un subsistema de una entidad.
		DatosDocReferencia(idDocumento)	Permite obtener un documento.

3.1.2 Estructura de empaquetamiento de los componentes nomenclador, configuración y amortización del módulo Activos Fijos Intangibles

Cumpliendo con las decisiones arquitectónicas de la dirección del proyecto ERP-CUBA y del equipo de arquitectura se define una estructura contenedora (ver **Figura 11**) 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.

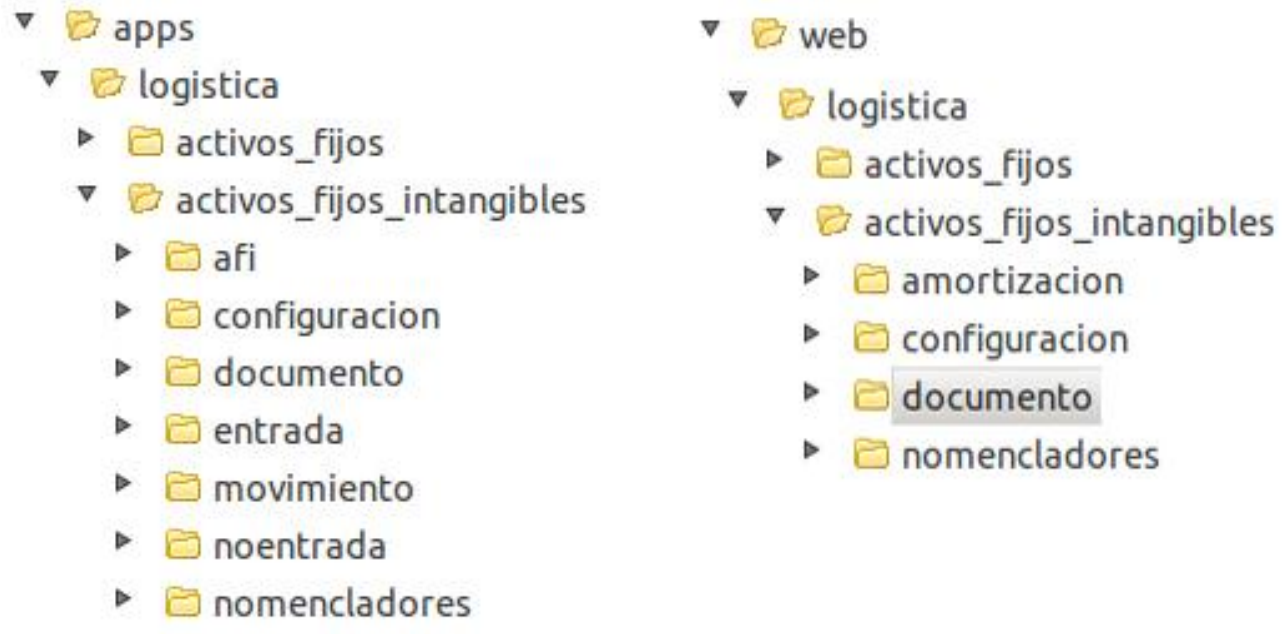


Figura 30. Contenido de las carpetas apps y web para AFI

A continuación se describen cada uno de los elementos representados en la figura:

***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 esta carpeta está comprendido todo lo referente a la configuración específica del subsistema. Contiene en sí la carpeta recursos, la cual a su vez guarda otra denominada xml, ésta incluye los ficheros explicados a continuación:

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

Estructura de los componentes dentro de la carpeta apps:

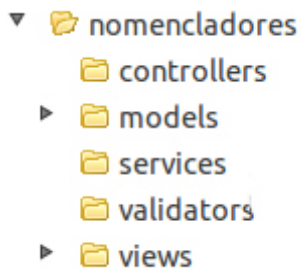


Figura 31. Paquete correspondiente al componente nomenclador en apps

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 clases modelo. 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 marco de trabajo 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.

Estructura de los componentes dentro de la carpeta web:

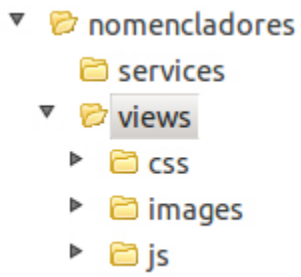


Figura 32. Paquete correspondiente al componente nomenclador en web

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).

3.1.3 Estándares de codificación

Las convenciones o 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 (47). Para el desarrollo de los componentes nomenclador, configuración y amortización del módulo AFI, se utilizarán algunos de los estándares y normas de codificación propuestos como parte de la Línea de Arquitectura determinada para el proyecto ERP-Cuba, los cuales se muestran a continuación:

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: DatDocAfiController.

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.

A continuación se muestran las diferentes nomenclaturas utilizadas en la implementación definida por la Línea de Arquitectura determinada para el proyecto ERP Cuba:

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: Clase controladora: después del nombre llevan la palabra "Controller".

```
11  */
12  class GestamortizacionController extends ZendExt_Controller_Secure {
13
14      function init() {
15          parent::init();
16          $this->model = new DatDocAfiModel();
17      }
```

Figura 33. Nomenclatura según el tipo de clases de tipo controladora

Clases del modelo.

Negocio: Las clases que se encuentran dentro de negocio después del nombre llevan la palabra "Model". A continuación se muestra un ejemplo de esta nomenclatura.

```php
<?php

class DatDocAfiModel extends DocumentoGeneralModel
{
 public function DatDocAfiModel() {
        $this->model = new DatDocAfi();
        parent :: __construct();
    }
```

Figura 34. Nomenclatura según el tipo de clases de tipo modelo

Dominio: El nombre que reciben las clases que se encuentran dentro de dominio es el de la tabla en la base de datos.

```php
<?php
class DatDocAfi extends BaseDatDocAfi
{
    public function setUp()
    {
        parent :: setUp ();
        $this->hasOne ('DatDocumentogeneral', array ('local' => 'id_doc_afi',
        'foreign'=> 'iddocumento'));
    }
```

Figura 35. Nomenclatura según el tipo de clases de tipo dominio

Generado: El nombre de las clases que se encuentran dentro de generados comienza con el prefijo “Base” y continúa con el nombre de la tabla en la base de datos.

```php
<?php

abstract class BaseNomGrupoSubgrupo extends Doctrine_Record
{
```

Figura 36. Nomenclatura según el tipo de clases de tipo generado

Nomenclatura de las funciones: El nombre a emplear para las funciones se escribe con la primera palabra en minúscula y en caso de ser una acción de la clase controladora se debe especificar el nombre de dicha acción en minúscula y seguido el sufijo “Action”.

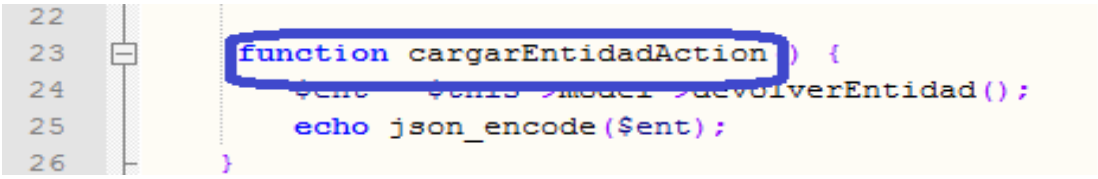


Figura 37. Nomenclatura de las funciones

Nomenclatura de las variables: El nombre a emplear para las variables se escribe con la primera palabra en minúscula.

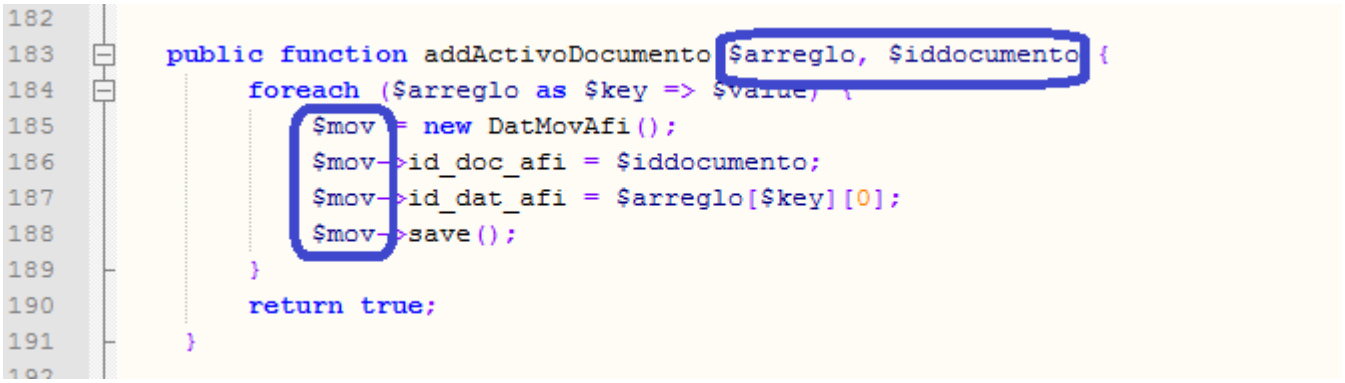


Figura 38. Nomenclatura de las variables

3.2 Disciplina de Pruebas Internas

Las pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del Software, permitiendo aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección.

3.2.1 Prueba de caja blanca

La prueba de caja blanca también se conoce como prueba de caja transparente o de cristal. Consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y a la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento (48).

Las pruebas de caja blanca intentan garantizar que: (48)

- Se ejecutan al menos una vez todos los caminos independientes de cada módulo.
- Se utilizan las decisiones en su parte verdadera y en su parte falsa.
- Se ejecuten todos los bucles en sus límites.

- Se utilizan todas las estructuras de datos internas.

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

- **Prueba del Camino Básico**: permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos.
- **Prueba de Condición**: ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- **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. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.
- **Prueba de Bucles**: se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

Para las pruebas de caja blanca se utilizó la técnica de camino básico que permite obtener una medida de la complejidad de un diseño procedimental, y utilizar esta medida como guía para la definición de una serie de caminos básicos de ejecución, diseñando casos de prueba que garanticen que cada camino se ejecuta al menos una vez cada sentencia del programa (48). Para utilizar esta prueba se hace necesario el cálculo de la complejidad ciclomática de todos los algoritmos que se encuentran implementados en las clases que forman parte de la propuesta de solución. Se enumeran las sentencias de código y a partir de ahí se elabora el grafo de flujo de cada funcionalidad.

A continuación se muestra el requisito funcional de Cancelar Documento de Amortización seleccionado como caso de prueba debido a que tiene una complejidad alta y es una de las funcionalidades principales de la propuesta de solución:

```
public function cancelarEstadoDocumentoAfi($iddocumento) {
    $idDatAfi = $this->pIntegrator->movimientoAFI->dameIdAfibyIdDoc($iddocumento);
    $docGen = parent::Buscar($iddocumento);
    $idEstado = $docGen['idestado'];
    if ($idEstado == 2) {
        foreach ($idDatAfi as $value) {
            $valor = $this->pIntegrator->AFI->dameValorActivobyIdDatAfi($value['id_dat_afi']);
              if($valor[0]['activo'] == 6){
                /*$datAfi = DatAfi::dameFilabyId($value['id_dat_afi']);
                $datAfi-> = 4; //baja elaborada
                $datAfi->save();*///llamar por servicio    cambiarValorActivobyIdDatAfi($value['id_dat_afi']
                $this->pIntegrator->AFI->cambiarValorActivobyIdDatAfi($value['id_dat_afi'], 4);
            }
        }
        parent::CambiarEstado($iddocumento, 1, $version);
        return 1;
    } else {
        return 2;
    }
}
```

Figura 39. Método cancelar documento de amortización

A continuación se muestra el grafo de flujo asociado al requisito funcional CalcularAmortizacion mencionado anteriormente:

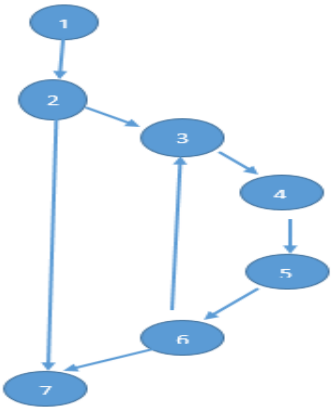


Figura 40. Grafo del flujo asociado al método calcular amortización

Se calcula la complejidad ciclomática a partir de un segmento de código.

La complejidad ciclomática es una métrica del software que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y ofrece un límite superior para el número de pruebas que se deben realizar para asegurar que sea ejecutada cada sentencia al menos una vez (50). Las fórmulas para realizar el cálculo de la complejidad ciclomática son las siguientes:

1. V (G) = (A - N) + 2, V (G) = (8 - 7) + 2 = 3

Siendo A la cantidad total de aristas del grafo y N la cantidad de nodo.

2. V (G) = P +1, V (G) = 2 + 1= 3

Siendo P la cantidad de nodos predicado (son aquellos de los cuales parten dos o más aristas).

3. V (G) = R, V (G) = 3

Siendo R la cantidad de regiones que posee el grafo.

Cada una de las fórmulas V(G) representan el valor del cálculo. A partir de los resultados obtenidos en cada uno, se determina que la complejidad ciclomática del código analizado es 3, que a su vez proporciona el límite superior de pruebas que debe diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones del programa.

A continuación se muestran los caminos básicos por donde puede circular el flujo:

Tabla 24. Caminos básicos del flujo

No.	Caminos básicos
1	1-2-7
2	1-2-3-4-5-6-7
3	1-2-3-4-5-6-7-3-4-5-6

Para cada uno de los caminos obtenidos se realiza un caso de prueba. Los casos de prueba realizados son los siguientes:

Caso de prueba para el camino básico # 1

Camino: 1-2-7

Descripción: los datos de entradas cumplirán con los siguientes requisitos: se registren los datos. A continuación se muestra los datos de entrada del requisito.

Entrada:

$POST= [[idEstado] =>! 2, [id_mov_afi] => 9000000384, [id_dat_afi] => 9000000001]

Resultados esperados: se espera que se cancele el estado del documento de amortización y se actualicen los datos del documento.

Resultados obtenidos: No satisfactorio.

Caso de prueba para el camino básico # 2

Camino: 1-2-3-4-5-6-7

Descripción: los datos de entradas cumplirán con los siguientes requisitos: se registraron cada uno de los datos que aparecen en el formulario. A continuación se muestra los datos de entrada del requisito.

Entrada:

$POST= [[idEstado] => 2, [id_mov_afi] => 9008942384, [id_dat_afi] => 9008912001]

Resultados esperados: se espera que se cancele el estado del documento de amortización y se actualicen los datos del documento.

Resultados obtenidos: satisfactorio.

Caso de prueba para el camino básico # 3

Camino: 1-2-3-4-5-6-7-3-4-5-6

Descripción: los datos de entradas cumplirán con los siguientes requisitos: se registraron cada uno de los datos que aparecen en el formulario. A continuación se muestra los datos de entrada del requisito.

Entrada:

$POST= [[idEstado] => 2, [id_mov_afi] => 9555000384, [id_dat_afi] => 9000888001]

Resultados esperados: se espera que se cancele el estado del documento de amortización y se actualicen los datos del documento.

Resultados obtenidos: satisfactorio.

A partir de la aplicación de los casos de prueba expuestos anteriormente se comprobó que el flujo de trabajo de las funcionalidades es correcto, ya que cada sentencia es ejecutada al menos una vez

con los parámetros de entrada y las salidas esperadas. Las pruebas de caja blanca permitieron demostrar que la implementación realizada no presenta errores. Se examinó el estado de la aplicación en varios puntos de la misma determinando que el estado real coincidía con el esperado.

3.2.2 Pruebas de caja negra

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software. O sea, la prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. 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 (50).

Técnica utilizada:

Partición de equivalencia: Técnica que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar (50).

A continuación se muestra un ejemplo del caso de prueba adicionar documento de amortización, en caso de existir alguna duda dirigirse al expediente de proyecto donde se archivaran las demás casos de prueba. Donde su objetivo es demostrar que las funciones del software son operativas, que la entrada y la salida se producen de forma correcta y que además se mantiene la integridad de la información.

Tabla 25. Descripción de caso de pruebas de Adicionar documentos de amortización.

Escenario	Descripción	Observaciones	Respuesta del sistema	Flujo central
EP 1.1: Adicionar un documento de amortización correctamente.	El sistema debe permitir adicionar un documento de amortización.	V(Activos)	El sistema muestra el siguiente mensaje de información: ¨El documento fue adicionado satisfactoriamente ¨.	Se introducen los datos correctamente. Se presiona el botón Aceptar. Se muestra un mensaje de información,

EP 1.2: Adicionar un documento de amortización presionando la opción Aplicar.		V(Activos)	El sistema muestra el siguiente mensaje de información: ¨El documento fue adicionado satisfactoriamente¨.	Se introducen los datos correctamente. Se presiona el botón Aplicar. Se muestra un mensaje de información.
EP 1.3: Adicionar un documento de amortización con valores inválidos presionando la opción Aceptar o Aplicar.		I(jhbsfgvjkdfhb fsjh^$^&%&95664 56479587jsdkzf uyg534674^$&^% &(texto con más de 255 caracteres))	El sistema muestra el siguiente mensaje de error: “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”	Se introducen los datos incorrectamente. Se presiona el botón Aceptar o Aplicar. Se muestra un mensaje de error.
EP 1.4: Adicionar un documento de amortización dejando campos vacíos.		V(vacío)	El sistema muestra el siguiente mensaje de información: “Por favor verifique nuevamente que hay campo(s) con valor(es) incorrecto(s).”	Se introducen los datos dejando campos vacíos. Se presiona el botón Aceptar. Se muestra un mensaje de información.
EP 1.5: Cancelar.		N/A	El sistema cierra la interfaz.	Se introducen los datos correctamente. Se presiona el botón Cancelar.

3.2.3 Resultados de las Pruebas

Con el objetivo de que las funcionalidades correspondientes a los componentes nomenclador, configuración y amortización tengan un correcto funcionamiento, se realizaron las pruebas en tres iteraciones. Durante la primera iteración se detectaron cinco errores de prioridad media y dos errores en la prioridad alta. En la segunda iteración se detectaron dos errores en la categoría baja, conjuntamente con uno de la categoría media. A los errores identificados en las dos primeras iteraciones se solucionaron rápidamente. Para finalizar en la tercera iteración no se detectaron no conformidades, es decir, que se verificó que se le ha dado solución a la problemática planteada con resultados satisfactorios. A continuación se muestra las iteraciones efectuadas y la cantidad de errores identificadas.

Tabla 26. Iteraciones de las pruebas

Iteración	Ortografía	Validación	Funcional	Prioridad
1	6		5	Significativa
		4		No significativa
2	2		1	Significativa
		1		No significativa
3	0	0	0	Significativa
	0	0	0	No significativa

Todo lo antes expuesto dio pasos a la liberación de la aplicación para lo cual los especialistas del centro en conjunto con la dirección del equipo de desarrollo firmo el acta de liberación (ver **Anexo 1**).

3.3 Conclusiones parciales

- En la disciplina de implementación se obtuvo el diagrama de componentes representando los componentes desarrollados para el presente trabajo y las relaciones con los componentes de otros subsistemas a través de los servicios. Además, se representa la estructura de paquetes de los archivos obtenidos en la implementación de la solución.
- Se realizaron pruebas de caja negra mediante la técnica de partición de equivalencia a la solución informática con el equipo de trabajo del proyecto. Durante las 3 iteraciones de pruebas se detectaron un conjunto de no conformidades las cuales fueron resultas después de culminada cada iteración.
- Las pruebas de caja blanca realizadas a un fragmento de código de la implementación arrojaron una complejidad ciclomática de 3 unidades dando lugar a 3 caminos básicos, comprobando que eran los únicos por los que siempre transitaba el flujo.

Conclusiones Generales

Una vez finalizado el presente trabajo de diploma donde se abarca todo el proceso de desarrollo del módulo activo fijo intangible, específicamente los componentes nomenclador, configuración y amortización para el subsistema de Logística, se puede concluir que se le dio cumplimiento a todos los objetivos propuestos, resaltando que:

El estudio de los principales conceptos relacionados con la gestión de los AFI permitió sentar las bases para el desarrollo de los componentes nomenclador, configuración y amortización para el módulo AFI del Xedro-ERP.

El estudio realizado sobre los sistemas ERP a nivel nacional e internacional demostró que los sistemas estudiados no cumplen con los principios de independencia tecnológica del país, o carecen de funcionalidades necesarias para llevar una completa gestión y control sobre los activos intangibles. Sin embargo poseen características que sirvieron como base para elaborar la propuesta de solución.

A partir de la realización del análisis y diseño de los componentes nomenclador, configuración y amortización se obtuvo como resultado los diagramas y artefactos necesarios para guiar el desarrollo de los mismos.

La disciplina de implementación proporcionó la creación de funcionalidades y la integración satisfactoria de las mismas al Xedro-ERP, lo que implica una gestión de los componentes nomencladores, configuración y amortización dando cumplimiento a los 33 requisitos funcionales y 8 no funcionales.

Las pruebas de caja blanca aplicadas a la solución demostraron que el flujo de trabajo de las funcionalidades es correcto, ya que se examinó el estado de la aplicación en varios puntos, determinando que el estado real coincidía con el esperado.

Las pruebas de caja negra realizadas a la solución permitieron confirmar la estabilidad de los procesos y el correcto funcionamiento de los componentes agregados a la aplicación.

El módulo AFI del Sistema Integral de Gestión Xedro-ERP en el que se incluyen los componentes nomenclador, configuración y amortización permite llevar un control de los activos fijos intangibles en las empresas que utilicen este sistema.

Recomendaciones

Una vez terminada la implementación de los componentes y analizado los resultados obtenidos del presente trabajo, se mantiene la idea de continuar perfeccionando la solución final, por lo que se recomienda:

- Continuar con el estudio de los procesos relacionados con los AFI con el fin de obtener mayores conocimientos e identificar soluciones comunes para las entidades.
- Identificar nuevos requisitos para actualizar el sistema desarrollado, garantizando mayor funcionalidad y aceptación del cliente.

Bibliografía

1. Ley 38 de la contabilidad. 1998.

2. elegirERP. [En línea] [Citado el: 22 de 03 de 2014] http://www.elegirerp.com/definicion-erp.

3. Thompson, Ivan. Promonegocios.net. [En línea] [Citado el: 22 de 04 de 2014] http://promonegocios.net.

4. ACTIVOS INTANGIBLES. [En línea] 26 de 01 de 2012. [Citado el: 22 de 04 de 2014] http://activosintangiblesforo.blogspot.com/2012/01/concepto-tradicional-de-activos_26.html.

5. gerencie. [En línea] [Citado el: 24 de 03 de 2014] http://www.gerencie.com/activos-intangibles.html.

6. [En línea] [Citado el: 20 de 05 de 2014] http://investigacion.contabilidad.unmsm.edu.pe/archivospdf/NIC/NIC38_04.pdf.

7. Real Academia Española [En línea] [Citado el: 14 de 05 de 2014] http://lema.rae.es/drae/?val=configuracion.

8. En línea] [Citado el: 25 de 05 de 2014] https://www.google.com.cu/?gws_rd=ssl#q=nomencladores+definicion&revid=784233885.

9. economic [En línea] [Citado el: 22 de 03 de 2014] http://www.economic.es/programa/glosario/definicion-amortizacion.

10. plangeneralcontable. [En línea] [Citado el: 25 de 05 de 2014] http://www.plangeneralcontable.com/?tit=amortizacion-de-inmovilizado-introduccion-a-los-sistemas-de-amortizacion&name=Manuales&fid=el0bcab.

11. Manual de infraestructura productiva. IPL-4020:2009, Declaraciones legales en las creaciones UCI, Sección: 02 – Infraestructura Productiva, Capítulo: 02.18, Instrucción 02. 18.06.

12. IPP-4000:2008, Registro de la Propiedad Intelectual, Sección: 02 – Infraestructura Productiva, Capítulo: 02.18, Instrucción 02.18.06.

13. Reglamento del decreto de Ley No 203, De “marcas y otros signos distintivos “, Gaceta Oficial de la República de Cuba.

14. Ley 36 de la contabilidad, 1998.

15. Assets. [En línea] 2004. http://www.assets.co.cu/.

16. Versión 3.0 Manual de Usuario Capítulo VIII Activos Fijos, Assets.

17. OpenERPSPAIN. [En línea] [Citado el: 24 de 05 de 2014] http://openerpspain.com/.

18. openerpsite. [En línea] [Citado el: 25 de 05 de 2014] http://www.openerpsite.com/tag/modulo-activos-openerp/.

19. econcept sistemas. [En línea] [Citado el: 25 de 05 de 2014.] http://econcept.es/why-openerp.

20. DISAIC casa consultora. [En línea] [Citado el: 25 de 04 de 2012] http://www.disaic.cu/modules.php?name=content&pa=showpage&pid=818.

21. citmatel [En línea] [Citado el: 25 de 04 de 2014.] http://www.citmatel.cu/servicios.php.

22. Obregón, Ing. William González. Modelo de desarrollo del Centro CEIGE. Cuidad de la Habana: s.n. 2012.

23. [En línea] [Citado el: 25 de 04 de 2014.] Jacobson, Ivar, Boch, Grady, James. El Proceso Unificado de Desarrollo de Software.Lenguaje UML 2000. pág. 13.

24. López, Patricia. Herramienta CASE Visual Paradigm. [En línea] 2013. [Citado el: 19 de 05 de 2014] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/practicas-1/is1-p01-trans.pdf.

25. cientec.com. [En línea] [Citado el: 25 de 04 de 2014.] UML: Un Lenguaje Modelo. CIENTEC. Grupo ENTEL. [En línea] 2013. [Citado el: 12 de Enero de 2014] http://www.cientec.com/analisis/ana-uml.html.

26. Manager. Free Download Manager. [En línea] 5 de Marzo de 2007. [Citado el: 20 de Mayo de 2014.].

27. Servidor de Aplicación, Apache. [En línea] [Citado el: 04 de 11 de 2013] http://www.digitallearning.es/blog/apache-servidor-web-configuracion-apache2-conf/.

28. NetBeans. [En línea] [Citado el: 28 de 04 de 2014] http://netbeans.org/index_es.html.

29. Mozilla Firefox. [En línea] [Citado el: 04 de noviembre de 2012] http://www.ecured.cu/index.php/Mozilla_Firefox.

30. Mozilla Firefox. GetFirefox. [En línea] 2009. [Citado el: 22 de 11 de 2012] http://www.getfirefox.es/firefoxfeatures.

31. Subversion. [En línea] [Citado el: 02 de noviembre de 2012] http://svnbook.red-bean.com/nightly/es/svn-book.pdf.

32. Ing.Oiner Gómez Baryolo, Ing.Yoandry Morejón Borbón, Ing.Darien García Tejo.ARQUITECTURA TECNOLÓGICA PARA EL DESARROLLO DE SOFTWARE. [En línea] 2009 [Citado el: 10 de Enero de 2013].

33. O.E.Sánchez, B.Garea,S.Lantigua y otros.Sistema de Información para la Gestión de Programas de Ciencia e Innovación en Cuba. 2008. [Citado el: 10 de Enero de 2013].

34. Sencha. 2008. Sencha. Designer. *Sitio Web oficial de Sencha Ext Js.* [En línea] 2008. [Citado el: 15 de Enero de 2014.] http://www.docs.sencha.com/designer/.

35. PHP. [En línea] [Citado el: 15 de noviembre de 2014] http://ivancamayo.files.wordpress.com/2010/09/php1.pdf.

36. PostgreSQL. The world's most advanced open source data base. PostgreSQL. The world's most advanced open source data base. [En línea] [Citado el: 05 de Marzo de 2014] http://www.postgresql.org.

37. RUMBAUHG J JACOBSON I El Lenguaje Unificado de Modelado. Manual de Referencia Book]. Madrid: Pearson Educación, 2000.

38. CVOSOFT. Ingeniería en Sistemas. [En línea] Editorial CVOSOFT. http://www.cvosoft.com/sistemas_sap_abaprecursos_tecnicos_abapque-es-sap-mm.php.

39. Sommerville, Ian y Sawyer, Pete. 1997. Requirements Engineering: A good practice guide. Fayetteville, AR, USA. 1997. ISBN: 9780471974444.Lancaster: s.n.Citado el 1 de febrero de 2014.

40. Sommerville, Ian. 2005. Ingeniería de Software.7ma. 2005.31.

41. Reynoso, B. 2004. Architect Academy Seminario de Arquitectura de Software. 2004.

40. Baryolo, Oiner Gómez. 2010. Solución informática de autorización en entornos multientidad y multisistema. Universidad de las Ciencias Informáticas. La Habana: s.n., 2010. Tesis de Maestría.

41. Larman, Craig. El mundo Informático. El mundo Infoprocesos relacionados con los AFI con el fin de obtener mayores conocimientos e identificar soluciones común mático. [En línea] 17 de Agosto de 2006. [Citado el: 21 de Enero de 2013] http://jorgesaavedra.wordpress.com/2006/08/17/patrones-grasp-craig-larman/.

42. Mundo Informático. [Citado el: 21 de Enero de 2013.] https://infow.wordpress.com/category/patrones-de-disenogof/.

43. Pressman, R. 1999.Software Engineering.A Practitioner's Approach. USA: s.n., 1999.

44. Scribd [En línea] [Citado el: 15 de noviembre de 2012] http://es.scribd.com/doc/15493687/DIAGRAMAS-DE-SECUENCIA.

45. CVOSOFT. Ingeniería en Sistemas. [En línea] Editorial CVOSOFT. http://www.cvosoft.com/sistemas_sap_abaprecursos_tecnicos_abapque-es-sap-mm.php.

46. Pressman, R. 1999.Software Engineering.A Practitioner's Approach. USA: s.n., 1999.

47. Arquitectura de Software Vista de Integración, Proyecto Sauxe.

48. Larman, Graig. 2004. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. UML y Patrones. 2004.

50. PRESSMAN, R. S. Ingeniería de Software. Vol. Quinta Edición.

Anexos

Anexo 1: Carta de aceptación

UCI Informáticas

CENTRO DE INFORMATIZACIÓN DE GESTIÓN DE ENTIDADES

Producto: Cedrux.
Módulo: Activos Fijos Intangibles.
Componentes: Nomenclador, configuración y amortización de AFI.
Involucrados en el proceso:

- **Estudiante:** Yeni Veliz.
  Geider Arévalo.
- **Tutores:** Ing. Annia Verdecia Boza.
  Ing. Carlos A. Giralt Torriente.
- **Especialistas del proyecto (Cedrux):** Ing. Annia Verdecia Boza.
  Ing. Leodanny Wilbert Polanco.

Observaciones del proceso:

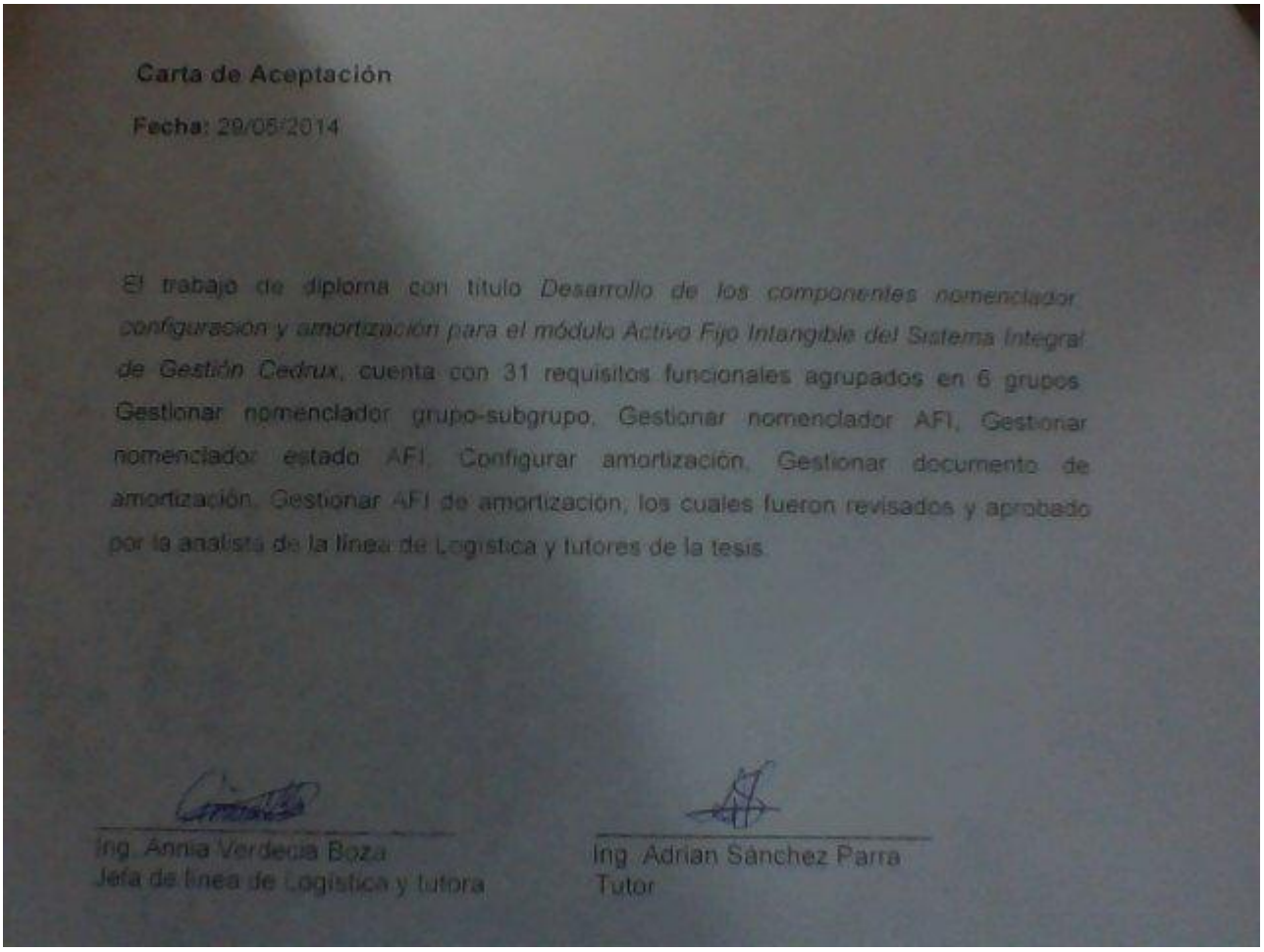
El trabajo de diploma con título "Desarrollo de los componentes nomenclador, configuración y amortización para el módulo Activo Fijo Intangible del Sistema Integral de Gestión XEDRO-ERP" de los estudiantes Yeni Veliz y Geider Arévalo, se le realizaron pruebas de caja negra con la técnica de partición de equivalencia de tres iteraciones, donde en cada una de ellas se detectaron no conformidades con errores funcionales, ortográficos, de interfaz de usuario, entre otros. Teniendo en cuenta que las No Conformidades han sido debidamente resueltas y validada la eficacia de la corrección por parte de los especialistas del proyecto, se ha tomado el acuerdo de Aceptar la solución con fecha 11 de Junio de 2014.

Para que conste la Aceptación de los resultados de las pruebas y por tanto la Aceptación de los entregables especificados, dando fe del acuerdo, se extiende la presente **Acta** en dos (2) ejemplares, rubricados por los principales.

Representantes de las Partes.

Entrega	**Recibe**
Nombre y Apellidos: Yeni Veliz.	**Nombre y Apellidos:** Ing. Annia Verdecia
Firma:	**Cargo:** Analista.
Nombre y Apellidos: Geider Arévalo.	**Firma**
Firma:	**Nombre y Apellidos:** Ing. Judiht García.
	Cargo: Jefa de Proyecto.
	Firma

Anexo 2: Revisión técnica formal a los documentos y artefactos



Carta de Aceptación

Fecha: 29/05/2014

El trabajo de diploma con título *Desarrollo de los componentes nomenclador configuración y amortización para el módulo Activo Fijo Intangible del Sistema Integral de Gestión Cedrux*, cuenta con 31 requisitos funcionales agrupados en 6 grupos Gestionar nomenclador grupo-subgrupo, Gestionar nomenclador AFI, Gestionar nomenclador estado AFI, Configurar amortización, Gestionar documento de amortización, Gestionar AFI de amortización, los cuales fueron revisados y aprobado por la analista de la línea de Logística y tutores de la tesis

Ing. Annia Verdecia Boza
Jefa de línea de Logística y tutora

Ing. Adrian Sánchez Parra
Tutor