Universidad de las Ciencias Informáticas

“Facultad de Ciencias y Tecnologías Computacionales”

“Sistema para la gestión de bajas técnicas de los activos fijos tangibles”

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor: Luis Miguel Fonseca Ponce

Tutores: MsC. Omar Mar Cornelio

Ing. David Cervantes Pérez

La Habana, 2017

“Año 59 de la Revolución”

DECLARACIÓN DE AUTORÍA

Declaro ser autor del trabajo ***"Sistema de gestión de bajas técnicas de los activos fijos tangibles"*** y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Luis Miguel Fonseca Ponce

_____

Firma del Autor

Msc. Omar Mar Cornelio

_____

Firma del Tutor

Ing. David Cervantes Pérez

_____

Firma del Tutor

DATOS DE CONTACTO

Autor:

Luis Miguel Fonseca Ponce

Universidad de las Ciencias Informáticas

Correo electrónico: lmfonseca@estudiantes.uci.cu

Tutor:

Msc. Omar Mar Cornelio

Universidad de las Ciencias Informáticas

Correo electrónico: omarmar@uci.cu

Tutor:

Ing. David Cervantes Pérez

Universidad de las Ciencias Informáticas

Correo electrónico: dacert@uci.cu

DEDICATORIA

Le dedico esta a tesis a mi abuela, mi mamá, mi abuelo, mis hermanos, mi primo, mi tio y a la familia de mi novia.

"Sólo llegarán a la cima, los que en su empeño tengan fe."

Ernesto Guevara de la Serna

Resumen

Las organizaciones cubanas, como base y sostén de la economía del país, son las encargadas de llevar el control de sus Activos Fijos Tangibles (AFT). En estos momentos la gestión de los AFT en la Universidad de las Ciencias Informáticas se realiza de forma manual por parte del personal encargado de manejar la información, almacenando los datos en formato duro, lo que resulta muy engorroso a la hora de manipular los datos, ya sea para realizar una búsqueda, modificar algún dato, obtener reportes, realizar revisiones; afectando la calidad y rapidez del proceso. Todo esto conlleva a que se produzcan una serie de irregularidades entre las que se incluyen, la pérdida de los medios y bienes debido a la mala planificación y asignación de los mismos. Con la solución que se propone en el presente trabajo se pretende dar respuesta a esta problemática, mediante la creación de un sistema informático para la gestión de bajas técnicas de los AFT. La presente investigación tiene como objetivo realizar el análisis, diseño, implementación y evaluación de un sistema informático, para la informatización del proceso de bajas técnicas de los AFT. Para llevar a cabo la propuesta de solución se utilizó la metodología de desarrollo AUP-UCI y como principal tecnología de desarrollo el *framework* Symfony 2.8, haciendo uso del lenguaje de programación PHP en su versión 5.5.

Palabras Claves: Activos fijos tangibles, sistema informático, toma de decisiones.

Abstract

Cuban organizations, as the base and support of the country's economy, are in charge of taking control of their Tangible Fixed Assets (TFA). At the moment the management of the TFA in the University of Computer Science is done manually by the personnel in charge of managing the information, storing the data in hard format, which is very cumbersome when manipulating the data, Whether to perform a search, modify some data, obtain reports, perform reviews; thus affecting the quality and speed of the process. All this leads to a series of irregularities, including loss of means and assets due to poor planning and allocation. With the solution proposed in the present work it is intended to respond to this problem, through the creation of a computer system for the management of low techniques of TFA. The present research aims to perform the analysis, design, implementation and evaluation of a computer system for the computerization of the process of low techniques of AFT. In order to carry out the solution proposal we used the AUP-UCI development methodology and as the main development technology the framework Symfony 2.8, making use of the programming language PHP in its version 5.5.

Keywords: Tangible Fixed Assets, computer system, decision making.

Índice

Índice de Figuras

Índice de Tablas

Introducción

A partir del perfeccionamiento empresarial y haciendo uso de las Tecnologías de la Información y las Comunicaciones (TIC), la sociedad ha visto nacer una era marcada por nuevas tendencias y mecanismos en el desarrollo de sus procesos. A nivel global, las empresas e instituciones han desarrollado técnicas y métodos, con el fin de incrementar al máximo la eficiencia y competitividad sobre la base de otorgarles facultades y atribuciones necesarias para un mejor manejo y control de sus medios. Su correcta utilización es capaz de definir en la actualidad una posición importante en el ámbito competitivo de la industria, el mercado y los servicios (Echevarría, 2010).

Las organizaciones cubanas, como base y sostén de la economía del país, son las encargadas de llevar el control del estado de sus Activos Fijos Tangibles (AFT), con la necesidad de evitar la pérdida de recursos lo que trae como consecuencia afectación en la economía. El control es un elemento muy importante dentro de cualquier organización, pues es el encargado de evaluar los resultados y saber si estos son adecuados a los planes y objetivos que desea conseguir la empresa. Solo a través de esta función se pueden precisar los errores, identificar a los responsables y corregir las fallas, para que la organización se encuentre encaminada de manera correcta.

La Universidad de las Ciencias Informáticas (UCI) es una entidad que posee varios centros productivos, laboratorios docentes, aulas y diferentes áreas donde se maneja un gran número de activos. A pesar de su avance informático, la gestión de bajas técnicas de los AFT se realiza de forma manual por parte del personal encargado de manejar la información, siendo esta almacenada en formato duro, lo cual resulta engorroso a la hora de manipular los datos ya sea para realizar una búsqueda, modificar algún dato, obtener reportes y revisiones afectando así la calidad y rapidez del proceso, la cual tiene la necesidad de contar con un mecanismo que permita controlar los AFT; todo esto puede traer consigo que:

- La búsqueda y recuperación de datos para la generación de reportes es lenta.
- El insuficiente control de los medios en proceso de bajas técnicas genera desactualización en la contabilidad.
- El deterioro de los documentos asociados al proceso de baja técnicas ocasiona la pérdida de información.

- Las demoras en el proceso de bajas técnicas imposibilita que los involucrados conozcan el estado de su solicitud.

Lo antes expuesto, dificulta y demora la gestión de bajas técnicas de los AFT a los trabajadores de la UCI lo cual conlleva a la búsqueda de una solución al siguiente **problema de investigación:** ¿Cómo contribuir a la gestión de bajas técnicas de AFT en la UCI, para garantizar un adecuado control de los mismos?

Partiendo del problema expuesto el **objeto de estudio:** se enfocará a los procesos asociados a la gestión de bajas técnicas de AFT, enmarcado en el **campo de acción:** la gestión de bajas técnicas de AFT en la UCI.

Para resolver el problema planteado con anterioridad se establece como **objetivo general:** desarrollar un sistema informático que agilice el proceso de gestión de bajas técnicas de AFT en la UCI, para garantizar un adecuado control de los mismos.

Partiendo del análisis del objetivo general se derivan los siguientes **objetivos específicos:**

- Analizar de los conceptos técnicos implicados para el desarrollo del sistema de gestión de bajas técnicas de los AFT de la UCI.
- Realizar el análisis del sistema de gestión de bajas técnicas de los AFT de la UCI.
- Realizar el diseño del sistema de gestión de bajas técnicas de los AFT de la UCI.
- Implementar las funcionalidades del sistema de gestión de bajas técnicas de los AFT de la UCI.
- Validar las funcionalidades del sistema de gestión de bajas técnicas de los AFT de la UCI.

Con el propósito de dar cumplimiento a los objetivos planteados se definen las siguientes **tareas de investigación**:

- Elaboración del marco teórico de la investigación para definir las bases teórico – científicas del desarrollo del sistema de gestión de bajas técnicas de los AFT en la UCI.
- Análisis de las herramientas que se utilizan en la actualidad para la gestión de bajas técnicas de los AFT en la UCI.
- Selección de las herramientas y tecnologías para el desarrollo del sistema de gestión de bajas técnicas de los AFT en la UCI.

- Especificación de los Requisitos de software del sistema de gestión de bajas técnicas de los AFT para describir las características del sistema.
- Definición de la arquitectura del sistema de gestión de bajas técnicas de los AFT para establecer la guía de implementación del sistema.
- Implementación de las funcionalidades del sistema de gestión de bajas técnicas de los AFT para dar respuesta a los requisitos del mismo, haciendo uso de los lenguajes de programación.
- Elaboración de los casos de prueba que serán aplicados al sistema de gestión de bajas técnicas de los AFT para comprobar el correcto funcionamiento de la solución encontrada.
- Ejecución de las pruebas a las funcionalidades del sistema de gestión de bajas técnicas de los AFT para detectar posibles defectos mediante la ejecución de pruebas al software.

Para guiar el proceso investigativo se plantean las siguientes **preguntas científicas**:

- ¿Cuáles son los fundamentos teórico – técnicos en los que se basa el proceso de desarrollo del sistema de gestión de bajas técnicas de los AFT?
- ¿Qué tecnologías, metodología y herramientas utilizar para el desarrollo del sistema de gestión de bajas técnicas de los AFT?
- ¿Cómo estructurar el proceso de implementación del sistema de gestión de bajas técnicas de los AFT que permita una mejor representación de los componentes a desarrollar?
- ¿Cómo validar el correcto funcionamiento del sistema de gestión de bajas técnicas de los AFT?
- ¿Qué pruebas aplicar para comprobar el correcto funcionamiento del sistema de gestión de bajas técnicas de los AFT?

Para el cumplimiento del objetivo general planteado se utilizarán los siguientes **métodos de investigación**:

Métodos teóricos:

- **Histórico y Lógico:** Mediante este método se analiza la trayectoria real de los fenómenos, su evolución y desarrollo. Se utiliza este método ya que entre los objetivos propuestos está identificar los procesos relacionados con la gestión de bajas técnicas de los AFT, además se utiliza para investigar sobre sistemas de gestión que tengan similitud con el que se va a implementar.

- **Analítico-Sintético:** Facilita el entendimiento del fenómeno en que se trabaja, es más útil la división de estas en diferentes fases y de esta forma descubrir sus características generales, lo que ayuda a seguir una correcta investigación. A partir del estudio y análisis documentación y en diversas fuentes bibliográficas sobre la implementación de sistemas de gestión y el uso de herramientas y tecnologías relacionados con la gestión de bajas técnicas de los AFT de la Universidad de las Ciencias Informáticas (UCI).
- **Modelación:** El modelo científico es un instrumento de la investigación de carácter material o teórico, creado para reproducir el fenómeno que se está estudiando. Se modelan todos los procesos relacionados con la gestión de bajas técnicas de los AFT de la Universidad de las Ciencias Informáticas (UCI).

Métodos empíricos:

- **Observación:** Es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. A través de este método se realiza la captura sistemática de información sobre la situación actual, lo que permite comprender mejor el comportamiento externo del fenómeno en cuestión; lo que constituye un apoyo fundamental para la posible realización de un sistema informático para la gestión de bajas técnicas de los AFT de la Universidad de las Ciencias Informáticas.

A continuación, se presenta la estructura del Trabajo de Diploma, contenido por tres capítulos, donde se exponen los resultados del estudio realizado referentes a los sistemas existentes de esta índole, logrando obtener las herramientas y metodología necesaria para la realización del diseño e implementación del sistema para la gestión de bajas técnicas de los AFT con el objetivo de cumplir las necesidades del cliente.

Capítulo 1 Fundamentación teórica del sistema de gestión de bajas técnicas de los AFT: En este capítulo se exponen los principales conceptos asociados a la solución en cuestión. Se realiza un análisis de las principales soluciones existentes para así identificar posibles características de las mismas que puedan aportar al diseño y desarrollo de la aplicación. Por último, se define la metodología, así como las herramientas que se utilizarán en la implementación y el futuro despliegue.

Capítulo 2 Características del sistema de gestión de bajas técnicas de los AFT: En este capítulo se definirán los procesos que se van a automatizar, se define el modelo de negocio y los diagramas para lograr una mejor comprensión del mismo. Además, se determinan y especifican los requisitos funcionales y no funcionales, se modela el sistema y se identifican los casos de uso con sus respectivos diagramas para

conocer las características del sistema. Se especificarán además los patrones utilizados, así como otras características fundamentales como parte del diseño, generándose además un grupo de artefactos teniendo en cuenta la metodología a utilizar.

Capítulo 3 Implementación y prueba del sistema de gestión de bajas técnicas de los AFT*:* En este capítulo se describe el proceso de implementación y pruebas de la solución. Se especifican la selección, diseño y aplicación de las pruebas realizadas en correspondencia a la tipología de la solución. Finalmente se muestran los resultados obtenidos una vez concluida la ejecución de los ensayos de validación.

Capítulo 1: Fundamentación teórica del sistema de gestión de bajas técnicas de los AFT

Introducción

En este capítulo se exponen los principales conceptos asociados a la solución en cuestión. Se realiza un análisis de las principales soluciones existentes para así identificar posibles características de las mismas que puedan aportar al diseño y desarrollo de la aplicación. Por último, se define la metodología, así como las herramientas que se utilizarán en la implementación y el futuro despliegue.

1.1 Principales conceptos asociados a la investigación

A continuación, se exponen los principales conceptos relacionados con el marco teórico de la investigación, con el objetivo de profundizar en los distintos puntos de vista, para enriquecer los conocimientos sobre la gestión de bajas técnicas de los AFT.

Activos Fijos Tangibles y vida útil

Los AFT son aquellos bienes tangibles que adquiere una empresa para hacer uso constante de ellos. En la producción, comercialización o administración son activos que contribuyen en la explotación económica de la empresa.

La vida útil de un activo fijo tangible no es más que el período durante el cual se espera utilizar el activo por parte de la empresa o bien el número de unidades de producción o similares que se espera obtener del mismo por parte de la empresa (AFT, 2012).

Proceso de bajas técnicas de los AFT

El proceso de bajas técnicas de los AFT tiene como objetivo retirar definitivamente todos los medios que, por su obsolescencia o daño, ya no presta servicios a la institución para el normal desarrollo de sus actividades. Este se aplica al proceso de la UCI, relacionados con la asignación, responsabilidad y manejo de los AFT (Administracion Universitaria, 2012).

Las bajas de los AFT inservibles no es más que el retiro definitivo de medios que por su desgaste, rotura u obsolescencia física no son útiles para la entidad. Se entiende por destrucción, la extinción total del medio y el inventario es la relación ordenada, completa y detallada de toda clase de AFT de la UCI. Permite

verificar, clasificar, analizar y controlar mediante información confiable y oportuna los medios de la entidad y así evitar su pérdida, deterioro o desperdicio del medio (Administracion Universitaria, 2012).

Gestión de inventario

Es el conjunto de técnicas, métodos y estrategias, utilizados para administrar los materiales existentes dentro de una empresa y de los cuales depende su actividad económica, los inventarios son de suma importancia dentro de una organización, ya que a través de ellos se pueden obtener las ganancias que la empresa espera en un ejercicio económico (TributosNet, 2016). La gestión de inventario es un elemento importante dentro cualquier entidad, ya que la misma permite un buen control de los AFT, evitando perdida y desvíos de los mismos.

Sistema de gestión de control

Se entiende por control de gestión al conjunto de procesos que la empresa aplica para asegurarse de que las tareas que en ella se realizan están encaminadas a la consecución de los objetivos previamente establecidos en el proceso de planeamiento estratégico. El control de gestión no se limita a comprobar que las tareas realizadas o las decisiones tomadas han sido correctas, sino que parte de su cometido es influir y orientar el comportamiento de la organización para que se alcancen los objetivos propuestos (Polanco, 2012). La gestión de control tiene una estrecha relación con la gestión de inventario, ya que para tener un bueno control de los medios y bienes con la que cuenta una institución, se necesita la realización de inventarios.

1.2 Sistemas informáticos existentes asociados a la gestión de bajas técnicas de AFT

Se investigará todo lo referente a los sistemas y herramientas relacionadas con la gestión de bajas técnicas. Las mismas tienen como objetivo gestionar el control de los medios de una institución así como la gestión de inventario para un mejor manejo y control de los bienes.

Herramienta Sorolla2

Dentro del sistema *SOROLLA2*, el módulo de gestión de inventario (GDI), se propone como una herramienta perfectamente integrada dentro de *SOROLLA2* para los diversos órganos gestores a los que pueda interesar, desde distintas perspectivas, la composición del inventario de un ente y su evolución en el tiempo.

Para ello, en GDI, se realiza la gestión de forma individual de todos los bienes por su inventario, es decir, aquellos que constituyen el inmovilizado, tanto material como inmaterial, del ente, sea cual sea el título que dé lugar a su inclusión en el inventario, así como de los derechos que puedan recaer sobre este tipo de bienes. El sistema se basa en un modelo organizativo que en la versión actual está compuesto por órganos gestores, unidades tramitadoras de tipo expediente, unidades tramitadoras de tipo caja, centros de información de gestión presupuestaria. La versión evolucionada incluirá también el órgano de contratación, las unidades proponentes y al gestor de inventario (Sorolla2, 2012).

Este sistema tiene como requisitos de software el empleo de un ordenador con conexión a la red *SARA*[1] (Sistemas de aplicaciones y redes para las administraciones) o a Internet con navegador de Internet: Internet Explorer 7.0 o superior y Firefox 3.5 o superior. Necesita un certificado electrónico para el acceso al sistema, visor de *Portable Document Format (PDF)* para la visualización de los informes generados y requisitos para la firma electrónica mediante certificado.

Este sistema es una aplicación web, y los requisitos de software que necesita constituyen una limitante para su uso en el país, ya que necesita de la conexión a la red SARA. Esta red implementa importantes medidas de seguridad entre las que destaca el establecimiento de redes virtuales privadas. Es una red extremadamente segura en la que todo el tráfico circula cifrado por la red Troncal. De esta manera queda asegurada la confidencialidad de la información que viaja a través de la red SARA (Sorolla2, 2012). Se requiere además de un certificado electrónico para poder acceder al sistema, lo cual implica un alto costo por concepto de licencia, que la institución no puede pagar.

Sistema Integrado Administrativo Contable (SIAC)

Es una herramienta que integra cada uno de los departamentos de la empresa, llevando el control del manejo de todas las operaciones, desde la compra, ventas movimientos de inventario, cobros, declaraciones de impuestos, entre otros.

SIAC está desarrollado en una plataforma acorde a la tecnología actual, ya que bajo los estándares del ambiente de Windows, utilizando para este fin *Visual Basic* como herramienta de desarrollo, *Crystal Reports* como herramienta para sus reportes y *SQL Server* como gestor de base de datos, que le permite poder manejar grandes cantidades de transacciones de una forma segura, confiable e integrada, por lo que cumple con las necesidades de cualquier empresa comercial.

[1]SARA: conjunto de infraestructuras de comunicaciones y servicios básicos que conecta las redes de las Administraciones Públicas Españolas e Instituciones Europeas facilitando el intercambio de información y el acceso a los servicios.

Este sistema no cumple con las normas para ser utilizada en la UCI ya que usa una red privativa que sólo pueden usar compañías españolas y otras europeas.

Versat-Sarasola

Es el primer sistema integral de gestión de contabilidad certificado, desarrollado para la gestión económica eficaz y fiable. Actualmente es utilizado en Cuba en alrededor de 200 entidades de varias provincias. Este sistema integrado cuenta con un conjunto de 12 módulos entre los que se encuentran: configuración y seguridad, contabilidad general y de gastos, costos y procesos, análisis económico empresarial, control de activos fijos, finanzas y caja, planificación y presupuestos, control de inventarios, pago de salario, paquete de gestión, contratación, facturación (Versat, 2012).

En el módulo de control de inventarios se definen formatos del clasificador de productos para lograr una uniformidad en el registro y la agregación de información en los reportes de salida, se conceptualizan los movimientos para lograr una información amplia sobre los orígenes y destinos de los recursos. Permite el control de las existencias y movimientos en diferentes monedas. Muestra el cuadre diario de cada uno de los almacenes por las diferentes cuentas. Ofrece la posibilidad de duplicar documentos para agilizar los pases de los mismos y lograr que un mismo documento se convierta en otro con solo adicionar un mínimo de información, realiza un control de las existencias y movimientos por custodios y se emiten diferentes reportes e información de utilidad para la correcta administración de los recursos materiales (Versat, 2012).

A pesar de los grandes beneficios que trae consigo la utilización del *Versat*, este software no resulta una solución factible para la UCI, pues fue desarrollada sobre plataformas de software propietario, por lo que no cumplen con la independencia tecnológica que se desea alcanzar en el país. Presenta gran integración entre módulos, y se maneja información mucho más complicada de la que se necesita en estos momentos en la UCI para el control de los AFT. No permite, desde el mismo sistema, la generación de reportes precisos de la empresa, como por ejemplo el acta de responsabilidad material que se generan para cada departamento o área de la empresa, sin necesidad de hacer este reporte manualmente.

Inventario en la UCI (Assets)

El sistema de gestión integral (ASSETS) es un sistema multiusuario que se integra en una plataforma de servidores SQL, dividido en módulos económicos que trabajan en conjunto para el control de las actividades económica, financiera y contable sobre los medios materiales y financieros. El módulo inventario del Assets divide los medios en dos tipos de cuentas: activos fijos y útiles herramientas. Estas últimas son las que se utilizan para realizar las actividades de mantenimiento, talleres, almacenes, así como los equipos de

protección física. Comprende entre otros, herramientas manuales, artículos de protección personal, utensilios de laboratorios, mini calculadoras, utensilios menores de cocina. Los medios que pertenecen a esta cuenta tienen poco tiempo de duración (Assets, 2005).

Entre las desventajas que presenta este sistema, es que controla los medios por áreas de inmuebles y es de necesidad para el centro controlar los medios por los diferentes locales existentes dentro de cada área, y agregarles diferentes características a los medios y locales existentes. Además no brinda toda la información a los usuarios y no cuenta con un catálogo de reportes que muestre parte de su información a otros usuarios para la toma de decisiones.

Conclusiones del estudio de los sistemas informáticos existentes

Luego de haber realizado el estudio de diversos sistemas, tanto nacionales como internacionales, se puede concluir que:

- Los sistemas internacionales tienen un alto nivel de configuración permitiendo cubrir la mayoría de las necesidades de la empresa, pero tienen la desventaja que alguno de ellos están implementados con tecnología que no es posible su acceso desde Cuba ya que algunas son privativas, lo que incrementaría los gastos en licencia y mantenimiento del software. Otra de las desventajas que estos sistemas presentan es que no cumplen con las necesidades ni las características de la economía cubana, pues generalmente se centran en sectores específicos o fueron desarrollados para economías no muy semejantes a la de Cuba.
- Los sistemas implantados en el país necesitan de grandes prestaciones y de integraciones entre módulos lo cual no es aplicable. Se maneja información mucho más complicada de la que en realidad se necesita. No permiten la generación de reportes específicos de gran utilidad para el control de los AFT en las áreas más pequeñas que serían en este caso las áreas de responsabilidad.

Después de haber realizado un estudio de diversos sistemas, tanto nacionales como internacionales, dedicados a esta tarea, cada uno de ellos tiene sus ventajas y desventajas pero finalmente se concluye que estas no son factibles para su aplicación en la UCI, ya que la misma debes estar conectada en redes privativas. Por tanto se decide realizar un software que responda los problemas anteriormente descritos.

1.3 Metodología de desarrollo de software

Desarrollar productos de software con alta calidad depende de las actividades que conllevan a su construcción, donde la selección de la metodología más adecuada juega un papel importante. La correcta selección es fundamental para la planificación, gestión, control y evaluación de forma sistemática, lo que garantiza grandes probabilidades de éxito.

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el objetivo de hacerlo más predecible y eficiente, donde predecir no significa perder la capacidad adaptativa, no significa evitar la introducción de cambios en los requisitos, ni evitar que nuevos requisitos surjan sino definir un camino reproducible para obtener resultados confiables. Definen, además, una representación que permite facilitar la manipulación de modelos, la comunicación e intercambio de información entre todas las partes involucradas en la construcción de un sistema (Figueroa, 2012).

Metodologías ágiles

Las metodologías ágiles están especialmente orientadas para entornos variables, proyectos pequeños donde los individuos y las interacciones entre ellos, son más importantes que las herramientas y los procesos empleados y donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

Proceso unificado ágil (AUP-UCI)

El Proceso Unificado Ágil de *Scott Ambler* o *Agile Unified Process* (AUP) en inglés es una versión simplificada del Proceso Unificado de *Rational* (RUP). Este describe de una manera sencilla de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (*test driven development* - TDD), Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad (UCI, 2014).

El proceso unificado (*Unified Process* o UP) es un marco de desarrollo de software iterativo e incremental. A menudo es considerado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto de software. Dado que es un marco de procesos, puede ser adaptado y la más conocida es RUP (*Rational Unified Process*) de IBM (UCI, 2014).

El proceso AUP establece un Modelo más simple que el que aparece en RUP por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP (UCI, 2014).

Las disciplinas se llevan a cabo de manera sistemática, se definen las actividades que realizan los miembros del equipo de desarrollo a fin de implementar, validar, y entregar el software de trabajo que responda a las necesidades de sus interlocutores (UCI, 2014). Las disciplinas son:

1. Modelo: el objetivo de esta disciplina es entender el negocio de la organización, el problema de dominio que se abordan en el proyecto, y determinar una solución viable para resolver el problema de dominio.
2. Aplicación: el objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y realizar un nivel básico de las pruebas, en particular, la unidad de pruebas.
3. Prueba: el objetivo de esta disciplina consiste en realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.
4. Despliegue: el objetivo de esta disciplina es la prestación y ejecución del sistema y que el mismo este a disposición de los usuarios finales.
5. Gestión de configuración: el objetivo de esta disciplina es la gestión de acceso a herramientas de su proyecto. Esto incluye no sólo el seguimiento de las versiones con el tiempo, sino también el control y gestión del cambio para ellos.
6. Gestión de proyectos: el objetivo de esta disciplina es dirigir las actividades que se lleva a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc.), coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que es entregado a tiempo y dentro del presupuesto.
7. Entorno: el objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que el proceso sea el adecuado, la orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario.

Se selecciona esta metodología, ya que es la que se desarrolla en el centro productivo, además de generar artefactos correspondientes al negocio.

1.4 Herramientas y tecnologías a utilizar

A continuación, se describen las principales herramientas y tecnologías que se utilizarán para el desarrollo de la investigación y la implementación de la solución.

Lenguaje unificado de modelado (UML 2.0)

Lenguaje Unificado de Modelado (UML) ha sido el estándar de la industria para visualizar, especificar, construir y documentar los artefactos de los sistemas software. Como lenguaje de modelado estándar de facto, UML favorece la comunicación y reduce la confusión entre los participantes de un proyecto software. La viabilidad y el ámbito del lenguaje han crecido con la reciente estandarización de UML 2.0. Su facilidad de uso permite a los usuarios modelar todo tipo de sistemas, desde sistemas de información de empresas y aplicaciones Web distribuidas hasta sistemas embebidos de tiempo real (Larman, 2003).

UML es el resultado del trabajo realizado por Grady Booch, James Rumbaugh e Ivar Jacobson. Está constituido por un conjunto de diagramas y permite generar un anteproyecto de la propuesta de solución que permite tanto a los autores como a los clientes una mayor comprensión y claridad acerca de lo que debe hacer el sistema (Jacobson, 2000).

Herramienta de modelado Visual Paradigm 8.0

Para el modelado se utilizó *Visual Paradigm* para UML 8.0 por ser una herramienta CASE *(Computer Aided Software Engineering*, Ingeniería de Software Asistida por Ordenador) que soporta el ciclo de vida completo en el desarrollo de software: análisis y desarrollo orientados a objetos, construcción, prueba y despliegue . Entre las características fundamentales que determinaron su selección para el modelado durante todo el proceso de desarrollo del software están las siguientes (Berberat, 2012):

- Apoya todo lo básico en cuanto a artefactos generados en las etapas de definición de requerimientos y de especificación de componentes.
- Brinda apoyo adicional en cuanto a generación de artefactos automáticamente.
- Genera modelos VP-UML instantáneamente a partir de código binario.
- Tiene disponibilidad en múltiples plataformas.

- Brinda la posibilidad de intercambiar información mediante la importación y exportación de ficheros.
- Permite la generación de código e ingeniería inversa, además de la generación de documentación.

Gestor de bases de datos (SGBD) PostgreSQL 9.4

Esta versión agrega muchas nuevas características que mejoran la flexibilidad, escalabilidad y rendimiento de *PostgreSQL* para diferentes tipos de usuarios de bases de datos, incluyendo mejoras al soporte para JSON, replicación y rendimiento de los índices (PostgreSQL, 2014).

Flexibilidad: con el nuevo tipo de datos JSONB para PostgreSQL, los usuarios ya no tienen que escoger entre almacenes de datos relacionales y no-relacionales: pueden tener los dos al mismo tiempo. JSONB soporta búsquedas rápidas y consultas de búsqueda con expresiones simples usando Generalized Inverted Indexes (GIN).

Escalabilidad: en 9.4, la Decodificación Lógica *(Logical Decoding)* provee una nueva API para leer, filtrar y manipular el flujo de replicación de *PostgreSQL*. Esta interfaz es la base para nuevas herramientas de replicación, como la Replicación Bi-Direccional, la cual soporta la creación de clústeres de *PostgreSQL* multi-maestros. Otras mejoras en el sistema de replicación, como las ranuras de replicación y réplicas temporizadas, mejoran la gestión y utilidad de los servidores réplica.

Rendimiento: la versión 9.4 también introduce varias mejoras de rendimientos que les permitirán a los usuarios sacar aún más provecho de cada servidor *PostgreSQL.* Estas incluyen:

- Mejoras a los índices GIN, haciéndolos hasta 50% más pequeños y hasta 3 veces más rápidos.
- Vistas materializadas actualizables de forma concurrente, para reportes más rápidos y actualizados.
- Recarga rápida del caché de la base de datos en un reinicio usando *pg_prewarm*.
- Escritura paralela más rápida en el log transaccional de *PostgreSQL.*

Entorno de desarrollo integrado (IDE) Netbeans 8.0

Un Entorno de Desarrollo Integrado (del inglés *Integrated Development Environment*, IDE) es un programa compuesto por un conjunto de herramientas para un programador. Provee un marco de trabajo amigable para la mayoría de los lenguajes de programación, pudiendo utilizarse en el mismo uno o varios lenguajes de programación. Para la construcción del sistema se hizo necesario escoger un entorno integrado de desarrollo que proporcionara el uso y la integración de todas las tecnologías y lenguajes mencionados en este capítulo (López, 2014).

Para la implementación de la propuesta de solución se utiliza el IDE Netbeans 8.0, el mismo es un producto libre, gratuito, sin restricciones de uso y con un importante número de módulos para extenderlo. Entre sus características están las administraciones de ventanas, almacenamiento, interfaces y configuraciones de usuario. Además, empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones (López, 2014).

Lenguaje de Programación PHP 5.5

Aunque oficialmente un lanzamiento puntual, PHP 5.5 es en realidad la actualización más significativa al lenguaje desde el lanzamiento de 5.0. generando una serie poderosa de nuevas características incluyendo espacios de nombres, vinculación estática tardía, funciones lambda y cierres, un nuevo controlador de *MySQL*, y una variedad de adiciones sintácticas como la sintaxis de *NOWDOC*, la versión 5.5 representa un gran paso adelante en la evolución de PHP (Heurtel, 2011).

A continuación, se detallan las siguientes características:

- Es un lenguaje multiplataforma.
- Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.
- El código fuente escrito en PHP es invisible al navegador web y al cliente ya que es el servidor el que se encarga de ejecutar el código y enviar su resultado HTML al navegador. Esto hace que la programación en PHP sea segura y confiable.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con *MySQL* y *PostgreSQL*.
- Capacidad de expandir su potencial utilizando módulos (llamados *ext's* o extensiones).
- Posee una amplia documentación en su sitio web oficial, entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite aplicar técnicas de programación orientada a objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.

- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Si bien PHP no obliga a quien lo usa a seguir una determinada metodología a la hora de programar (muchos otros lenguajes tampoco lo hacen), aun haciéndolo, el programador puede aplicar en su trabajo cualquier técnica de programación o de desarrollo que le permita escribir código ordenado, estructurado y manejable. Un ejemplo de esto son los desarrollos que en PHP se han hecho del patrón de diseño Modelo Vista Controlador (MVC), que permiten separar el tratamiento y acceso a los datos, la lógica de control y la interfaz de usuario en tres componentes independientes.
- Como es un lenguaje que se interpreta en ejecución, para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no la impide y, en ciertos casos, representa un costo en tiempos de ejecución.

Lenguaje de Programación JavaScript

JavaScript es uno de los recursos surgidos en la programación para darle dinamismo y capacidad al lenguaje HTML. En la actualidad es una de las tecnologías más extendidas cuando se trata de enriquecer páginas web del lado del cliente. Es un lenguaje interpretado, por lo que no es necesario compilar los programas para ejecutarlos (Gauchat, 2013).

Se hace imprescindible el uso de este lenguaje para la solución propuesta con la finalidad de validar los datos entrados por los usuarios al sistema permitiendo ejecutar instrucciones como respuesta a las acciones del usuario.

Framework de desarrollo Symfony 2.8

Symfony es uno de los marcos de trabajo PHP más populares entre los usuarios y las empresas, pues permite que los programadores sean mucho más productivos a la vez que crean código de más calidad y más fácil de mantener. *Symfony* es maduro, estable, profesional y está muy bien documentado Este framework de desarrollo cuenta con un grupo de características que influyen en su uso generalizado (Potencier, 2014).

- Fácil de instalar y configurar en la mayoría de las plataformas.

- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Fácil de extender permitiendo su integración con las bibliotecas de otros fabricantes.
- Resulta sencillo de usar en la mayoría de los casos permitiendo un mejor desenvolvimiento de los desarrolladores en el proceso de implementación.

1.5 Conclusiones del capítulo

El análisis de la bibliografía y la investigación de cuatros sistemas informáticos para la gestión de bajas técnicas de los AFT, arrojó los elementos suficientes para corroborar la necesidad de implementar un sistema que sea capaz de ajustarse a las necesidades de la UCI, nutriéndose de las ventajas que presentan las aplicaciones analizadas. Teniendo en cuenta el análisis de la bibliografía consultada y en correspondencia con las características de la solución a realizar, se define la utilización de la metodología de desarrollo de software AUP-UCI, que guiará la implementación de la solución con *Visual Paradigm 8.0* como herramienta CASE para modelar los procesos, el IDE Netbeans 8.0, los lenguajes de programación JavaScripts y PHP 5.3, *Symfony 2.8* como marco de trabajo y el sistema gestor de bases de datos *PostgreSQL 9.4*.

Capítulo 2: Características del sistema de gestión de bajas técnicas de los AFT

Introducción

En este capítulo se definirán los procesos que se van a automatizar, se define el modelo de negocio y los diagramas para lograr una mejor comprensión del mismo. Además, se determinan y especifican los requisitos funcionales y no funcionales, se modela el sistema y se identifican los casos de uso con sus respectivos diagramas para conocer las características del sistema. Se especificarán además los patrones utilizados, así como otras características fundamentales como parte del diseño, generándose además un grupo de artefactos teniendo en cuenta la metodología a utilizar.

Propuesta de solución

Se propone realizar un sistema informático que permita gestionar el proceso de bajas técnicas de los AFT que se lleva a cabo en la UCI. Este sistema permitirá asignar recursos a las diferentes áreas, donde se desea realizar el correcto flujo del proceso, de forma tal que dicho sistema pueda darle respuesta a la problemática planteada.

2.1 Modelo de negocio

El proceso de bajas técnicas de los AFT en la UCI, es realizado por un conjunto de trabajadores, donde el jefe de un área inicia el proceso creando y enviando la(s) solicitud(es) de dictamen, siendo recibidas y aprobadas por el jefe de estructura; una vez aprobada la solicitud, el jefe de comisión recibe la solicitud, asignando una comisión para la realización del dictamen referente a la solicitud en cuestión. Luego de realizarse el dictamen por cada medio, la comisión envía los dictámenes al jefe de comisión revisando y exportando los dictámenes a formato *PDF* para firmar y enviar al jefe de estructura, finalizando de esta manera el proceso de bajas técnicas de los AFT en la UCI.

Según la metodología AUP-UCI, se realiza el modelo de casos de uso del negocio, las reglas de negocio, diagrama de actividades, diagramas de clases del diseño y diagramas de componentes para un mejor entendimiento y organización del sistema.

Según Ivar Jacobson un modelo de negocio describe los procesos de negocio de una empresa en términos de casos de uso del negocio y actores del negocio, que se corresponden con los procesos del negocio y los

clientes, respectivamente. Al igual que el modelo de casos de uso para un sistema de software, el modelo de casos de uso del negocio presenta un sistema desde la perspectiva de su uso y esquematiza cómo proporciona valor a sus usuarios (Jacobson, 2000).

Reglas del negocio

En una organización, tanto los procesos como los datos que estos manejan, están restringidos por reglas del negocio. Estas consisten en el conjunto de elementos del negocio que actúan como reguladores de la ejecución, supervisión, control y toma de decisiones de los diferentes procesos y actividades (Abran, 2004). A continuación, se listan las reglas del negocio:

- La solicitud de bajas técnicas de los AFT es realizada por los jefes de áreas.
- El documento a importar tiene que ser en formato Excel.
- Los usuarios solo tendrán acceso en el sistema según los permisos otorgados por el administrador del sistema.
- La solicitud realizada por los jefes de áreas serán aprobadas únicamente por su jefe de estructura.
- La solicitud sólo será visible por el jefe de comisión cuando será aprobada por el jefe de estructura.
- El dictamen será realizado solamente después de que el jefe de comisión asigne una comisión y apruebe la solicitud.
- La comisión realizará el dictamen de los AFT correspondiente a la solicitud asignada por el jefe de comisión.
- La comisión realizará los dictámenes por cada solicitud por separadas, emitiendo los dictámenes de todos los AFT existentes en cada solicitud.
- La solicitud de baja es enviada al jefe de comisión, después de haber sido aprobada por el jefe de comisión.
- El dictamen emitido por la comisión debe estar firmada por el jefe de comisión antes de ser enviada al jefe de estructura en la extensión PDF.
- Los dictámenes a imprimir deben estar en la extensión PDF.

Actores del negocio y trabajadores del negocio

Una vez identificados los procesos del negocio correspondientes a la solicitud de bajas técnicas de los AFT y aprobación de la misma, es posible determinar los actores involucrados en su realización. Los actores constituyen elementos externos que interactúan con los procesos del negocio. A continuación, se muestran los actores que participan en el proceso de solicitud de bajas técnicas de los AFT de la UCI

Tab. 1: Actores del negocio

Actor	Descripción
Jefe de Estructura	Es el encargado de aprobar las solicitudes de dictámenes que realizan los jefes de áreas.

Tab. 2: Trabajadores del negocio

Trabajador	Descripción
Jefe de Comisión	Es el encargado de aprobar las solicitudes de dictámenes que realizan los jefes de estructuras.
Jefe de Área	Es el encargado de realizar las solicitudes de bajas técnicas.
Comisión	Es el encargado de realizar los dictámenes de las solicitudes realizadas por los jefes de estructuras.

Diagrama de caso de uso del negocio (CUN)

Un diagrama de CUN describe los procesos de un negocio vinculados al campo de acción, y cómo se benefician e interactúan los socios y clientes en estos procesos (González, 2013). A continuación, se muestra el diagrama de CUN correspondiente a la propuesta de solución:

CUN "Gestionar solicitud de bajas técnicas de los AFT"

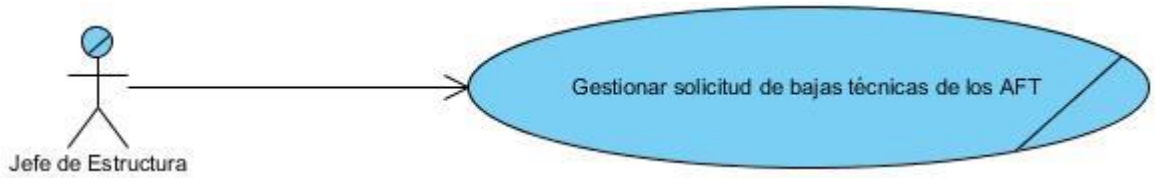


Fig. 1: Diagrama de Caso de Uso del Negocio

Descripción textual del CUN "Gestionar solicitud de bajas técnicas de los AFT"

Tab. 3: Descripción administrar solicitud de dictamen

Caso de Uso	Gestionar solicitudes de bajas técnicas de los AFT.	
Objetivo	Solicitar el dictamen de uno o varios activos fijos.	
Actores	Jefe de Área (inicia)	
Resumen	El CUN se inicia cuando el jefe de área envía la solicitud de dictamen de uno o varios AFT al jefe de estructura. El mismo aprueba la solicitud, siendo esta enviada al jefe de comisión, donde este asigna una comisión para realizar el dictamen a los AFT de la solicitud. La comisión recibe la solicitud y le realiza el dictamen, una vez realizada esta operación se le envía al jefe de comisión. El mismo exporta y firma el(los) dictamen(es), enviando el(los) mismo(s) al jefe de estructura, culminando así el proceso.	
Complejidad	Alta	
Prioridad	Crítico	
Prioridad:	Alta	
Flujo de eventos "Gestionar solicitud de bajas técnicas de los AFT"		
Flujo básico < Gestionar las bajas técnicas de los AFT >		
No.	**Actor**	**Negocio**
1	El jefe de área realiza la solicitud y es enviada al jefe de estructura.	
2		El jefe de estructura recibe la solicitud de dictamen realizada por el jefe de área.
3	El jefe estructura revisa la solicitud y la envía al jefe de comisión.	
4		El jefe de comisión recibe la solicitud de dictamen.
5		El jefe de comisión revisa que la solicitud sea correcta, en caso de ser incorrecta ver el curso alterno 5.

6	El jefe de área recibe la solicitud de dictamen rechazada.	
7		El jefe de comisión envía la solicitud de dictamen a la comisión.
8		La comisión recibe la solicitud y verifica el listado de AFT de la solicitud.
9		La comisión emite un dictamen técnico y lo entrega al jefe de la comisión.
10		El jefe de la comisión recibe el dictamen y lo revisa en caso de que sea correcto exporta el dictamen a PDF firmando el mismo y enviándolo al jefe de estructura, en caso contrario ver el curso alterno 10.
11	El jefe de estructura recibe el dictamen y lo firma.	
Cursos alternos		
5	El jefe de comisión rechaza la solicitud de dictamen y recomienda corregir los datos necesarios de la solicitud.	
10	El jefe de la comisión envía el dictamen a la comisión para que vuelvan a emitir correctamente el dictamen.	
Casos de Uso Incluidos:	No existe.	

Diagrama de actividades

Un diagrama de actividades describe un proceso que explora el orden de las tareas o actividades que logran los objetivos del negocio (Larman, 2003). En el diagrama se muestran actividades de color amarillo, lo que significa que se consideran automatizables. A continuación, se muestra el diagrama de actividades correspondiente al CUN: Gestionar solicitud de bajas técnicas de los AFT.

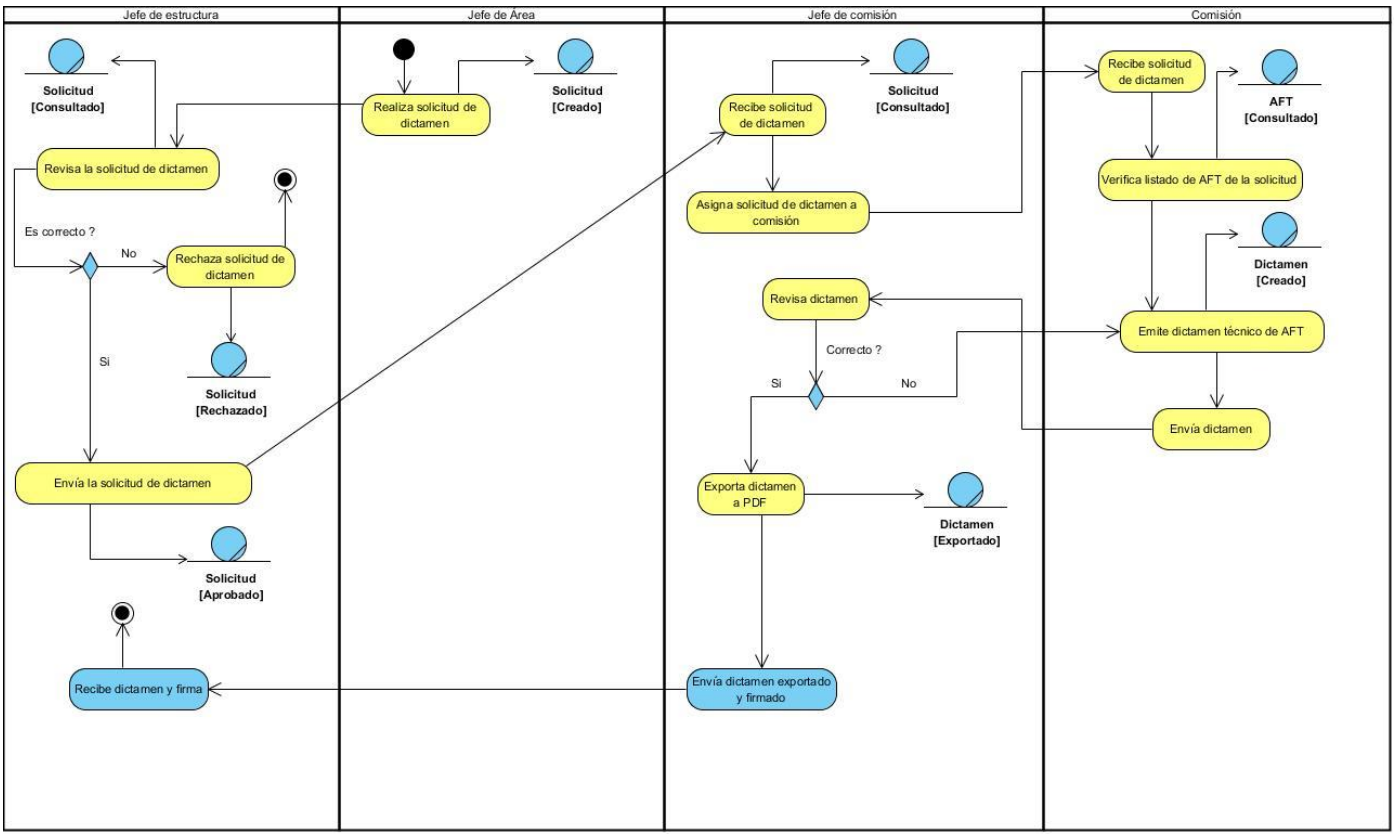


Fig. 2: Diagrama de actividades

Modelo de objetos del negocio

El modelo de objetos del negocio es un modelo interno a un negocio. Describe cómo cada caso de uso del negocio, es llevado a cabo por parte de un conjunto de trabajadores que utilizan un conjunto de entidades del negocio y unidades de trabajo **(EUMED, 2012)**. En este diagrama se evidencia la relación que existe entre los trabajadores del sistema con las entidades del negocio. A continuación, se muestra el modelo de objetos propuesto:

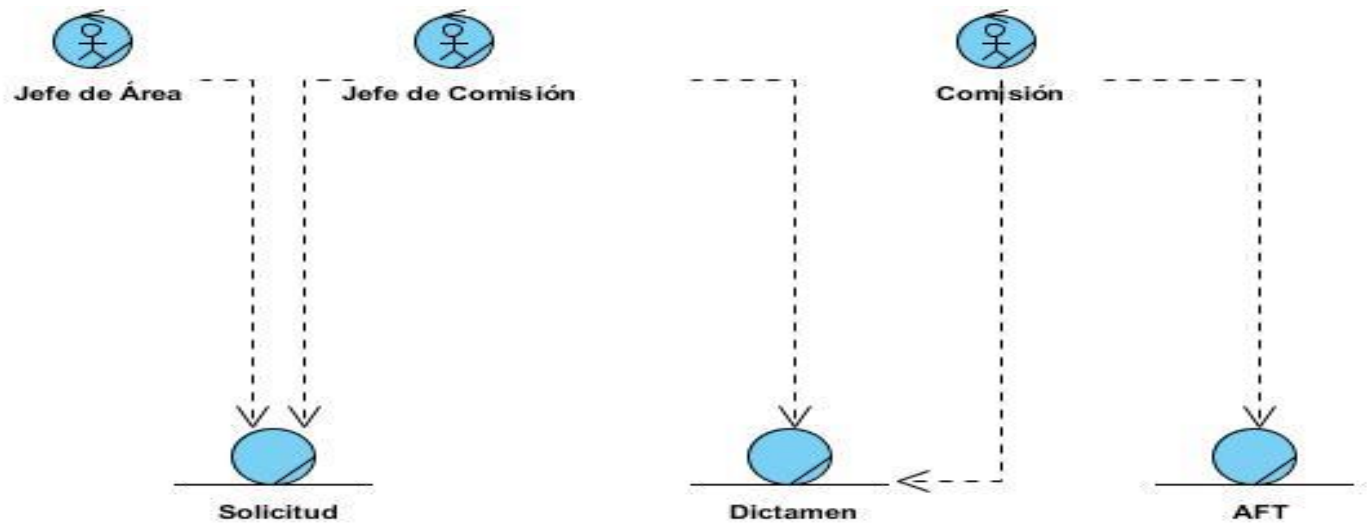


Fig. 3: Modelo de objeto del negocio

2.2 Levantamiento de los requisitos

Según Sommerville, durante el levantamiento u obtención de requisitos, los desarrolladores del software trabajan directamente con los clientes y los usuarios finales del sistema, para determinar el dominio de la aplicación, los servicios que debe proporcionar el sistema, el rendimiento requerido, las restricciones de hardware, entre otros elementos, que de conjunto garantizarán la entrega de un producto final que satisfaga las necesidades planteadas por el cliente (Sommerville, 2005).

Por su parte Pressman plantea que los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones en las que debe operar (Pressman, 2008). En dependencia de sus características, los requisitos de software se dividen en funcionales y no funcionales, lo cual se detalla a continuación.

Listado de Requisitos Funcionales

Tab. 4: Listado de Requisitos Funcionales

CU 1: Autenticar usuario	
RF 1: Autenticar Usuario	Acción para comprobar autenticidad y controlar el acceso de los usuarios a los modelos, informaciones y funcionalidades del sistema

CU 2: Gestionar usuario	
RF 2: Editar usuario	Acción donde se modifican permisos y visibilidad de las funcionalidades a los que puede acceder el usuario
RF 3: Eliminar Usuario	Acción donde se elimina un usuario como tupla de la tabla de una base de datos, mediante la eliminación
RF 4: Listar usuarios	Acción donde se hace un listado o relación de datos de distintos elementos de tipo usuario
RF 5: Buscar usuario	Acción para encontrar o hallar un elemento que coincida con el criterio introducido por el usuario, dentro de una estructura de datos
RF 6: Adicionar usuario	Acción donde se adiciona un usuario como tupla de la tabla de una base de datos
CU 3: Gestionar rol	
RF 7: Adicionar rol	Acción donde se adiciona un rol para asignarle permisos a un usuario
RF 8: Modificar rol	Acción donde se edita el rol asignado a un usuario
RF 9: Buscar rol	Acción para encontrar los usuarios con un rol específico
RF 10: Eliminar rol	Acción se desea eliminar el rol asignado a un usuario
CU 4: Gestionar estructura	
RF 11: Adicionar estructura	Acción donde se crea una estructura
RF 12: Modificar estructura	Acción donde se modifica una estructura
RF 13: Buscar estructura	Acción para la búsqueda de una estructura
RF 14: Eliminar estructura	Acción donde se elimina una estructura ya existente
CU 5: Gestionar centro de costo	

RF 15: Adicionar centro de costo	Acción donde se adiciona un centro de costo
RF 17: Modificar centro de costo	Acción donde se modifica un centro de costo
RF 18: Buscar centro de costo	Acción para la búsqueda de un centro de costo
RF 19: Eliminar centro de costo	Acción donde se elimina un centro de costo
CU 6: Gestionar área de responsabilidad	
RF 20: Adicionar área de responsabilidad	Acción donde se crea un área de responsabilidad
RF 21: Modificar área de responsabilidad	Acción donde se modifica el área de responsabilidad
RF 22: Buscar área de responsabilidad	Acción para encontrar o hallar un área que coincida con el criterio introducido por el usuario dentro de una estructura de datos
RF 23: Eliminar área de responsabilidad	Acción donde se elimina un área de responsabilidad
CU 7: Gestionar comisión de baja	
RF 24: Adicionar comisión de baja	Acción donde se adiciona una comisión de baja
RF 25: Modificar comisión de baja	Acción donde se modifica una comisión de baja
RF 26: Buscar comisión de baja	Acción para la búsqueda de una comisión de baja
RF 27: Listar comisión de baja	Acción donde se listan todas las comisiones de bajas
RF 28: Eliminar comisión de baja	Acción donde se elimina una comisión de baja
CU 8: Gestionar miembros a comisión	
RF 29: Adicionar miembro	Acción donde se adiciona un miembro
RF 30: Modificar miembro	Acción donde se modifican los datos de un miembro
RF 31: Listar miembro	Acción donde se listan los miembros
RF 32: Buscar miembro	Acción para la búsqueda de miembro correspondiente al criterio de búsqueda del usuario
RF 33: Eliminar miembro	Acción donde se elimina a un miembro
CU 9: Exportar reportes	

RF 34: Exportar reportes	Acción que exportar un modelo creando un documento en formato pdf, que la misma aplicación no podrá editar luego de ser guardado por el usuario
CU 10: Gestionar AFT	
RF 35: Adicionar AFT	Acción donde se adiciona un AFT
RF 36: Modificar AFT	Acción donde se modifica un AFT
RF 37: Buscar AFT	Acción donde se realiza la búsqueda de un AFT correspondiente al criterio de búsqueda del usuario
RF 38: Listar AFT	Acción donde se listan los AFT existentes dentro de un área
RF 39: Eliminar AFT	Acción donde se elimina un AFT
RF 40: Importar AFT	Acción donde se importa desde un documento con formato xlxs u ods
RF 41: Buscar AFT de baja	Acción donde se realiza la búsqueda de un AFT dado de baja correspondiente al criterio de búsqueda por el usuario
RF 42: Listar AFT de baja	Acción donde se listan los AFT que han sido dado de baja
CU 11: Administrar solicitud de dictamen	
RF 43: Adicionar solicitud	Acción donde se adiciona una solicitud de dictamen
RF 44: Buscar solicitud	Acción donde se realiza la búsqueda de una solicitud de dictamen a través de un criterio de búsqueda por el usuario
RF 45: Visualizar solicitud	Acción que permite visualizar los detalles de una solicitud de dictamen
RF 46: Eliminar solicitud	Acción que permite eliminar una solicitud de dictamen
CU 12: Administrar solicitud por área	

RF 47 : Listar solicitud	Acción que permite listar las solicitudes de dictamen por área
RF 48: Buscar solicitud	Acción que realiza la búsqueda de una solicitud de dictamen por área correspondiente al criterio de búsqueda por el usuario
RF 49: Aprobar solicitud	Acción que permite aprobar las solicitudes de dictamen por área
RF 50: Visualizar solicitud	Acción que permite visualizar los detalles de una solicitud de dictamen por área
CU 13: Administrar solicitudes jefe de comisión	
RF 51: Listar solicitud	Acción que permite listar las solicitudes de dictamen
RF 52: Buscar solicitud	Acción que realiza la búsqueda de una solicitud de dictamen correspondiente al criterio de búsqueda por el usuario
RF 53: Aprobar solicitud	Acción que permite aprobar las solicitudes de dictamen
RF 54: Rechazar solicitud	Acción que permite rechazar las solicitudes de dictamen
RF 55: Asignar solicitud de dictamen	Acción donde se asigna una comisión para realizar el dictamen de la solicitud
CU 14: Administrar dictamen	
RF 56: Adicionar dictamen	Acción donde se adiciona un dictamen
RF 57: Listar dictamen	Acción donde se listan todos los dictámenes
RF 58: Buscar dictamen	Acción donde se realiza la búsqueda de un dictamen correspondiente al criterio de búsqueda por el usuario
RF 59: Enviar notificación por correo	Acción que permite realizar el envío de notificaciones a través del correo

Requisitos no Funcionales

Se entiende como Requisitos No Funcionales (RNF) al conjunto de propiedades o cualidades que el producto debe poseer. Luego de analizado el entorno de desarrollo y teniendo en cuenta el futuro despliegue de la solución propuesta. Para un correcto funcionamiento el sistema debe tener como características principales para su funcionamiento los siguientes requisitos no funcionales:

Usabilidad

RnF1: El sistema puede ser utilizado por cualquier persona que tenga al menos conocimientos básicos en el manejo de la computadora, navegación y exploración de los sitios Web en sentido general, las operaciones se realizan con bajo nivel de complejidad.

Requisitos de fiabilidad

RnF2: La información manejada por el sistema está protegida de acceso no autorizado y divulgación. El sistema garantiza la gestión de roles y su asignación a los usuarios, lo cual permite que cada usuario tenga privilegios establecidos de acuerdo a su rol.

Requisitos de eficiencia

RnF3: El tiempo de respuesta promedio de las peticiones realizadas al sistema no excede los 8 segundos con 150 usuarios conectados simultáneamente.

Software en la parte del cliente

RnF4: El sistema está desarrollado sobre tecnologías *web*, lo cual permite que sea multiplataforma y solamente se necesite un navegador web para acceder al mismo. Se puede utilizar con los sistemas operativos Linux o Windows además de navegar con el Mozilla (versión 8 o superior).

Software en la parte del servidor

RnF5: Los servidores del sistema pueden tener como Sistema Operativo Linux 12.04 y Windows 8, utilizando el servidor web Apache, con PHP 5.5 configurado y con PostgreSQL 9.4 para el manejo de la base de datos.

Hardware en la parte del cliente

RnF6: Las estaciones de trabajo del cliente deben contar con al menos 1 GB de memoria RAM, un procesador de 1.3 GHz como mínimo.

Hardware en la parte del servidor

RnF7: El servidor del sistema debe tener como mínimo 2 GB de memoria RAM, un procesador de al menos 2.0 GHz, una tarjeta de red con velocidad de transición de 100 Mbps o superior y en el caso del servidor de la base de datos.

2.3 Modelo de casos de uso del sistema

Jacobson plantea que el modelo de casos de uso del sistema está conformado por los actores y casos de uso que interactúan en el sistema y que describe las funcionalidades que este tendrá, lo cual posibilita al cliente, a los usuarios y a los desarrolladores llegar a un acuerdo sobre cómo utilizarlo (Jacobson, 2000). A través de este modelo, se logra obtener un mejor entendimiento y organización, ya que los actores y trabajadores del sistema estarán relacionado con los casos de uso correspondientes.

Casos de uso del sistema (CUS)

El modelo de casos de uso permite describir los RF del sistema, dando lugar a un acuerdo entre el cliente y los desarrolladores de la aplicación. Proporciona una explicación clara y consistente de lo que debería hacer el software de modo que el modelo se use a lo largo del proceso de desarrollo. Posibilita que se obtenga una base para realizar verificaciones del producto informático, siendo la entrada principal para el análisis, el diseño y las pruebas (Larman, 2003).

Casos de uso

Los casos de uso son descripciones funcionales del sistema que permiten definir los límites del software y las relaciones entre la aplicación y el entorno; describen como los actores pueden usar un sistema. Especifican una secuencia de acciones que debe devolver algún resultado de valor a un actor (González, 2013).

2.3 Actores del sistema

Los actores constituyen cada uno de los roles que agrupan a los diversos tipos de usuarios que interactúan con los diferentes CUS. A partir del análisis de las características que debe tener la solución propuesta se definen para la misma los siguientes actores:

Tab. 5: Actores del sistema

Actores	Descripción

Administrador	Es el encargado de administrar los usuarios, gestionar roles, estructura, centro de costos y área de responsabilidad.
Jefe de Estructura	Es el encargado de administrar las solicitudes de dictámenes por área.
Jefe de Comisión	Es el encargado de gestionar la comisión, directivo y administrar las solicitudes de dictámenes de la comisión.
Comisión	Es la encargada de gestionar los dictámenes.
Jefe de Área	Es el encargado de administrar las solicitudes de bajas y de gestionar los AFT.
Usuario	Es el encargado de exportar los reportes.

Diagrama de Casos de Uso del Sistema

Los diagramas de CUS describen parte del modelo de casos de uso y muestran un conjunto de casos de uso y actores con una asociación entre cada par de estos que interactúan en el sistema (Jacobson, 2000). A esto se puede agregar que constituye una excelente representación del contexto del sistema, ya que sirve como herramienta de comunicación para visualizar, especificar, resumir y documentar el comportamiento de este y sus actores.

A continuación, se presenta el diagrama de CUS correspondiente a la solución propuesta, en el cual se representan los 15 casos de uso que agrupan a los 59 RF identificados.

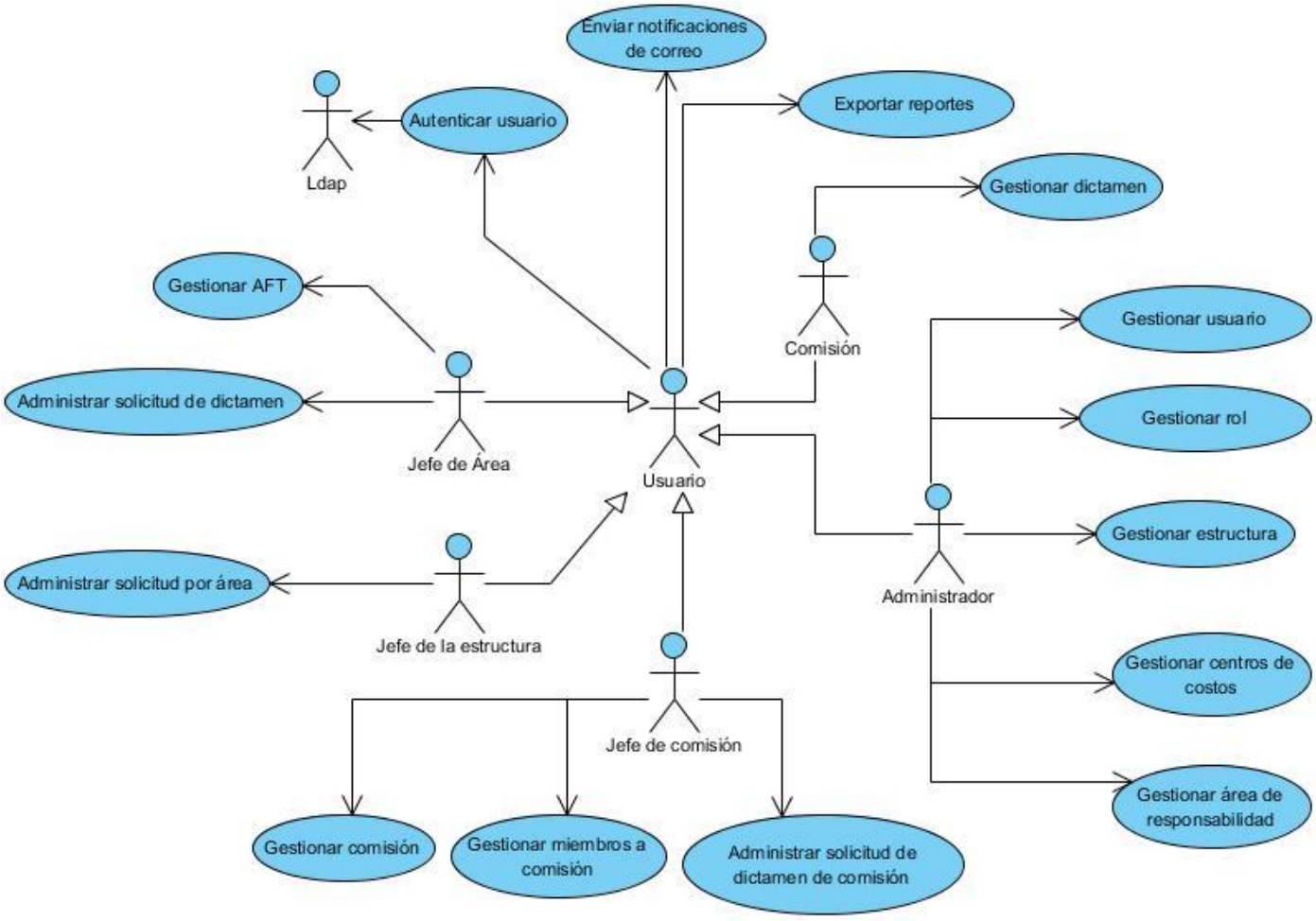


Fig. 4: Diagrama de "Casos de Uso del Sistema"

Descripción textual del CUS Administrar solicitud de dictamen

La descripción textual o especificación de los CU tiene como objetivo describir en detalle el flujo normal de eventos así como los flujos alternos, de una manera precisa y fácilmente comprensible. Además, se incluyen las precondiciones y poscondiciones y se define cómo comienza, termina e interactúa cada CU con sus actores.

A continuación se presenta la descripción textual del CUS Administrar solicitud de dictamen:

Tab. 6: Descripción CUS "Administrar solicitud de dictamen"

Caso de Uso	Administrar solicitud de dictamen

Objetivo	Este CU se desarrolla con el objetivo de realizar las solicitudes de dictamen, para que posteriormente sean aprobadas por el Jefe de estructura a la que pertenece el mismo.
Actores	Jefe de área (inicia)
Resumen:	El CU se inicia cuando el jefe de área selecciona la opción de crear una nueva solicitud y culmina con la realización de una de las siguientes operaciones sobre una solicitud: Adicionar, listar, buscar o visualizar.
Complejidad	Alta
Prioridad	Crítico
Precondiciones	El jefe de área debe estar autenticado en el sistema y debe tener privilegios para realizar las acciones correspondientes. En caso de que se requiera Mostrar alguna solicitud, que anteriormente haya sido creada, se debe seleccionar de la lista de solicitudes y posteriormente seleccionar la operación.
Poscondiciones	Al finalizar el CU la solicitud se crea, se lista, se visualiza o se muestra, en dependencia de la opción seleccionada por el jefe de área.

Flujo de eventos		
Flujo básico <Administrar solicitud>		
No.	**Actor**	**Sistema**
1	Selecciona la pestaña “Solicitudes” y escoge la operación a realizar.	
2		Muestra una lista de solicitudes previamente creadas, que puede estar vacía o con solicitudes ya creadas anteriormente.
3	Selecciona la acción de adicionar una nueva solicitud o a partir de la lista escoge, listar, visualizar o mostrar una solicitud.	
4		Si el usuario escoge la opción “Nueva solicitud”, ver la sección “Nueva solicitud.

Sección 1: "Nueva solicitud"		
Flujo básico <Nueva solicitud>		
No.	**Actor**	**Sistema**
1		Muestra una lista de los AFT existentes en el sistema por el área.
2	Selecciona por el chekBox los AFT que se desea adicionar a la solicitud.	
3		Al ser seleccionado algún AFT, se activa el botón "Crear" y al escoger esta opción se crea la solicitud.
4		El sistema adiciona una nueva solicitud, le asigna el estado "Pendiente Estructura" y envía una notificación al Jefe de estructura a la que pertenece el usuario. Esta acción da paso a la ejecución del CU Administrar solicitud por área, donde es aprobada o rechazada.
Sección 2: "Buscar solicitud"		
Flujo básico <Buscar solicitud>		
No.	**Actor**	**Sistema**
1	El usuario inserta el criterio de búsqueda correspondiente a la solicitud.	
2		El sistema muestra una lista de las solicitudes relacionada con el criterio de búsqueda insertada por el usuario.
Sección 3: "Visualizar solicitud"		
Flujo básico <Visualizar solicitud>		
No.	**Actor**	**Sistema**

1		Muestra una interfaz con la lista de solicitudes y con la opción de visualizar la solicitud seleccionada.
2	Selecciona la opción del botón ver solicitud.	
3		Muestra una interfaz con una lista de los AFT que contenga esa solicitud.
Sección 4: "Eliminar solicitud"		
Flujo básico <Eliminar solicitud>		
No.	**Actor**	**Sistema**
1	El usuario selecciona el botón eliminar correspondiente a la solicitud.	
2		El sistema muestra un mensaje donde la solicitud se ha eliminado de satisfactoriamente.

Relaciones	CU Incluidos	No procede
	CU Extendidos	No procede
Requisitos no funcionales	No procede	
Asuntos pendientes	No procede	

Patrones de casos de uso utilizados

Durante la construcción del diagrama de CUS se utilizaron los patrones de CU que se detallan a continuación:

- **CRUD (Creating, Reading, Updating, Deleting) Completo**

Este patrón consta de un CU, llamado Información CRUD o Gestionar información, modela todas las operaciones que pueden ser realizadas sobre una parte de la información de un tipo específico, tales como creación, lectura, actualización y eliminación. Suele ser utilizado cuando todos los flujos contribuyen al mismo valor del negocio, y estos a su vez son cortos y simples (Larman, 2003). Este patrón se pone de manifiesto en los CU Gestionar usuario, Gestionar rol, Gestionar estructura, Gestionar comisión, Gestionar área de responsabilidad, Gestionar centro de costos y Gestionar dictamen de solicitud.

- **CRUD Parcial**

Se considera una versión del CRUD Completo y su principal diferencia con este radica en que una o varias de las operaciones (creación, lectura, actualización y eliminación) no están presentes en el CU (Larman, 2003). Este patrón se ve evidenciado en los casos de uso “Administrar solicitudes dictamen”, “Administrar solicitudes por área”, “Administrar solicitudes por jefe de comisión” y “Administrar dictamen”.

- **Generalización / Especialización entre actores**

Permite agrupar varios actores que comparten similares roles con respecto a un CU determinado. En la generalización de actores existe una clase padre de actor y una clase hija de actor que es una especialización de la clase padre. En estos casos la hija hereda todas las características de la clase padre y además puede agregar nuevas características (Larman, 2003). Se pone de manifiesto cuando todos los trabajadores del sistema heredan del usuario.

2.4 Diseño del sistema

El diseño del sistema permite determinar cómo funcionará el producto final de forma general sin entrar en detalles incorporando consideraciones de la implementación tecnológica. Consiste en el diseño de los componentes del sistema que dan respuesta a las funcionalidades y requisitos descritos en la etapa de especificación. Generalmente se realiza en base de diagramas que permitan describir las interacciones entre las entidades y su secuenciado, impone una estructura del sistema (Jacobson, 2000).

Estilo arquitectónico

Un estilo arquitectónico es una lista de tipos de componentes que describen los patrones o las interacciones a través de ellos. Un estilo afecta a toda la arquitectura de software y puede combinarse en la propuesta de solución. Los estilos ayudan a un tratamiento estructural que concierne más bien a la teoría, la investigación académica y la arquitectura en el nivel de abstracción más elevado, expresando la arquitectura en un sentido más formal y teórico (Almeira, 2007).

Para el desarrollo de la propuesta de solución se decide el uso del estilo de Llamada y Retorno, en el cual el sistema se constituye de un programa principal que lo controla y varios subprogramas que se comunican con él. Según Pressman el empleo de este estilo posibilita además la comunicación, la coordinación y cooperación entre los componentes y las restricciones que definen cómo se integran para conformar el sistema, así como los modelos semánticos que facilitan al diseñador el entendimiento de todas las partes

del sistema, evitando que las variaciones realizadas a funcionalidades o componentes específicos afecten el funcionamiento general (Pressman, 2008).

Arquitectura y Patrones de Diseño

Según Sommerville, el diseño arquitectónico o arquitectura del software es la primera etapa en el proceso de diseño y representa un enlace crítico entre los procesos de ingeniería de diseño y de requerimientos. Está relacionado con el establecimiento de un marco estructural básico que identifica los principales componentes de un sistema y las comunicaciones entre estos componentes (Sommerville, 2005).

Patrón arquitectónico Modelo – Vista – Controlador (MVC)

Un patrón arquitectónico brinda la descripción de un problema particular y recurrente de diseño, que aparece en contextos de diseño específico, y presenta un esquema genérico demostrado con éxito para su solución. Para el desarrollo de la propuesta de solución se define el patrón arquitectónico MVC, el cual se basa en las ideas de reutilización de código y la separación de conceptos, buscando facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento. Su principal característica radica en que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos, el modelo, la vista y el controlador, los cuales se detallan a continuación:

- El modelo es la representación de los datos en nuestro sistema, gracias a esto se evita mezclar los accesos a bases de datos con el código que implementa la lógica de negocio de las aplicaciones. Todo lo relativo al acceso, modificación, persistencia de datos se encuentra en el modelo, que actúa como capa de abstracción entre el usuario y los datos. Cuando se desea acceder a los datos, no se realiza un acceso de manera directa desde un controlador, sino que se delega la acción al modelo.
- La vista es la capa de presentación de los datos de la aplicación. Se abstrae por completo de la lógica de negocio y del acceso a datos, y se limita a representar de manera gráfica los datos y todos los elementos que va a visualizar el usuario de la aplicación.
- El controlador es el núcleo de la aplicación, donde se implementa la lógica de negocio. Por una parte, responde a las peticiones que hace el usuario, y por otra es el encargado de realizar las solicitudes de datos al modelo. Es un intermediario entre el modelo de datos y las vistas para los usuarios.

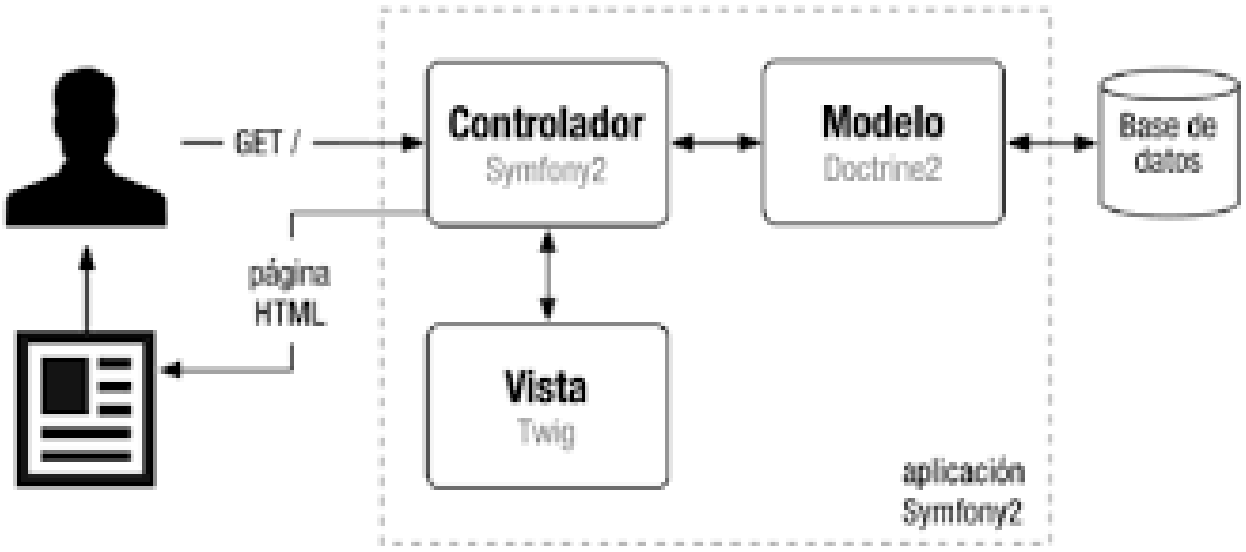


Fig. 5: Patrón arquitectónico MVC

Patrones de diseño

Los patrones de diseño constituyen una descripción de un problema y la solución, a la que se da un nombre y que se puede aplicar a nuevos contextos; idealmente, proporcionan consejos sobre el modo de aplicarlos en varias circunstancias, y consideran los puntos fuertes y compromisos. Muchos patrones proporcionan guías sobre el modo en el que deberían asignarse las responsabilidades a los objetos, dada una categoría específica del problema (Larman, 2003).

Patrones de Principios Generales para Asignar Responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones. A continuación, se explica cómo se evidencia el uso de estos patrones en la propuesta de solución.

Experto: propone asignar las responsabilidades a las clases de acuerdo a la información que contienen las mismas cumpliendo así un principio básico e intuitivo de la programación orientada a objetos. Se utiliza con frecuencia en la asignación de responsabilidades; es un principio de guía básico que se utiliza continuamente en el diseño de objetos (Larman, 2003). Su utilización se pone de manifiesto ya que Symfony utiliza el mapeo objeto-relacional (*ORM* por sus siglas en inglés) Doctrine, para la capa del modelo. Doctrine se encarga de crear una clase experta por cada tabla de la base de datos del modelo.

Controlador: sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, manejando los eventos de entrada de dicha interfaz. Es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema, sino que define el método para la operación del mismo (Larman, 2003). Este patrón está presente en la propuesta de solución pues se crearon varios controladores como: "AFTController", "SolicitudDictamenController", "DictamenController" entre otros que se encargan de manejar eventos entre la vista y el modelo.

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación (Larman, 2003). El uso de este patrón se evidencia en algunas clases como: "SolicitudDictamen.php", "AFT.php", "Dictamen.php" en la capa del modelo, que se encargan de crear instancias de las clases que proveen la información necesaria para su propio manejo.

Bajo acoplamiento: propone la asignación de responsabilidades de manera tal que la dependencia entre una clase y otra sea la menor posible, de tal forma que se potencie la reutilización y se mitiguen los efectos que puedan producir en una, la realización de cambios en la otra. Para esto soporta el diseño de clases que son más independientes, lo que reduce el impacto del cambio (Larman, 2003). Se pone de manifiesto en la solución propuesta pues entre las clases del controlador y la vista o el controlador y el modelo, por ejemplo la clase "SolicitudDictamenController" estará relacionada solamente con la página cliente correspondiente a ella, por lo que puede hacer modificaciones en ella sin afectar a otras.

Alta cohesión: se basa en que los elementos de un componente colaboran para producir algún comportamiento bien definido, como una clase que tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas (Larman, 2003). Se pone de manifiesto en la clase controladora "SolicitudDictamenController" que está relacionada con la clase del modelo "SolicitudDictamen" que tiene una relación de composición con las clases "AFT" y "SolicitudDictamenAFT".

Patrones de la Banda de los Cuatro (GoF)

Los patrones GoF (por sus siglas en inglés de *The Gang of Four*), se utilizan para diseñar objetos y solucionar problemas de creación de instancias, ya que ayudan a encapsular y abstraer dicha creación. A continuación, se explica cómo se manifiestan estos patrones en la propuesta de solución:

Decorador: es un patrón de tipo estructural que permite añadir dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender funcionalidades (Larman, 2003). Este patrón está presente en el diseño de la propuesta de solución, pues en la misma se utilizan

plantillas twig para el desarrollo de las vistas. Estas plantillas heredan de la plantilla “base.html.twig”, lo cual permite declarar regiones o bloques editables dentro de los cuales se realizan las modificaciones particulares de cada página, manteniendo una apariencia homogénea entre todas.

Agente remoto: este patrón se pone de manifiesto cuando el sistema requiere comunicarse con un servicio externo y no se desea o no es posible acceder a este directamente (Larman, 2003). La utilización de este patrón en el sistema se evidencia en la autenticación mediante el servicio “Ldap” disponible para la UCI, el cual provee los datos de los usuarios necesarios para desarrollar este proceso de manera segura. Para evitar acceder directamente a estos datos se creó la clase “Ldap.php”, que es utilizada como mediadora por la clase “LoginController”, para obtener los datos, comprobarlos y proceder a la autenticación.

2.5 Modelo de diseño

El modelo de diseño es un refinamiento y formalización adicional del modelo del análisis, donde se toman en cuenta las consecuencias del ambiente de implementación. Describe la realización de los casos de uso y al mismo tiempo constituye una abstracción del modelo de implementación y del código fuente. El resultado del modelo de diseño son especificaciones muy detalladas de todos los objetos incluyendo sus operaciones y atributos (Jacobson, 2000).

Diagrama de clases de diseño

Un Diagrama de Clases de Diseño (DCD) representa los detalles de las especificaciones de las clases e interfaces software en una aplicación. A diferencia de las clases conceptuales del modelo del dominio, las clases de diseño de los DCD muestran las definiciones de las clases del software en lugar de los conceptos del mundo real (Larman, 2003).

A continuación, se muestra la figura del diagrama de clases del diseño del caso de uso Administrar solicitud de dictamen:

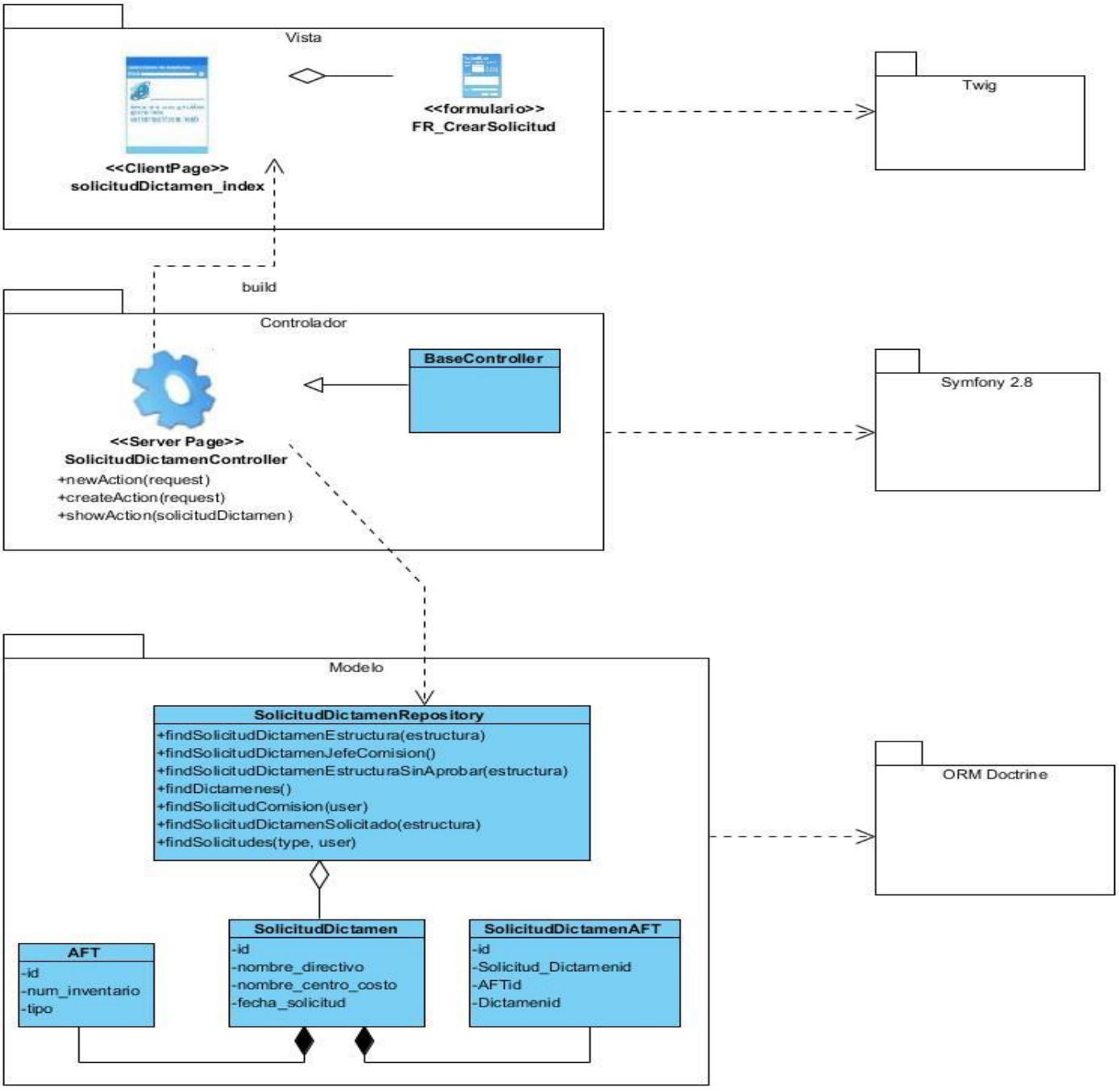


Fig. 6: Diagrama de clases del diseño "Administrar solicitud de dictamen"

Diagrama entidad-relación

El Diagrama Entidad Relación (E-R) proporciona una herramienta para representar información del mundo real a nivel conceptual. Permite describir las entidades involucradas en una base de datos, así como las

relaciones y restricciones de ellas (Amparo López Gaona, 2012). En la figura 6 se muestra el diagrama E-R correspondiente al modelo de datos de la propuesta de solución, donde se agrupan las entidades que lo conforman y las relaciones entre estas.

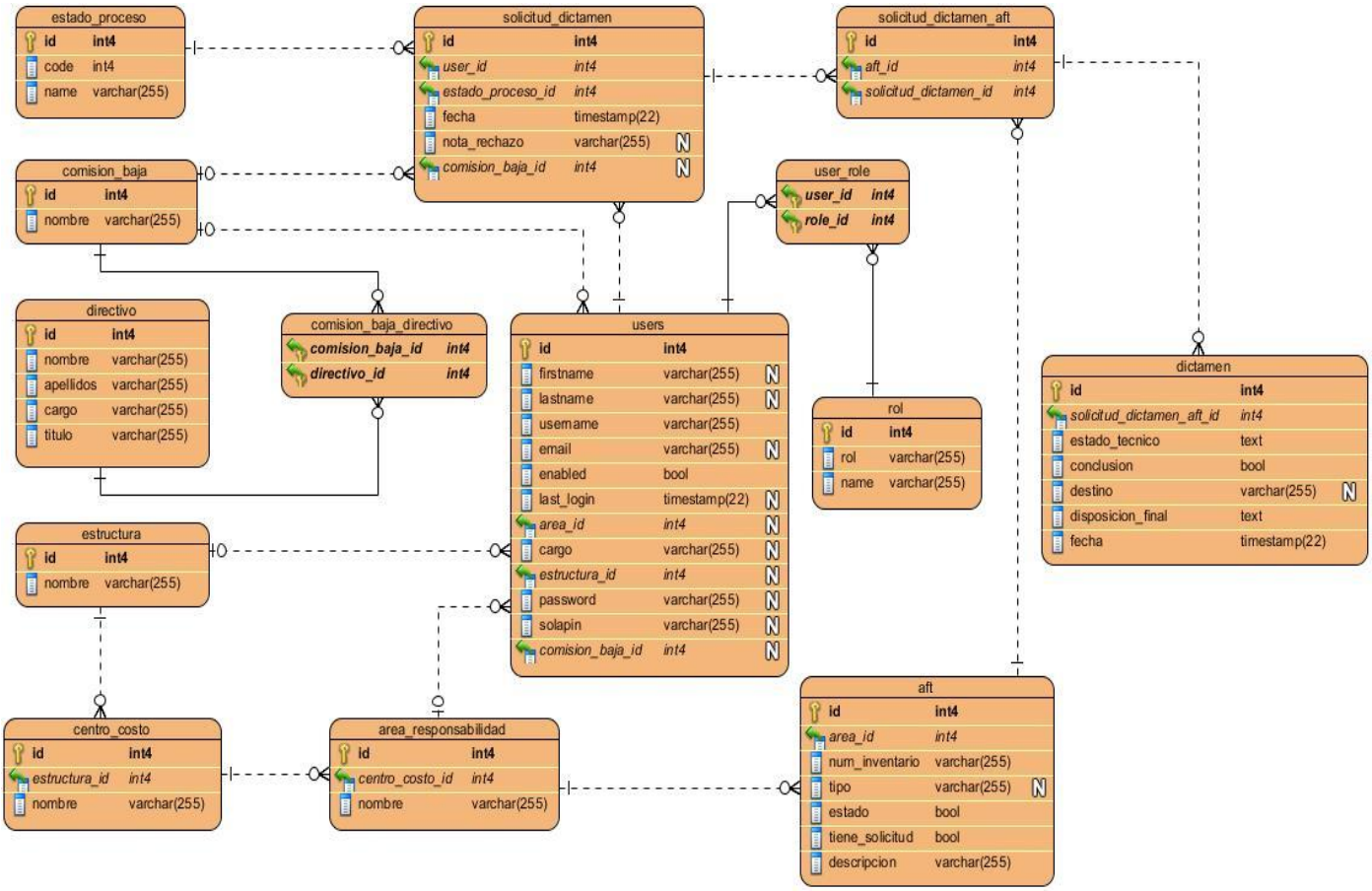


Fig. 7: Modelo Entidad-Relación

2.6 Conclusiones del capítulo

En el presente capítulo se realizó el análisis del negocio en cuestión a partir del cual se pudo identificar con el cliente 59 requisitos funcionales, agrupados en 15 casos de uso, utilizándose los patrones CRUD tanto parciales como totales y especialización entre actores , y 7 requisitos no funcionales. Una vez realizado el diseño de la propuesta de solución, se obtuvo el diagrama E-R del modelo de datos y los diagramas de clases del diseño, aplicándose los patrones de diseños GRASP y GoF: creador, experto, alta cohesión, bajo acoplamiento y controlador, decorador y agente remoto respectivamente.

Capítulo 3: Implementación y prueba del sistema de gestión de bajas técnicas de los AFT

Introducción

En este capítulo se describe el proceso de implementación y pruebas de la solución. Se especifican la selección, diseño y aplicación de las pruebas realizadas en correspondencia a la tipología de la solución. Finalmente se muestran los resultados obtenidos una vez concluida la ejecución de los ensayos de validación.

3.1 Modelo de implementación

El modelo de implementación describe cómo los elementos de diseño, como las clases, se implementan en términos de componentes, como ficheros de código fuente y ejecutables. Además describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en los lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (Jacobson, 2000).

Diagrama de componentes

Los diagramas de componentes describen los elementos físicos y sus relaciones en el entorno. Los componentes representan los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas, y las relaciones de dependencia se utilizan para indicar que un componente utiliza los servicios ofrecidos por otro componente. Además agrupan varias partes de un sistema modular, desplegables y reemplazables, que encapsulan la implementación y expone un conjunto de interfaces, como por ejemplo, código fuente, binario o ejecutable (Larman, 2003).

A continuación, se muestra el diagrama de componentes que describe el caso de uso Administrar solicitud de dictamen, donde se representan las principales dependencias entre los elementos fundamentales que conforman la estructuración del sistema, ubicados en el paquete ***"Sistema"*** y la estructuración del *framework* representados en el paquete ***"Symfony 2.8"***. Dentro de los elementos que componen al sistema se puede apreciar el paquete ***"View"***, que agrupa la vista correspondiente a la acción que se ejecuta en el caso de uso, que constituyen una extensión "html.twig" y que hereda del componente "base.html.twig"; en el paquete ***"Model"*** se encuentran las entidades "AFT", "SolicitudDictamenAFT" y "SolicitudDictamen"

encargadas de manejar los elementos referentes al negocio; en el paquete ***"Controller"*** se ubica el "SolicitudDictamenController", que se encarga de manejar la lógica del control del sistema que hereda del componente "BaseController.

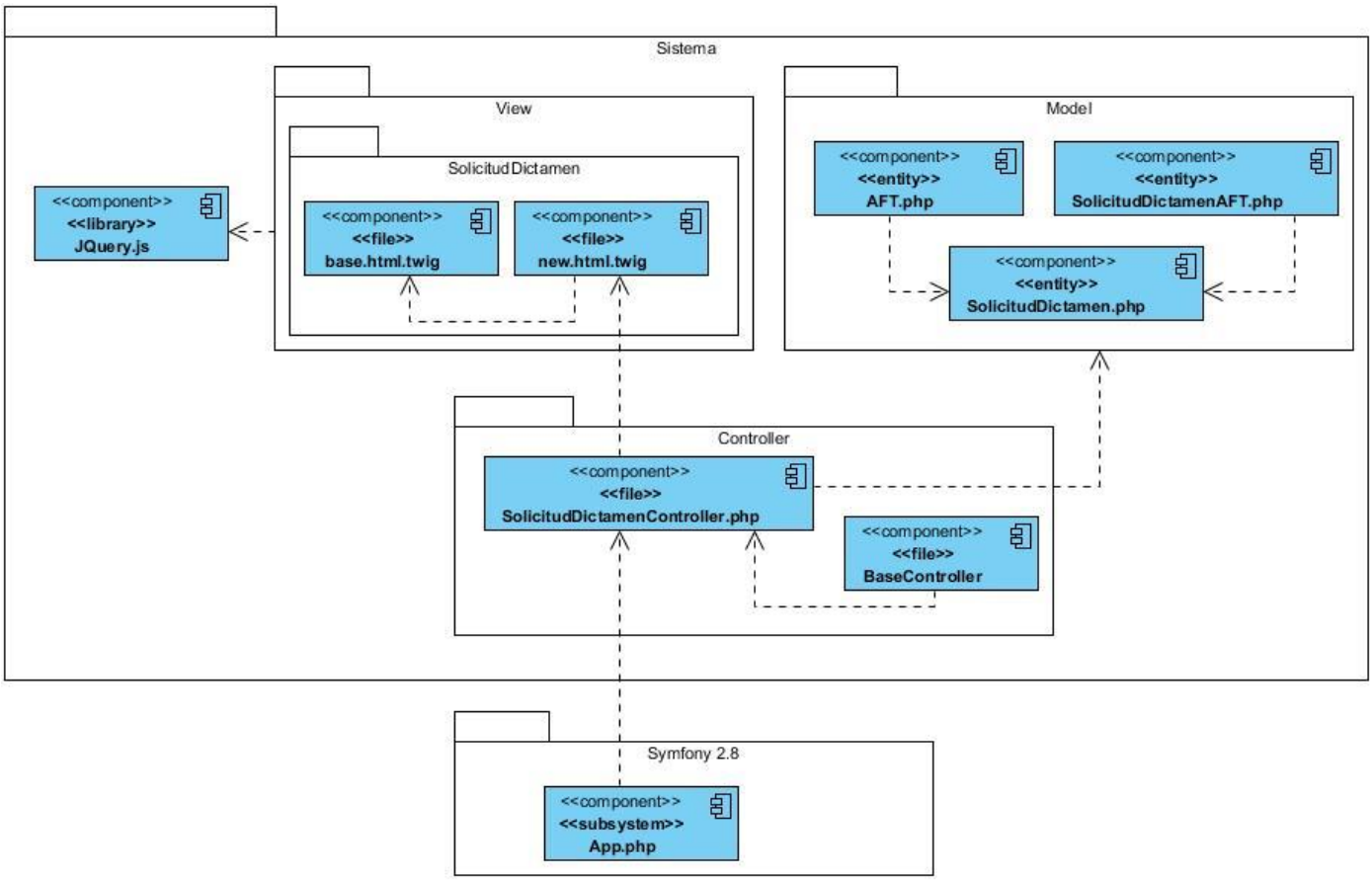


Fig. 8: Diagrama de componentes "Administrar solicitud de dictamen"

Modelo de despliegue

Un modelo de despliegue muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Según Jacobson se le puede considerar como una distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo, además agrega que el modelo de despliegue se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño ***(Jacobson, 2000)***.

En la figura 8 se muestra el diagrama correspondiente al modelo de despliegue de la propuesta de solución, donde se representan físicamente el servidor de la aplicación, el de la base de datos, el servicio web Ldap

disponible para la UCI y la estación cliente. Además, en cada relación se representan los diferentes protocolos de comunicación.

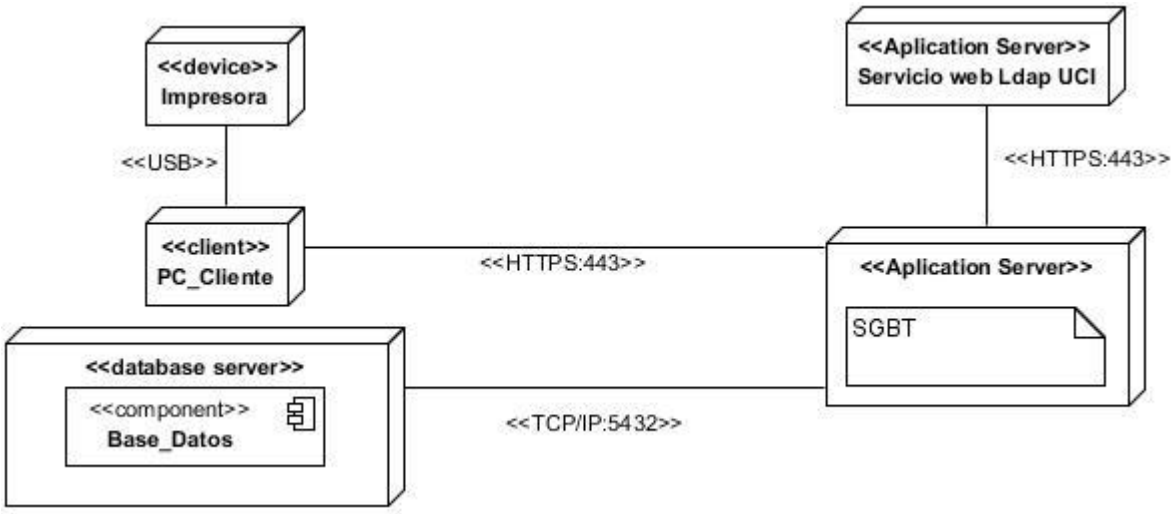


Fig. 9: Diagrama de despliegue

3.2 Código fuente

El código fuente de un sistema informático constituye el conjunto de líneas de texto que contienen las instrucciones, que deben ser procesadas mediante compiladores, ensambladores o intérpretes hacia el lenguaje código de máquina para su posterior ejecución. En diversas ocasiones la calidad de un software depende en gran medida de las buenas prácticas que se manifiesten en su proceso de implementación. A continuación, se muestran los principales elementos referentes a la implementación de la propuesta de solución.

Estándares y estilos de codificación

El propósito fundamental de los estándares de codificación es que el sistema tenga una arquitectura y un estilo consistente, independiente del autor, con lo cual resulte fácil de entender y por supuesto fácil de mantener. Partiendo de que la propuesta de solución se desarrolla utilizando el *framework* Symfony 2.3, se define el uso de los estándares PSR-1 *(PHP Standards Recommendation 1)* y PSR-2 *(PHP Standards Recommendation 2)* propuestos por esta tecnología y que de conjunto constituyen unos de los estándares

más difundidos dentro del lenguaje de programación PHP (Potencier, 2014). Estos estándares plantean fundamentalmente los siguientes elementos:

Se deben utilizar solamente las etiquetas <?php y <?=.

Se debe emplear solamente la codificación UTF-8 para el código PHP.

El código debe usar 4 espacios como indentación, no tabuladores.

Debe haber una línea en blanco después de la declaración del *"namespace"* y otra después del bloque de declaraciones *"use"*.

Las llaves de apertura de las estructuras de control deben estar en la misma línea, y las de cierre deben ir en la línea siguiente al cuerpo.

Los paréntesis de apertura en las estructuras de control no deben tener un espacio después de ellos, y los paréntesis de cierre no deben tener un espacio antes de ellos.

En el caso de las convenciones de nomenclatura que hacen uso de las mayúsculas y minúsculas en sus identificadores se utilizan tres estilos:

- El estilo **Camel Case** (camelCase) que define que la primera letra del identificador debe estar en minúscula y la primera letra de las siguientes palabras concatenadas en mayúscula, se utiliza para identificar los métodos y los parámetros que pudiesen contener estos.
- El estilo **Mayúsculas** (ALL_CAPS o UPPER_CASE) que plantea que todas las letras deben ir en mayúsculas se utiliza para identificar las constantes.
- El estilo **Minúsculas** (small_caps o lower_case) que precisa que todas las letras deben ir en minúsculas se utiliza para identificar las variables.

3.3 Modelo de pruebas

El modelo de pruebas describe principalmente cómo se prueban los componentes ejecutables en el modelo de implementación con pruebas de integración y de sistema, así como otros aspectos específicos del sistema (Jacobson, 2000). Las pruebas son una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos especificados, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente (IEEE, 1995).

Según Pressman los objetivos fundamentales del proceso de prueba están dados por las siguientes afirmaciones:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

Una prueba tiene éxito si descubre un error no detectado hasta entonces (Pressman, 2008).

Tipos de pruebas

Existe toda una tipología de pruebas aplicables a distintos contextos de desarrollo de aplicaciones. Según Pressman, existen siete tipos fundamentales de pruebas recomendadas sobre aplicaciones web las cuales son: pruebas de contenido, interfaz, navegación, componentes, configuración, seguridad y desempeño o rendimiento (Pressman, 2008).

Para la presente solución se requiere la realización de un grupo de pruebas que redunden en la correcta validación del sistema, como son:

- Pruebas unitarias: son pruebas que realiza el equipo de desarrollo y que, según Pressman: verifiquen que los componentes unitarios están codificados bajo condiciones de robustez, soportando el ingreso de datos erróneos o inesperados y demostrando así la capacidad de tratar errores de manera controlada (Pressman, 2008).
- Pruebas de función o a nivel de componentes, que ejercitan el contenido y las unidades funcionales dentro de la aplicación y se enfocan sobre un conjunto de pruebas que intentan descubrir errores en la misma (Pressman, 2008).
- Pruebas de desempeño o de rendimiento, que se aplican para descubrir problemas debido a la falta de recursos en el lado del servidor, ancho de banda de red inapropiado, capacidades inadecuadas de bases de datos, defectuosas o débiles capacidades del sistema operativo, funcionalidades mal diseñadas y otros conflictos de hardware o software que pueden conducir a un pobre desempeño cliente - servidor (Pressman, 2008).
- Pruebas de aceptación, que representan aquella fase del ciclo de vida de desarrollo de software en el que el equipo de desarrollo y el área usuaria de un sistema de información tienen que garantizar que el sistema desarrollado se corresponde con los requerimientos definidos (González, 2014).

Métodos de pruebas

Existen diversos métodos para realizar las pruebas de software, entre las más importantes se encuentran la prueba de Caja Blanca y prueba de Caja Negra.

Pruebas de Caja Blanca: También suelen ser llamadas estructurales o de cobertura lógica. En ellas se pretende investigar sobre la estructura interna del código, exceptuando detalles referidos a datos de entrada o salida, para probar la lógica del programa desde el punto de vista algorítmico. Realizan un seguimiento del código fuente según se va ejecutando los casos de prueba, determinándose de manera concreta las instrucciones, bloques, etc. que han sido ejecutados por los casos de prueba (Pressman, 2008).

En las pruebas de Caja Blanca se desarrollan casos de prueba que produzcan la ejecución de cada posible ruta del programa o módulo, considerándose una ruta como una combinación específica de condiciones manejadas por un programa (Pressman, 2008).

En las pruebas de Caja Blanca, se pretende indagar sobre la estructura interna del código, omitiendo detalles referidos a datos de entrada o salida. Su objetivo principal es probar la lógica del programa desde el punto de vista algorítmico. Estas se basan en el diseño de Casos de Prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante las pruebas de Caja Blanca el ingeniero de software puede obtener Casos de Prueba que (Pressman, 2008):

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

Las pruebas de Caja Blanca son consideradas entre las más importantes que se aplican a los sistemas, con la que se obtienen como resultados la disminución en un gran porciento el número de errores existentes en el software y por ende una mayor calidad y confiabilidad en la codificación (Pressman, 2008).

Como parte de las pruebas unitarias se utilizó este método utilizando la técnica de camino básico, para lo cual es necesario conocer el número de caminos independientes de un determinado algoritmo mediante el cálculo de la complejidad ciclomática (Pressman, 2008). Este se realiza de tres formas diferentes:

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

- La complejidad ciclomática V (G) de un grafo de flujo G se define como: V (G)= A-N+2 donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo.
- V (G) también se calcula como el resultado de P+1 donde P es el número de nodos predicados (nodos de los cuales parten dos o más aristas) que tiene contenido el grafo de flujo G.

Ejemplo de código de la funcionalidad newAction

```
public function newAction(Request $request)
{
    $aFT = new Aft();
    $form = $this->createForm('GestionBundle\Form\AFTType', $aFT);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $aFT->setEstado(true);
        $aFT->setTieneSolicitud(false);
        $aFT->setArea($this->getUser()->getArea());
        $em = $this->getDoctrine()->getManager();
        $em->persist($aFT);
        $em->flush();
        //setcookie('mensaje','Se ha creado el AFT satisfactoriamente');
        //setcookie('sms_type','success');
        return $this->redirectToRoute('aft_index');
    }

    return $this->render('GestionBundle:AFT:new.html.twig', array(
        'aFT' => $aFT,
        'form' => $form->createView(),
    ));
}
```

1
2
3
4
5

Fig. 10: Código de la funcionalidad newAction.

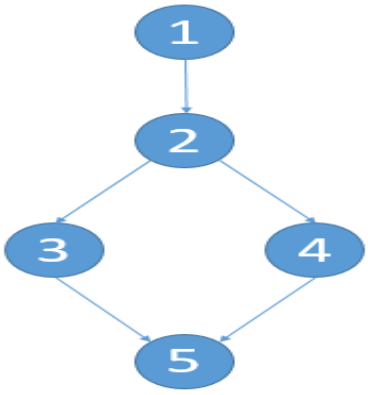


Fig. 11: Grafo de flujo asociado a la funcionalidad newAction.

Resultado del cálculo de la complejidad ciclomática:

V (G) = A-N+2=5-5+2=2

V (G) = P+1=1+1=2

A partir del resultado obtenido se determina que la funcionalidad presenta una complejidad ciclomática de 2, lo que deriva que existe a lo sumo 2 caminos lógicos por donde ejecutarse dicha funcionalidad.

A continuación, se muestran los caminos básicos:

Tab. 7: Camino básico

No.	Camino básico
1	1-2-3-5
2	1-2-4-5

Pruebas de Caja Negra: También suelen ser llamadas funcionales y basadas en especificaciones. En ellas se pretende examinar el programa en busca de que cuente con las funcionalidades que debe tener y como lleva a cabo las mismas, analizando siempre los resultados que devuelve y probando todas las entradas en sus valores válidos e inválidos (Pressman, 2008).

Al ejecutar las pruebas de Caja Negra se desarrollan casos de prueba reales para cada condición o combinación de condiciones y se analizan los resultados que arroja el sistema para cada uno de los casos. En esta estrategia se verifica el programa considerándolo una caja negra. Las pruebas no se hacen en base al código, sino a la interfaz. No importa que se cubran todas las rutas dentro del programa, lo importante es probar todas las entradas en sus valores válidos e inválidos y lograr que el sistema tenga una interfaz amigable (Pressman, 2008).

Las pruebas de Caja Negra según Pressman buscan encontrar errores en cinco categorías:

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y terminación.

Dentro del método de Caja Negra se decide el uso de la técnica de **Partición de equivalencia**, que es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. Además, esta técnica se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de pruebas a desarrollar. Para su implementación se divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software a partir de las cuales pueden derivarse casos de prueba (Pressman, 2008).

Para desarrollar las pruebas de desempeño o de rendimiento se pueden aplicar pruebas de carga y de tensión. En el caso de la presente investigación debido a que la propuesta de solución durante su despliegue no estará expuesta a grandes niveles de tensión ni concurrencia significante de usuarios, se decide realizar solamente **pruebas de carga**, que tienen como objetivo determinar cómo la aplicación en su ambiente del lado del servidor responderá ante varias condiciones de carga, a partir de condiciones de pruebas definidas por las permutaciones entre variables que dependen del número de usuarios concurrentes, el número de transacciones en línea por usuario por unidad de tiempo y la carga de datos procesada por el servidor por transacción (Pressman, 2008).

Diseño de Casos de Prueba (DCP)

Un caso de prueba constituye el conjunto de entradas, condiciones de ejecución y resultados esperados que se desarrollada para validar una funcionalidad específica en la aplicación bajo prueba. La calidad de las pruebas es proporcional al número de casos de prueba, debido a que cada caso de prueba refleja diferentes escenarios, condiciones, o flujo de trabajo (Lewis, 2005). A continuación, se muestra el DCP del CU Administrar solicitud de dictamen. Para profundizar en el resto de los DCP realizados, se deben consultar los artefactos del modelo de pruebas del expediente de proyecto de la propuesta de solución.

DCP del caso de uso: Administrar solicitud de dictamen

- **Descripción general**

El presente CU se inicia cuando el usuario despliega la opción Solicitudes, selecciona las operaciones de adicionar una nueva solicitud o listar las solicitudes existentes y culmina con la realización de una de las acciones *Adicionar*, *Buscar, Listar o Visualizar* alguna solicitud.

- **Condiciones de ejecución**

El usuario debe estar autenticado en el sistema, así como contar con los permisos para realizar alguna de las operaciones antes mencionadas. En el caso de que se pretenda *Visualizar o Mostrar* alguna solicitud, esta debe haber sido adicionada anteriormente al sistema.

- **Secciones a probar en el CU: Administrar solicitud de dictamen**

Tab. 8: DCP Administrar solicitud de dictamen

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC 1: "Adicionar Solicitud".	**EC 1.1:** El usuario selecciona los AFT que desea adicionar a la solicitud.	El sistema adiciona una nueva solicitud y envía una notificación al usuario que confirma que su solicitud se creó correctamente.	1. Seleccionar la pestaña "Solicitudes" en el menú de la izquierda. 2. Seleccionar dentro de esta pestaña la opción "Nueva solicitud". 3. Seleccionar los AFT que se desean adicionar a la solicitud. 4. Seleccionar la opción "Crear" que se muestra en la parte inferior del formulario.
SC 2: "Buscar Solicitud".	**EC 2.1:** Buscar una solicitud.	El sistema busca la solicitud.	1. Seleccionar la pestaña "Solicitudes" en el menú de la izquierda. 2. Inserta los datos de la solicitud que desea buscar.
SC 3: "Visualizar Solicitud".	**EC 3.1:** Visualizar una solicitud.	El sistema visualiza la solicitud seleccionada con todos sus datos y documentos asociados.	1. Seleccionar la pestaña "Solicitudes" en el menú de la izquierda. 2. Seleccionar el ícono "Ver Solicitud" de la solicitud, de la tabla que muestra la lista de solicitudes creadas.
SC 4: "Listar Solicitud".	**EC 4.1:** Listar una o varias solicitudes.	El sistema lista las solicitudes existentes.	1. Seleccionar la pestaña "Solicitudes" en el menú de la izquierda.

- **Descripción de las Variables**

Tab. 9: Descripción de las variables

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	AFT	Campo de selección múltiple.	No	Debe seleccionarse al menos uno para crear una solicitud.
2	Buscar	Campo de texto	No	Muestra la solicitud según la búsqueda realizada.

- **Matrices de datos**

Adicionar solicitud

Tab. 10: Matriz de datos. Adicionar solicitud

Id. Del escenario	Escenario	Nombre	Buscar	Respuesta del sistema	Resultado de la prueba
EC 1.1	El usuario selecciona los AFT que desea adicionar a la solicitud.	V/(Seleccionado)	V	El sistema adiciona una nueva solicitud y envía una notificación al usuario que confirma que su solicitud se creó correctamente.	Satisfactorio

Buscar solicitud

Tab. 11: Matriz de datos. Mostrar solicitud

Id. Del escenario	Escenario	Nombre	Buscar	Respuesta del sistema	Resultado de la prueba
EC 2.1	Buscar una solicitud.	V(No seleccionado)	V/(descripción o número de inventario)	El sistema muestra los datos y documentos asociados a la solicitud de la búsqueda realizada.	Satisfactorio

Visualizar solicitud

Tab. 12: Matriz de datos. Visualizar solicitud

Id. Del escenario	Escenario	Nombre	Buscar	Respuesta del sistema	Resultado de la prueba
EC 3.1	Visualiza los detalles de una solicitud.	V	V	El sistema visualiza los datos y documentos asociados a la solicitud seleccionada.	Satisfactorio

Listar solicitud

Tab. 13: Matriz de datos. Listar solicitud

Id. Del escenario	Escenario	Nombre	Buscar	Respuesta del sistema	Resultado de la prueba
EC 3.1	Lista las solicitudes.	V	V	El sistema lista las solicitudes existentes.	Satisfactorio

Resultados de la aplicación de las pruebas

Como parte de la ejecución de las pruebas de caja negra se realizaron 3 iteraciones de pruebas que se representan en la figura 11. En la primera se identificaron 15 no conformidades, clasificadas en 13 no significativas y 2 significativas. Una vez corregidas, se procedió a realizar una segunda iteración, en la que se identificaron 4 nuevas no conformidades de tipo no significativas. Finalmente se realizó una última iteración en la que no se encontraron deficiencias, razón por la que se definió no realizar más iteraciones.

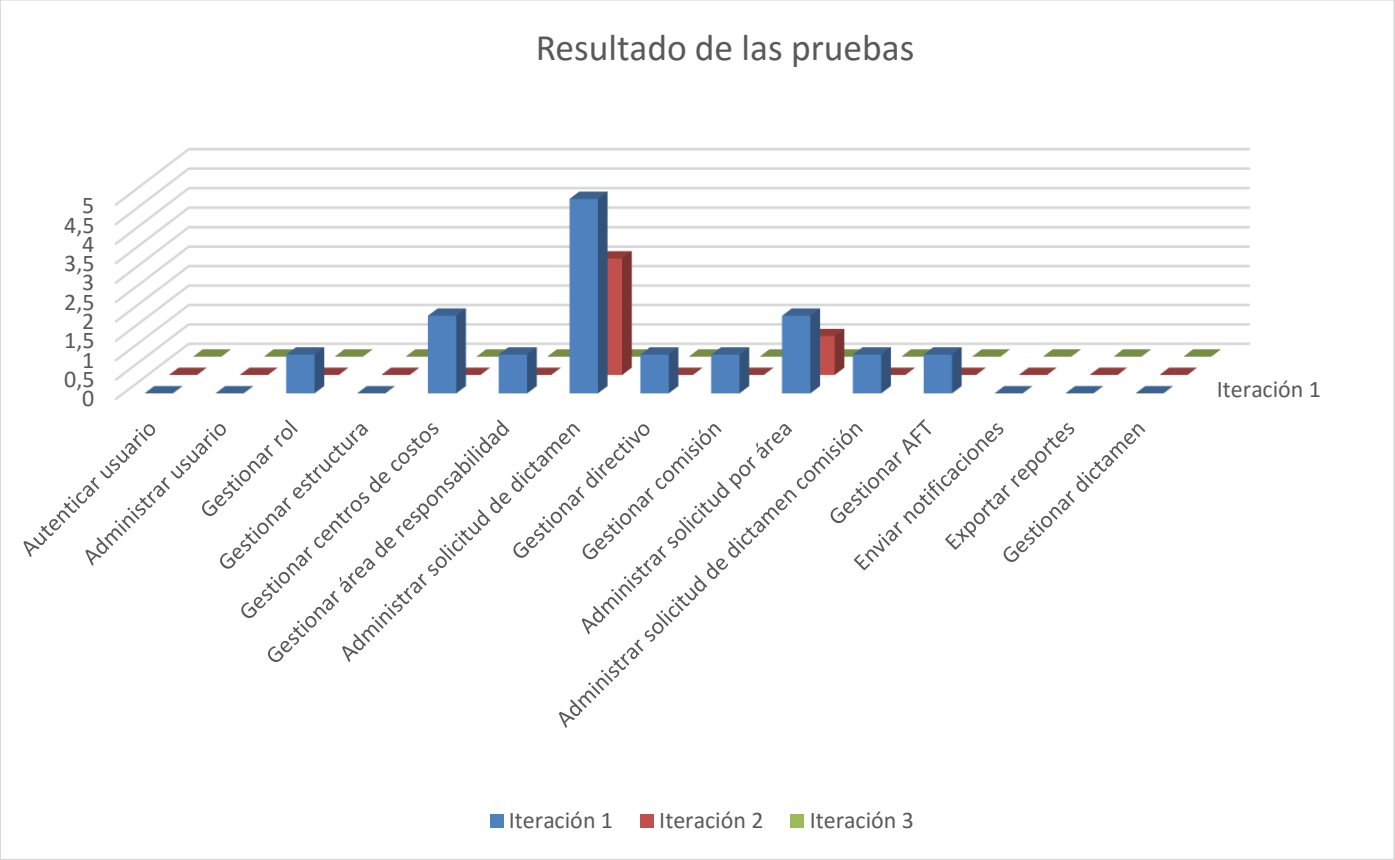


Fig. 12: Resultados de las iteraciones de las pruebas funcionales

Para la realización de las pruebas de carga se utilizó la herramienta JMeter en su versión 2.10, mediante la cual se validó el RnF3 que plantea que el tiempo de respuesta promedio del sistema no debe superar los 8 segundos con 150 usuarios conectados simultáneamente. El resultado de estas pruebas se presenta en la figura 12, donde se muestra de las primeras 27 peticiones por ejemplo, el tiempo de respuesta en milisegundos para cada una en la columna “Tiempo de muestra (ms)”, los errores detectados por la herramienta durante el proceso de petición y respuesta, en la columna “Estado”, el tiempo medio de respuesta para la muestra de 150 peticiones simultáneas (Media) y el valor de la desviación entre el mayor y el menor tiempo de respuesta (Desviación).

tra #	Tiempo de comie...	Nombre del hilo	Etiqueta	Tiempo de Muest...	Estado	Bytes	Latency
1	10:57:34.146	Grupo de Hilos 1-5	/luism/web/app_...	557		809	557
2	10:57:34.125	Grupo de Hilos 1-2	/luism/web/app_...	646		809	646
3	10:57:34.195	Grupo de Hilos 1...	/luism/web/app_...	639		809	639
4	10:57:34.140	Grupo de Hilos 1-4	/luism/web/app_...	789		809	789
5	10:57:34.156	Grupo de Hilos 1-6	/luism/web/app_...	825		809	825
6	10:57:34.184	Grupo de Hilos 1-9	/luism/web/app_...	974		809	974
7	10:57:34.205	Grupo de Hilos 1...	/luism/web/app_...	1051		809	1051
8	10:57:34.171	Grupo de Hilos 1-8	/luism/web/app_...	1143		809	1143
9	10:57:34.116	Grupo de Hilos 1-1	/luism/web/app_...	1306		809	1306
10	10:57:34.219	Grupo de Hilos 1...	/luism/web/app_...	1375		809	1375
11	10:57:34.248	Grupo de Hilos 1...	/luism/web/app_...	1393		809	1393
12	10:57:34.236	Grupo de Hilos 1...	/luism/web/app_...	1471		809	1471
13	10:57:34.265	Grupo de Hilos 1...	/luism/web/app_...	1574		809	1574
14	10:57:34.216	Grupo de Hilos 1...	/luism/web/app_...	1740		809	1740
15	10:57:34.133	Grupo de Hilos 1-3	/luism/web/app_...	1840		809	1840
16	10:57:34.307	Grupo de Hilos 1...	/luism/web/app_...	1761		809	1761
17	10:57:34.165	Grupo de Hilos 1-7	/luism/web/app_...	2007		809	2007
18	10:57:34.293	Grupo de Hilos 1...	/luism/web/app_...	1944		809	1944
19	10:57:34.189	Grupo de Hilos 1...	/luism/web/app_...	2119		809	2119
20	10:57:34.307	Grupo de Hilos 1...	/luism/web/app_...	2064		809	2064
21	10:57:34.288	Grupo de Hilos 1...	/luism/web/app_...	2162		809	2162
22	10:57:34.404	Grupo de Hilos 1...	/luism/web/app_...	2114		809	2114
23	10:57:34.440	Grupo de Hilos 1...	/luism/web/app_...	2142		809	2142
24	10:57:34.344	Grupo de Hilos 1...	/luism/web/app_...	2315		809	2315
25	10:57:34.448	Grupo de Hilos 1...	/luism/web/app_...	2293		809	2293
26	10:57:34.264	Grupo de Hilos 1...	/luism/web/app_...	2550		809	2550
27	10:57:34.472	Grupo de Hilos 1...	/luism/web/app_...	2405		809	2405

automatically? Child samples? **No. de Muestras** 150 **Última Muestra** 11635 **Media** 6970 **Desviación** 3593

Fig. 13: Resultados de las pruebas de carga

3.4 Conclusiones del capítulo

Durante el desarrollo de este capítulo quedó definido el modelo de implementación del sistema de gestión mediante el diagrama de componentes. La distribución física del sistema mediante el diagrama de despliegue ofreció desde la perspectiva física, la distribución de los elementos operativos para el despliegue. Los estándares de codificación y los estilos de programación utilizados fueron descritos para hacer más fácil el entendimiento del código y permitir un mejor mantenimiento de la aplicación en un futuro. Mediante el desarrollo del proceso de pruebas se pudo identificar y resolver los errores en la implementación aumentando la calidad del producto final obtenido. Se comprobó que el mismo satisface las funcionalidades requeridas y cumple con los Requisitos No Funcionales planteados.

Conclusiones

Una vez culminada la investigación se puede afirmar que se les dio cumplimiento a los objetivos planteados, arribando a las siguientes conclusiones.

Con el estudio de los sistemas para la gestión de información, enmarcados en los sistemas para la gestión de bajas técnicas de los AFT existentes, se pudo constatar que estas soluciones no satisfacen la problemática, lo cual evidencia la necesidad de la presente investigación.

A partir de la realización del análisis del sistema para la gestión de bajas técnicas de los AFT, se seleccionaron las tecnologías, herramientas y metodología de desarrollo que se ajustan a la situación del negocio.

Como parte del proceso de diseño se obtuvieron los diagramas y artefactos definidos por la metodología de desarrollo AUP-UCI, lo cual contribuirá a que el sistema sea escalable en el futuro.

Para validar el correcto funcionamiento de la propuesta de solución, se realizaron pruebas funcionales en 3 iteraciones que arrojaron 15, 4 y 0 no conformidades respectivamente.

Se realizaron pruebas de carga, de funcionalidad y de comportamiento con las cuales se pudo corroborar que el sistema satisface los requisitos funcionales y no funcionales planteados.

Recomendaciones

Desplegar el sistema en un ambiente profesional, que permita su explotación por la comunidad universitaria.

Publicar la documentación del proyecto para facilitarle a futuros desarrolladores la implementación de nuevas funcionalidades.

Referencias Bibliográficas

Abran, Alain and Moore, James W. 2004. *Swebok. Guide to the Software Engineering Body of Knowledge.* Estados Unidos de América : IEEE Computer Society Professional Practices Committee, 2004. 204.

Administracion Universitaria. 2012. *Administracion de bienes y servicios.* Mexico : s.n., 2012.

AFT. 2012. Administracion . [En línea] junio de 2012. [Citado el: 2016 de 10 de 10.] http://www.administracionmoderna.com/2012/06/activo-fijo-tangible.html.

Almeira, Adriana. 2007. *Arquitectura de Software: Estilos y Patrones.* Argentina : Universidad Nacional De La Patagonia San Juan Bosco, 2007.

Amparo López Gaona. 2012. El modelo Entidad-Relación. [En línea] Universidad Nacional Autónoma de México. Facultad de Ciencias, 2012. [Citado el: 14 de 04 de 2017.] http://hp.fciencias.unam.mx/~alg/bd/.

Assets. **ASSETS. 2005.** Cuba : s.n., 2005.

Bartak, Pedro. 2013. [En línea] 9 de 12 de 2013. [Citado el: 5 de 4 de 2017.] http://www.grandespymes.com.ar/2013/12/29/definicion-y-componentes-del-sistema-de-control-de-gestion-en-las-empresas/.

Berberat, Esteve. 2012. *Visual Paradigm.* Francia : s.n., 2012. Tesis.

BPD. **IBM. 2011.** 2011.

Canales, Pedro. 2007. Visual Paradigm. [En línea] 2007. https://www.adictosaltrabajo.com/tutoriales/vparadigm/.

Cano, Jose H. 2003. Valencia : s.n., 2003.

Echevarría, Dayma. 2010. *Proceso de perfeccionamiento empresarial.* La Habana : Universidad de la Habana, 2010.

EUMED. 2012. *Ingenieria del Software.* Mexico : Universidad de Mexico, 2012.

Figueroa, Robert G y Solis, Camilo J. 2012. Metodologías Tradicionales vs. Metodologías Ágiles. [En línea] 2012. [Citado el: 26 de noviembre de 2016.] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.

Gauchat, Juan Diego. 2013. *Gran libro de html5, css3 y javascript.* España : CEDRO, 2013.

González, Dra. Anaisa Hernández. 2013. *Diagramas de Casos de Uso del Negocio y del Sistema.* Habana, Cuba : Instituto Superior Politécnico Jose Antonio Hechavarría (CUJAE), 2013.

González, Jesus Ponce, Dominguez, Fransisco Jose, Rodriguez, Javier Gutierrez and Escalona, Maria Jose. 2014. *Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental.* España : Universidad de Sevilla, Universidad de Sevilla, 2014. ISSN 1888-0967.

Heurtel, Olivier. 2011. *Desarrollar un sitio Web dinámico e interactivo.* s.l. : ENI Ediciones, 2011. 497.

IBM. *Reglas de negocio.*

IEEE. 1995. IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools. 1995.

INEGI. 2007. Mexico : s.n., 2007.

Jacobson, Ivar, Boocch, Grady and Rumbaugh, James. 2000. [aut. libro] Ivar Jacobson. *El proceso unificado de desarrollo de software.* Madrid, España : Pearsons Educacion S.A, 2000.

Larman, Craig. 2003. *UML y Patrones.* s.l. : Prentice Hall 2da Edición, 2003.

Lewis, William E. 2005. *Software testing and continuous quality improvement.* Estados Unidos : CRC Press LLC, 2005. ISBN: 0-8493-2524-2.

LLonch. 2016. Symfony. [En línea] 2016. [Citado el: 2017 de febrero de 2017.] http://symfony.com.

Lopes-Martinez, Igor. 2012. 2012.

López, David. 2014. Genbeta. [En línea] 9 de enero de 2014. https://www.genbetadev.com/herramientas/netbeans-1.

Mejora de Gestión. Gestión. [En línea] [Citado el: 2016 de 10 de 15.] http://mejoratugestion.com/mejora-tu-gestion/que-es-un-sistema-de-gestion/.

Patrones de Caso de Uso. **Cuesta, Saúl.** s.l. : SG.

Pilar. 2011. programación web. [En línea] mayo de 2011. [Citado el: 22 de febrero de 2017.] http://progpagweb.blogspot.com/2011/05/definicion-y-caracteristicas-de-php-con.html.

Polanco, Manuel Ponce. 2012. *Sistema de control de gestión.* Lima, Perú : s.n., 2012.

PostgreSQL. 2014. Postgresql. [En línea] 2014. [Citado el: 22 de febrero de 2017.] http://www.postgresql.org.es/node/2984.

Potencier, Fabien, Weaver, Ryan and Eguiluz, Javier. 2014. *Buenas prácticas oficiales de Symfony.* 2014.

Pressman, Roger S. y David Lowe. 2008. *Web Engineering: A Practitioner's Approach.* Estados Unidos : RS Pressman & Associates, 2008. 765.

SIAC. SIAC. [En línea] http://sistemacontableec.com/sistema-contable-administrativo/.

SIAC. **Sistema Contable Aministrativo . 2000.** España : SIAC, 2000.

Sommerville. 2005. *Ingeniería del Software.* Madrid, España : Pearson Education S. A., 2005.

Sorolla2. **Sorolla2. 2012.** España : España, 2012.

Torrecilla, Pablo. 2012. 2012.

TributosNet. 2016. Inventarios. [En línea] julio de 2016. [Citado el: 2016 de 10 de 15.] http://www.tributos.net/definicion-de-gestion-de-inventarios-1013/.

UCI. 2014. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : UCI, 2014.

Versat. **Versat - Sarasola. 2012.** Cuba : s.n., 2012.

Bibliografía

Abran, Alain and Moore, James W. 2004. *Swebok. Guide to the Software Engineering Body of Knowledge.* Estados Unidos de América : IEEE Computer Society Professional Practices Committee, 2004. 204.

Administracion Universitaria. 2012. *Administracion de bienes y servicios.* Mexico : s.n., 2012.

AFT. 2012. Administracion . [En línea] junio de 2012. [Citado el: 2016 de 10 de 10.] http://www.administracionmoderna.com/2012/06/activo-fijo-tangible.html.

Almeira, Adriana. 2007. *Arquitectura de Software: Estilos y Patrones.* Argentina : Universidad Nacional De La Patagonia San Juan Bosco, 2007.

Amparo López Gaona. 2012. El modelo Entidad-Relación. [En línea] Universidad Nacional Autónoma de México. Facultad de Ciencias, 2012. [Citado el: 14 de 04 de 2017.] http://hp.fciencias.unam.mx/~alg/bd/.

Assets. **ASSETS. 2005.** Cuba : s.n., 2005.

Bartak, Pedro. 2013. [En línea] 9 de 12 de 2013. [Citado el: 5 de 4 de 2017.] http://www.grandespymes.com.ar/2013/12/29/definicion-y-componentes-del-sistema-de-control-de-gestion-en-las-empresas/.

Berberat, Esteve. 2012. *Visual Paradigm.* Francia : s.n., 2012. Tesis.

BPD. **IBM. 2011.** 2011.

Canales, Pedro. 2007. Visual Paradigm. [En línea] 2007. https://www.adictosaltrabajo.com/tutoriales/vparadigm/.

Cano, Jose H. 2003. Valencia : s.n., 2003.

Echevarría, Dayma. 2010. *Proceso de perfeccionamiento empresarial.* La Habana : Universidad de la Habana, 2010.

EUMED. 2012. *Ingenieria del Software.* Mexico : Universidad de Mexico, 2012.

Figueroa, Robert G y Solis, Camilo J. 2012. Metodologías Tradicionales vs. Metodologías Ágiles. [En línea] 2012. [Citado el: 26 de noviembre de 2016.] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.

Gauchat, Juan Diego. 2013. *Gran libro de html5, css3 y javascript.* España : CEDRO, 2013.

González, Dra. Anaisa Hernández. 2013. *Diagramas de Casos de Uso del Negocio y del Sistema.* Habana, Cuba : Instituto Superior Politécnico Jose Antonio Hechavarría (CUJAE), 2013.

González, Jesus Ponce, Dominguez, Fransisco Jose, Rodriguez, Javier Gutierrez and Escalona, Maria Jose. 2014. *Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental.* España : Universidad de Sevilla, Universidad de Sevilla, 2014. ISSN 1888-0967.

Heurtel, Olivier. 2011. *Desarrollar un sitio Web dinámico e interactivo.* s.l. : ENI Ediciones, 2011. 497.

IBM. *Reglas de negocio.*

IEEE. 1995. IEEE Recommended Practice for the Adoption of Computer-Aided Software Engineering (CASE) Tools. 1995.

INEGI. 2007. Mexico : s.n., 2007.

Jacobson, Ivar, Boocch, Grady and Rumbaugh, James. 2000. [aut. libro] Ivar Jacobson. *El proceso unificado de desarrollo de software.* Madrid, España : Pearsons Educacion S.A, 2000.

Larman, Craig. 2003. *UML y Patrones.* s.l. : Prentice Hall 2da Edición, 2003.

Lewis, William E. 2005. *Software testing and continuous quality improvement.* Estados Unidos : CRC Press LLC, 2005. ISBN: 0-8493-2524-2.

LLonch. 2016. Symfony. [En línea] 2016. [Citado el: 2017 de febrero de 2017.] http://symfony.com.

Lopes-Martinez, Igor. 2012. 2012.

López, David. 2014. Genbeta. [En línea] 9 de enero de 2014. https://www.genbetadev.com/herramientas/netbeans-1.

Mejora de Gestión. Gestión. [En línea] [Citado el: 2016 de 10 de 15.] http://mejoratugestion.com/mejora-tu-gestion/que-es-un-sistema-de-gestion/.

Patrones de Caso de Uso. **Cuesta, Saúl.** s.l. : SG.

Pilar. 2011. programación web. [En línea] mayo de 2011. [Citado el: 22 de febrero de 2017.] http://progpagweb.blogspot.com/2011/05/definicion-y-caracteristicas-de-php-con.html.

Polanco, Manuel Ponce. 2012. *Sistema de control de gestión.* Lima, Perú : s.n., 2012.

PostgreSQL. 2014. Postgresql. [En línea] 2014. [Citado el: 22 de febrero de 2017.] http://www.postgresql.org.es/node/2984.

Potencier, Fabien, Weaver, Ryan and Eguiluz, Javier. 2014. *Buenas prácticas oficiales de Symfony.* 2014.

Pressman, Roger S. y David Lowe. 2008. *Web Engineering: A Practitioner's Approach.* Estados Unidos : RS Pressman & Associates, 2008. 765.

SIAC. SIAC. [En línea] http://sistemacontableec.com/sistema-contable-administrativo/.

SIAC. **Sistema Contable Aministrativo . 2000.** España : SIAC, 2000.

Sommerville. 2005. *Ingeniería del Software.* Madrid, España : Pearson Education S. A., 2005.

Sorolla2. **Sorolla2. 2012.** España : España, 2012.

Torrecilla, Pablo. 2012. 2012.

TributosNet. 2016. Inventarios. [En línea] julio de 2016. [Citado el: 2016 de 10 de 15.] http://www.tributos.net/definicion-de-gestion-de-inventarios-1013/.

UCI. 2014. *Metodología de desarrollo para la actividad productiva de la UCI.* La Habana : UCI, 2014.

Versat. **Versat - Sarasola. 2012.** Cuba : s.n., 2012.

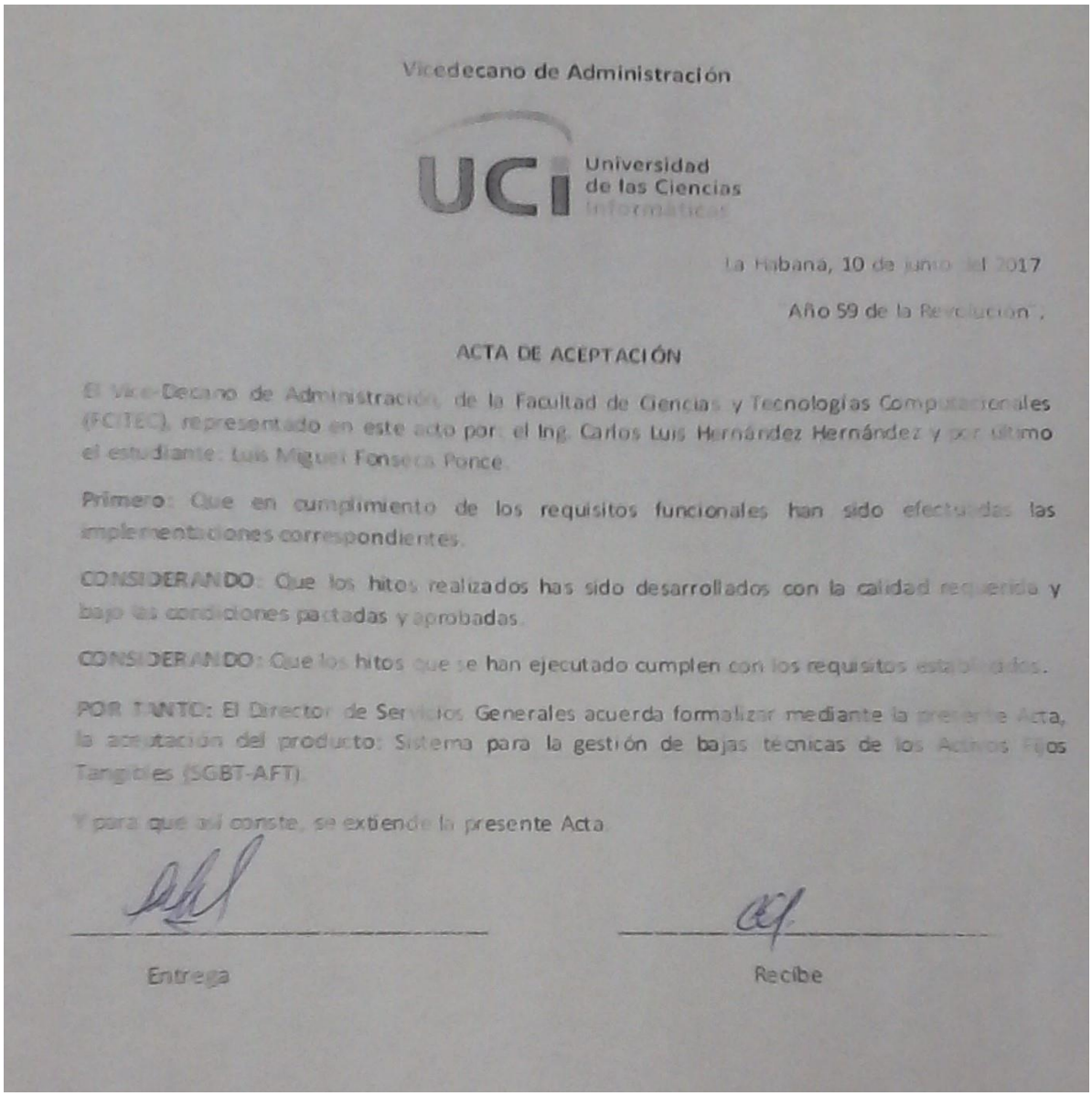
Portal en español sobre postgres SQL. http://www.postgresql.org.es/sobre_postgresql**.**

Bolivia, Tribunal Constitucional de. Manual de Administración de Activos Fijos. 2006.

Implementación SIG, Sistema Integrado de Gestión. www.implementacionsig.com/index.php/23-noticiac/28-que-es-un-sistema-de-gestion.

Anexos

Anexo1: Acta de aceptación de la propuesta de solución



Vicedecano de Administración

Universidad de las Ciencias Informáticas

La Habana, 10 de junio del 2017

Año 59 de la Revolución".

ACTA DE ACEPTACIÓN

El Vice-Decano de Administración de la Facultad de Ciencias y Tecnologías Computacionales (FCITEC), representado en este acto por el Ing. Carlos Luis Hernández Hernández y por último el estudiante: Luis Miguel Fonseca Ponce.

Primero: Que en cumplimiento de los requisitos funcionales han sido efectuadas las implementaciones correspondientes.

CONSIDERANDO: Que los hitos realizados has sido desarrollados con la calidad requerida y bajo las condiciones pactadas y aprobadas.

CONSIDERANDO: Que los hitos que se han ejecutado cumplen con los requisitos establecidos.

POR TANTO: El Director de Servicios Generales acuerda formalizar mediante la presente Acta, la aceptación del producto: Sistema para la gestión de bajas técnicas de los Activos Fijos Tangibles (SGBT-AFT).

Y para que así conste, se extiende la presente Acta.

Entrega

Recibe