Universidad de las Ciencias Informáticas

Facultad 1

“Sistema automatizado para la gestión de inventario de Activos Fijos Tangibles en la empresa UEB CAI Bartolomé Masó Márquez”

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Leticia Rodríguez Viera

Tutores: Ing. Eduardo Rojas Escobar

Ing. Gleidis Y. Rosabal Espinosa

La Habana, junio del 2014

DECLARACIÓN DE AUTORÍA

Declaración de Autoría

Declaro ser el legítimo autor del presente trabajo y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor
Leticia Rodríguez viera

_____

Firma del Tutor
Eduardo Rojas Escobar

_____

Firma del Tutor
Gleidis Y. Rosabal Espinosa

Síntesis de los tutores

Tutor: Ing. Gleidis Yuriannis Rosabal Espinosa

Clasificación: Profesional

Correo: gyrosabal@uci.cu

Síntesis: Graduada en la Universidad de la Ciencias Informáticas en el año 2011. Se desempeña como especialista general en el centro CIGED de la facultad 2.

Tutor: Ing. Eduardo Rojas Escobar

Clasificación: Profesional

Correo: erescobar@uci.cu

Síntesis: Graduado en la Universidad de la Ciencias Informáticas en el año 2012. Se desempeña como especialista general en el centro CEIGE de la facultad 3.

DEDICATORIA

A quien me ha inspirado la fuerza y la voluntad de seguir luchando por un buen futuro, mi hijito del alma.

A quienes me forjaron, educaron y siempre estarán guiándome donde quiera que estén, mi abuela y mi abuelo.

A los eternos inspiradores de todo lo sabio y bueno que hago, mi madre y mi padre.

A mi hermano no imagino que sería la vida sin tí.

A mi esposo fiel acompañante y gran apoyo en toda esta dura trayectoria.

A ustedes todos queridos, dedico yo esta tesis.

Agradezco:

A mis abuelos, Noemí y Juan por forjar en mí las cualidades de una buena persona, por el amor y la dulzura de toda una vida.

A mis padres, Idelisa y Jorge por estar siempre presente en las buenas y malas, darme todo su amor y confiar en mí.

A lo más bello que me ha dado la vida mi hijito Bruno, que dios me lo bendiga por siempre.

A mi hermano Jorgito por quererme tanto y ser siempre genial.

A mi esposo Rafael por apoyarme y estar presente dándome toda la fuerza que necesitaba para poder terminar lo que ya había comenzado.

A mis tutores, Gleidis y Eduardo por ser tan dedicados y apoyarme en todo el recorrido de la tesis, les agradezco de corazón sin su ayuda nada fuese posible.

A mis tías Alina, Aida, Ruby, Delvis, por brindarme siempre su ayuda y su cariño.

A mis primas Yordania y Anais por ser las mejores del mundo, a Liliana y a mi primo Yoan en fin a toda la familia que siempre está presente dando su granito de arena.

A la Universidad de la Ciencias Informáticas por acogerme y convertirme en una nueva persona de bien.

A todos mis profesores por contribuir en mi formación humana y profesional.

A todos aquellos amigos de aquí y de allá que brindaron su mano para levantarme y poder seguir adelante.

A todas estas lindas personas muchas gracias, agradecida para toda la vida:

Letícia

RESUMEN

Las empresas 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 de la empresa UEB CAI Bartolomé Masó Márquez 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 así la calidad y rapidez del proceso. Todo esto conlleva a que se produzcan una serie de irregularidades entre las que se incluyen, en el peor de los casos, pérdidas de los medios y bienes debido a la mala planificación y asignación de los mismos. DELTA, la solución que se propone en el presente trabajo pretende dar respuesta a esta problemática, mediante la creación de un sistema automatizado para la gestión de inventario de los AFT.

Para desarrollar el trabajo se estudiaron sistemas informáticos similares al que se deseaba implementar. Se describieron las principales tecnologías, herramientas y metodologías de desarrollo, así como los patrones utilizados. Se generaron los artefactos más importantes de la metodología seleccionada y se realizaron pruebas para verificar la calidad del sistema implementado.

El resultado del trabajo es un sistema que permite la agilización de los procesos relacionados con la gestión de inventario de los AFT, permitiendo un mayor control sobre los medios con que cuenta la empresa.

Palabras clave: activo fijo tangible, inventario, sistema de gestión.

Índice

Introducción 4

Capítulo.1 Fundamentación teórica 9

1.1. Introducción 9
1.2. Conceptos fundamentales 9
1.3. Análisis de sistemas informáticos 10
1.3.1. Sistemas internacionales 11
1.3.2. Sistemas nacionales 12
1.3.3. Valoración crítica sobre los sistemas existentes 14
1.4. Metodología de desarrollo de software 14
1.5. Lenguaje unificado de modelado (UML) 19
1.6. Herramienta de modelado 19
1.7. Herramientas de desarrollo 21
1.7.1. Entorno de desarrollo integrado (IDE) 21
1.7.2. Herramienta para el diseño de los reportes 22
1.8. Sistema gestor de base de datos 22
1.9. Lenguaje de programación 23
1.10. Frameworks 23
1.11. Librerías 24
1.12. Conclusiones parciales 25

Capítulo.2 Análisis y diseño 26

2.1. Introducción 26
2.2. Propuesta de solución 26
2.3. Objetivos estratégicos de la organización 26
2.4. Modelo de dominio 27
2.4.1. Descripción de las clases del modelo de dominio 27
2.5. Procesos de negocio 28
2.5.1. Descripción de los procesos de negocio. 30
2.6. Requisitos funcionales 31
2.7. Requisitos no funcionales 33
2.8. Diagrama de casos de uso 35

ÍNDICE

2.9. Especificación y resumen de los casos de uso 36
2.10. Patrones utilizados 45
2.10.1. Patrones de diseño 45
2.10.2. Patrón arquitectónico 48
2.11. Modelo de diseño 49
2.11.1. Diagrama de paquetes 49
2.11.2. Diagrama de Clases del Diseño 51
2.12. Modelo de datos 52
2.13. Diagrama de interacción del diseño 54
2.14. Conclusiones parciales 55
Capítulo.3 Implementación y pruebas 56
3.1. Introducción 56
3.2. Modelo de implementación 56
3.2.1. Diagrama de componente 56
3.2.2. Diagrama de despliegue 57
3.3. Justificación del estándar de codificación 58
3.4. Descripción de la implementación del sistema 59
3.5. Pruebas 60
3.5.1. Resultados de las pruebas 63
3.6. Conclusiones parciales 66
Conclusiones 67
Recomendaciones 68
Glosario de Términos 69
Referencias bibliográficas 71
Anexos 74
Anexo.1 Especificación y resumen de los casos de uso 74
Anexo 2. Diagrama de las clases de diseño del paquete Seguridad 113
Anexo.3 Caso de prueba para el CU Gestionar Departamento 113
Anexo.4 Diagramas de componentes. 120

Índice de figuras

Figura 1. Comparación entre metodologías ágiles y metodologías tradicionales (13) 16
Figura 2. Modelo de dominio 27
Figura 3 Mapa de procesos informe de recepción 29
Figura 4 Mapa de procesos solicitud de AFT 29
Figura 5 Mapa de procesos solicitud de baja de AFT 30
Figura 6 Diagrama de casos de uso del paquete seguridad 35
Figura 7 Diagrama de casos de uso del paquete departamento 35
Figura 8 Diagrama de casos de uso del paquete configuración 36
Figura 9 Diagrama de casos de uso del paquete almacén 36
Figura 10 Diagrama de casos de uso del paquete reportes 36
Figura 11 Ejemplo de utilización del patrón singleton 47
Figura 12 Diagrama de paquetes 50
Figura 13 Diagrama de clases GestionarDepartamento 51
Figura 14 Modelo de datos 53
Figura 15 Diagrama de secuencia de la funcionalidad RegistrarDepartamento 54
Figura 16 Estructura general del diagrama de componentes del sistema 56
Figura 17 Estructura del diagrama de componentes GestionarDepartamento 57
Figura 18 Diagrama de despliegue 58
Figura 19 Clase ActivoFijoDAOTesr empleada para la realización de las pruebas 65
Figura 20 Método testFindAllByDepartamneto de la clase ActivoFijoDaoTest 65
Figura 21 Resultado de las pruebas de caja blanca con JUnit 66

Introducción

En la actualidad la utilización de las Tecnologías de la Información y las Comunicaciones (TIC) se ha extendido grandemente, generando incontables beneficios para la sociedad. Su creciente y constante avance han venido a desempeñar un papel muy importante en el mundo empresarial, permitiendo recopilar y almacenar gran cantidad de información de todo tipo de procesos: industriales, empresariales, bancarios, de marketing, entre otros; en un mundo en que las tecnologías, la información y las estadísticas marcan el ritmo del progreso y las pautas de la vida. Por lo que cada día son más las empresas conscientes del hecho, de que gran parte de su éxito y estabilidad depende de la disponibilidad y seguridad de todas sus informaciones y datos, tanto en la rapidez del acceso como su fiabilidad.

El hecho de poder compartir la información existente entre todos los departamentos de una misma organización pudiendo limitar accesos determinados en función de la responsabilidad dentro de la misma, favorece una mayor comunicación que lleva consigo un incremento del rendimiento laboral, pudiendo realizar el mismo trabajo de una forma más sencilla, rápida y eficaz, traduciéndose en un ahorro sensible de costos y agilidad en las acciones.

Cuba a pesar de ser un país subdesarrollado, no ha quedado al margen de estos avances informáticos, sino que ha perfeccionado e impulsado el desarrollo del conocimiento en aras de ajustarse a los cambios que se desarrollan en la nueva era de la informatización. Las empresas cubanas, como base y sostén de la economía del país, son las encargadas de llevar el control de sus Activos Fijos Tangibles (AFT). Por otra parte el control es un elemento primordial en cualquier ámbito social y poseer información lo más exacta posible es sinónimo de poder.

El control es un elemento muy importante dentro de cualquier organización, pues es el que permite 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 Empresa Azucarera Bartolomé Masó Márquez ubicada en el Municipio con igual nombre, provincia Granma, zona de inmensa riquezas históricas y humanas, tiene como misión producir azúcar crudo de caña, alimento y otras producciones a precio competitivo para satisfacer la demanda interna. Esta empresa también tiene como objetivo generar electricidad, prestar servicios con las normas y calidad requeridas para satisfacer a los clientes internos y externos, mediante introducción de nueva

tecnología, con una adecuada gestión de recursos humanos, alcanzando alto niveles de eficiencia económica, apoyándose en la experiencia de buenos productores cañero.

En estos momentos la gestión de los AFT de dicha empresa 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 auditorías y revisiones, afectando así la calidad y rapidez del proceso. Todo esto puede traer consigo:

1. La búsqueda y recuperación de datos para la generación de reportes es lenta.
2. Gastos innecesarios o pérdidas de los medios y bienes debido a la mala planificación y asignación de los mismos.
3. Dificultad para llevar el control de las entradas y salidas de los medios en el almacén.
4. Pérdida de información asociada a la gestión de inventario de los AFT debido al deterioro de los documentos archivados.
5. El ineficiente control del nivel de acceso a los documentos perjudica la confidencialidad e integridad de los datos.

Lo antes expuesto, dificulta y demora la gestión de inventario de los AFT a los trabajadores de la empresa lo cual conlleva a la búsqueda de una solución al siguiente **problema científico**: ¿Cómo agilizar los procesos de gestión de inventario de AFT dentro de la empresa UEB CAI Bartolomé Masó?
Partiendo del problema expuesto el **objeto de estudio** se enfocará a los procesos asociados a la gestión de inventario de AFT en las entidades empresariales.
Para resolver el problema planteado con anterioridad se establece como **objetivo general:** Desarrollar un sistema informático para la gestión de inventario de AFT dentro de la empresa UEB CAI Bartolomé Masó que permita agilizar este proceso.
El **campo de acción** de la investigación es la gestión de inventario de AFT en la empresa UEB CAI Bartolomé Masó.

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

1. Elaborar la Fundamentación Teórica de la investigación.
2. Realizar el diseño de las funcionalidades del sistema de gestión de inventario de AFT.
3. Realizar la implementación de las funcionalidades del sistema de gestión de inventario de AFT.

4. Realizar pruebas a las funcionalidades del sistema de gestión de inventario de AFT.

El aporte práctico del trabajo lo constituyen:

1. Especificación de Requisitos y Modelo de Casos de Uso.
2. Diseño de los casos de prueba de los requisitos del sistema.
3. Modelos de diseño de los módulos que constituyen el sistema.
4. Implementación de los módulos Seguridad, Configuración, Reportes, Departamento y Almacén.

La investigación se basa en la siguiente **idea a defender**: El desarrollo un sistema informático permitirá agilizar los procesos de gestión de inventario de AFT que tienen lugar dentro de la empresa UEB CAI Bartolomé Masó.

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

1. Elaboración del Marco teórico de la investigación.
2. Análisis de las técnicas de captura de requisitos existentes.
3. Caracterización de las tecnologías, patrones y la arquitectura a utilizar para el desarrollo del sistema.
4. Valoración de los sistemas informáticos de gestión de inventario de AFT similares al que se desea desarrollar.
5. Análisis de las herramientas a emplear durante el desarrollo del sistema.
6. Definición de los requisitos funcionales del sistema.
7. Descripción de los requisitos funcionales y no funcionales identificados del sistema.
8. Validación de los requisitos funcionales definidos a través de revisiones con el cliente y prototipos no funcionales.
9. Modelación del diagrama de casos de uso del sistema.
10. Especificación de los casos de uso del sistema a partir de los requisitos identificados
11. Elaboración de los casos de prueba de los requisitos funcionales definidos.
12. Modelación de los diagramas de paquetes de los módulos del sistema.
13. Modelación del diagrama de clases del diseño de los módulos del sistema.

14. Modelación de los diagramas de secuencia de las funcionalidades de los módulos del sistema.
15. Modelación del diagrama de componentes del sistema.
16. Validación del diseño de las funcionalidades del sistema.
17. Obtención del modelo de datos del sistema.
18. Descripción del modelo de datos definido para el sistema.
19. Implementación de las funcionalidades del sistema.
20. Validación del sistema a través de pruebas de Caja blanca, empleando el *framework* JUnit, y pruebas de Caja negra a través de la técnica Partición de equivalencia.

Para el avance exitoso de la investigación se propone el uso de los siguientes métodos de investigación:

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 inventario, 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 de un estudio detallado de la información obtenida se hace necesario organizarla y sintetizarla para lograr una estructura adecuada.
- **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 inventario de los AFT dentro de la empresa UEB CAI Bartolomé Masó.

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 al investigador en la formación de conocimiento para valorar una solución.

- **Entrevista:** Es una conversación planificada entre el investigador y el entrevistado para obtener información. Se utiliza con el objetivo de obtener la mayor información posible sobre los requisitos que el cliente desea.

El trabajo de diploma está estructurado en los siguientes capítulos:

Capítulo 1 Fundamentación teórica: En este capítulo se muestra el resultado de la investigación bibliográfica realizada sobre el objeto de estudio. Se describen además las características del lenguaje de programación, sistema gestor de base de datos, metodología de desarrollo y las diferentes herramientas propuestas para el desarrollo del sistema.

Capítulo 2 Características del sistema: En este capítulo se presenta un estudio del funcionamiento y la información que se maneja para desarrollar el sistema, se muestran los requisitos de la propuesta de solución y se describirá en detalle sus características, así como los requisitos no funcionales que debe cumplir. Se modelan los diagramas principales para lograr un entendimiento de las funcionalidades previamente definidas.

Capítulo 3 Implementación y pruebas: En este capítulo se hace alusión a las fases de Implementación y pruebas, propias de la metodología de desarrollo utilizada para guiar la elaboración del sistema que se propone. Se exponen además los artefactos generados durante el transcurso de la misma.

Capítulo.1 Fundamentación teórica

1.1. Introducción

La tecnología en la actualidad se encuentra en constante evolución, por lo que se hace necesario tener un conocimiento avanzado y actualizado de esta a la hora de desarrollar aplicaciones informáticas. Su incidencia ha propiciado innumerables ventajas como el almacenamiento y centralización de grandes volúmenes de información, rapidez en la obtención de resultados, facilidades para encontrar información adecuada y/o actualizada, ahorro de tiempo y dinero, posibilidades para procesar datos; todo esto hace que los sistemas informáticos constituyan un mecanismo de apoyo para la toma de decisiones. Teniendo en cuenta lo antes planteado se ha decidido en el presente capítulo abordar sobre las herramientas, lenguajes, tecnologías y arquitectura utilizados, se realiza una síntesis acerca de algunos sistemas, que permiten el control de inventario y se exponen los principales conceptos y aspectos relacionados con el tema de investigación.

1.2. Conceptos fundamentales

Inventario

El inventario consiste en verificar físicamente los medios con que cuenta la organización. Su finalidad es llevar a cabo un registro de la existencia, cantidad, características, condiciones de uso, valor de los medios y las personas responsables de su manejo. Al hacer un inventario hay que tener el mayor cuidado posible, para evitar repeticiones, para que no se incluyan materiales o mercancías que no correspondan. Un inventario es además una relación de los activos circulantes que posee la entidad en un momento dado y que pueden estar destinados para ser vendidos o como insumos en el proceso productivo. (1)

Gestión de inventario

Se puede definir como la administración de existencias de todo producto o artículo que es utilizado para la comercialización dentro de una organización. Es decir, todo lo relativo al control y manejo de las existencias de determinados bienes, en la cual se aplican métodos y estrategias que pueden hacer rentable y productivo la tenencia de estos bienes y a la vez sirve para evaluar los procedimientos de entradas y salidas de dicho producto. (2)

Sistema de gestión

Un sistema de gestión ayuda a lograr las metas y objetivos de una organización mediante una serie de estrategias, que incluyen la optimización de procesos, el enfoque centrado en la gestión y el pensamiento disciplinado. Por tanto el sistema de gestión es un conjunto de etapas unidas en un proceso continuo, que deja trabajar ordenadamente una idea hasta lograr mejoras y su continuidad.

La implementación de un sistema de gestión eficaz en una organización puede ayudar a esta en: gestionar los riesgos sociales, medioambientales y financieros, mejora en la efectividad operativa, reducción de costos, aumento en la satisfacción de clientes y partes interesadas, protege la marca y la reputación, logra mejoras continuas, potencia la innovación, eliminar las barreras al comercio y aporta claridad al mercado. (3)

Activo fijo

El activo fijo es el conjunto de bienes de naturaleza relativamente permanente, adquiridos, desarrollados o construidos por administración propia o por contrato, necesarios para el cumplimiento de objetivos institucionales. (4)

Activos Fijos Tangibles (AFT)

Los activos fijos tangibles (AFT) lo constituyen bienes con todas las características anotadas anteriormente para los activos fijos, los cuales están connotados por tener materialidad, presencia física la que se puede apreciar con los sentidos. (5) Maquinaria, mobiliario.

Luego de haber definido conceptos fundamentales abordados en diferentes bibliografías, el autor del presente trabajo define, que un sistema de gestión de inventario de ATF es un conjunto de métodos y estrategias, los cuales permiten administrar la existencia física de los bienes de naturaleza permanente que serán almacenados en la empresa para hacer rentable y productivo la tenencia de estos bienes y para evaluar los procedimientos de sus entradas y salidas.

1.3. Análisis de sistemas informáticos

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

análisis de algunos sistemas de gestión de inventario tanto nacionales como internacionales que servirán de apoyo en el desarrollo de la investigación.

1.3.1. Sistemas internacionales

Herramienta Sorolla2. (España)

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 individualizada de todos los bienes inventariables, 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. Queda excluido el inmovilizado financiero.

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. (6)

Este sistema tiene como requisitos de Software:

- 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 ó superior
  - Firefox 3.5 ó superior.
- Certificado electrónico para el acceso al sistema.
- Visor de PDF para la visualización de los informes generados.
- Requisitos para la firma electrónica mediante certificado

Este sistema es una aplicación web, y los requisitos de software que necesitan constituyen una limitante para su uso en nuestro país. Primeramente necesita de la conexión a la red SARA.

[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.

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 Troncal. De esta manera queda asegurada la confidencialidad de la información que viaja a través de la red SARA. (7)

Se requiere además de un certificado electrónico para poder acceder al sistema, lo cual implica un alto costo por concepto de licencia, que la empresa no puede pagar.

CS – Almacenes (España)

CS-Almacenes es una plataforma que permite controlar el inventario de almacenes. El programa se basa en ir registrando las entradas y salidas de manera que siempre conocen qué cantidad hay de un producto en tiempo real. Controla productos en ilimitados almacenes, llevando a cabo una realización de ajustes manuales si se producen inventarios físicos y conteos, así como realizar traspasos entre almacenes. (8) Es una aplicación fácil de utilizar debido a que está dirigida a profesionales que no necesitan el control de trazabilidad compleja, si no que permite conocer en todo momento su existencia.

La desventaja que presenta esta plataforma es que no permite la generación de estadísticas para determinados productos ni el conocimiento de la localización de estos. Es una plataforma privativa.

1.3.2. Sistemas nacionales

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 y en lo adelante será introducido en más de dos mil 500 unidades presupuestadas. Este sistema integrado cuenta con un conjunto de 12 módulos entre los que se encuentran: configuración y seguridad, contabilidad general y de gastos, costos y procesos, análisis económico empresarial, control de activos fijos, finanzas y caja, planificación y presupuestos, control de inventarios, pago de salario, paquete de gestión, contratación, facturación.
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. (9)

A pesar de los grandes beneficios que trae consigo la utilización del Versat, este software no resulta una solución factible para la empresa que se analiza, 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 empresa 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.

ConDor

El sistema de control de inventarios ConDor está concebido para el control de las entradas o salidas de uno o varios almacenes y además para la contabilización de los movimientos que se realizan con los productos almacenados. Está generalizado en más de mil entidades del territorio nacional siendo los organismos más representativos el MITRANS y el Instituto nacional de recursos hidráulicos (INRH). Cuenta con una suite integrada por siete módulos: contabilidad general, activos fijos, inventario, nómina/prenómina, disponibilidad financiera, condexce y efectos. Actualmente este software se encuentra implementado para el sistema operativo *windows* en cualquiera de sus versiones. La versión del módulo existente presenta la característica que está implementado en un entorno monousuario, lo que ha sido señalado por parte de los clientes como una significativa desventaja. También este sistema presenta la característica que cada módulo tiene una base de datos diferente, lo que trae redundancia en la información. (10)

Inventario en la UCI. Assets

El sistema de gestión integral (ASSETS) es un sistema multiusuario que se monta 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. (11)

Entre las desventajas que presenta este sistema es que controla los medios por áreas de inmuebles y es de necesidad para la empresa controlar los medios por los diferentes locales existentes dentro de cada área, y agregarles diferentes características a los medios y locales existentes.

1.3.3. Valoración crítica sobre los sistemas 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 nuestro país pues 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 nuestra como lo es la capitalista que tiene un modelo de gestión y de procesos muy diferente a las empresas cubanas.

- Los sistemas implantados en nuestro país necesitan de grandes prestaciones y de integraciones entre módulos lo cual no es aplicable, pues es una empresa pequeña que no cuenta con la tecnología y equipamientos necesarios para su funcionamiento. Se manejan 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 los departamentos.

1.4. Metodología de desarrollo de software

La metodología no es más que un conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar un nuevo software. Compuesta por tareas (actividades en que se dividen los procesos), procedimientos (define la forma de ejecutar la tarea),

técnica (herramienta utilizada para aplicar un procedimiento), herramientas (herramientas de software que automatizan la aplicación) y el producto (resultado de cada etapa). (12)

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. (12)

Metodologías tradicionales

Las metodologías tradicionales centran su atención en llevar una documentación absoluta de todo el proyecto, están focalizadas en documentación, planificación y procesos. Se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado. Para el desarrollo de los productos generalmente existe un contrato prefijado y el cliente interactúa con el equipo de desarrollo mediante reuniones. Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. (12)

Metodología ágil vs metodología tradicional

En la siguiente tabla se muestra una comparación entre las metodologías agiles y las tradicionales teniendo en cuenta diferentes características. Estas diferencias que afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

Figura 1. Comparación entre metodologías ágiles y metodologías tradicionales (13)

¿Por qué usar metodología ágil?

Teniendo en cuenta los aspectos de la tabla anterior, se puede decir que las metodologías tradicionales presentan los siguientes problemas a la hora de abordar proyectos: (14)

- Existen unas costosas fases previas de especificación de requisitos, análisis y diseño. La corrección durante el desarrollo de errores introducidos en estas fases será costosa, es decir, se pierde flexibilidad ante los cambios.
- El proceso de desarrollo está reducido por documentos firmados.
- El desarrollo es más lento. Es difícil para los desarrolladores entender un sistema complejo en su globalidad.

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación. Dividir el trabajo en módulos abordables minimiza los fallos y el coste. Las metodologías ágiles presentan diversas ventajas, entre las que se destacan:

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo
- Entrega continua y en plazos breves de software funcional
- Trabajo conjunto entre el cliente y el equipo de desarrollo
- Importancia de la simplicidad, eliminado el trabajo innecesario
- Atención continua a la excelencia técnica y al buen diseño
- Mejora continua de los procesos y el equipo de desarrollo

eXtreme programing

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Las características esenciales de XP están organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas. (14)

Scrum

Scrum es un método iterativo e incremental que enfatiza prácticas y valores de *project management* por sobre las demás disciplinas del desarrollo. La intención de Scrum es la de maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de metodologías ágiles, siendo combinado con otras – como XP – para completar sus carencias. Cabe mencionar que Scrum no propone el uso de ninguna práctica de desarrollo en particular; sin embargo, es habitual emplearlo como un *framework* ágil de administración de proyectos que puede ser combinado con cualquiera de las metodologías ágiles. La metodología resulta sencilla definiendo algunos roles y artefactos que contribuyen a tener un proceso que maximiza el *feedback* para mitigar cualquier riesgo que pueda presentarse. (14)

OpenUP

OpenUP (proceso unificado abierto) es una versión más ágil de lo que es el *Rational Unified Process*

(RUP). Es un proceso unificado que aplica propuestas iterativas e incrementales dentro del ciclo de vida, tratando de ser manejable en relación con RUP. Plantea que se debe tener un software ya funcional, o lo que es lo mismo, un proyecto ejecutable en poco tiempo, para esto se debe utilizar sólo los procesos que sean necesarios, sin demasiados artefactos y sobre todo que el proyecto debe acoplarse a las necesidades del usuario, pudiendo ser éste modificado, mejorado y extendido. OpenUP tiene dos ventajas importantes, este tipo de método disminuye los riesgos y además puede utilizarse tanto en proyectos pequeños como en proyectos grandes, aunque está concebida para proyectos pequeños. Si se maneja con cuidado y con profesionalismo se puede desarrollar un software de gran calidad, a pesar de que se le diseñe en poco tiempo y con poca documentación. Se utiliza preferentemente en proyectos pequeños, dígase proyectos de 3 a 6 personas. OpenUP contiene las características esenciales de RUP, que incluye el desarrollo iterativo de casos de uso, además de proporcionar un acercamiento a la arquitectura central del sistema. El resultado es un proceso mucho más simple que sigue fielmente los principios de RUP. (15)

Según la *Eclipse Foundation* (15), OpenUP se organiza en dos dimensiones diferentes: método y proceso.

- Método: Los roles, las tareas y los artefactos están definidos, sin tener en cuenta cómo son aplicados en el ciclo de vida del proyecto.
- Proceso: Es cuando los elementos del proceso son aplicados en el sentido conductual, donde el mismo brinde los roles, las tareas y los artefactos. Pueden crearse ciclos de vida diferentes para proyectos diferentes.

¿Por qué OpenUP?

Luego del estudio realizado sobre diversas metodologías ágiles se determinó que se utilizará como metodología de desarrollo OpenUP. Entre las causas que hicieron posible su elección se destaca que preserva la esencia de RUP y al igual que él, posee un modelo de desarrollo iterativo e incremental, pero a diferencia de este es un proceso más ligero. Permite micro incrementos en breves períodos de tiempo, lo que facilita ir creando liberaciones del producto mientras se desarrolla y efectuar modificaciones a los requisitos sin altos costos en cuanto a tiempo, precio e implementación. Es apropiado para proyectos pequeños y de bajos recursos (como es el caso). Permite detectar errores tempranos a través de un ciclo iterativo. No es tan exhaustiva en cuanto a la generación de documentos como RUP y debido a que es una metodología ágil, tiene un enfoque centrado en el cliente con iteraciones muy cortas.

1.5. Lenguaje unificado de modelado (UML)

El lenguaje de modelamiento unificado es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos. Estimula el crecimiento de herramientas orientadas a objetos. Integra las mejores prácticas para la modelación y puede soportar todos los lenguajes de programación. (16)

De forma general las principales características son:

- Lenguaje unificado para la modelación de sistemas
- Tecnología orientada a objetos
- El cliente participa en todas las etapas del proyecto
- Corrección de errores viables en todas las etapas

1.6. Herramienta de modelado

Actualmente existen varias herramientas que se utilizan para el modelado UML, las cuales son conocidas como CASE (una sigla, que corresponde a las iniciales de *Computer Aided Software Engineering*; y en su traducción al Español significa Ingeniería de software asistida por computación. Estas herramientas permiten automatizar o apoyar una o más fases del proceso de desarrollo de software así como organizar y manejar la información de un proyecto informático. Las herramientas que se tendrán en cuenta para la realización de este trabajo son Visual Paradigm y Rational Rose.

Rational Rose

Racional Rose es un lenguaje unificado de modelado (UML) herramienta de diseño de software orientado a objetos destinados para el modelado visual y componentes para la construcción de aplicaciones de software de nivel empresarial.

El software permite acelerar el desarrollo de estas aplicaciones con código generado a partir de modelos visuales mediante el lenguaje UML (*Unified Modeling Language*).

Rational Rose Enterprise ofrece una herramienta y un lenguaje de modelado común para simplificar el entorno de trabajo y permitir una creación más rápida de software de calidad.

Modelado de las aplicaciones más habituales: proporciona prestaciones de modelado visual para desarrollar muchos tipos de aplicaciones de software.

Desarrollo de aplicaciones para la web: contiene herramientas web y XML para el modelado de aplicaciones web.

Integración del diseño de aplicaciones con el desarrollo: unifica el equipo del proyecto proporcionando una ejecución y una notación de modelos UML comunes. (17)

Visual Paradigm

Es una herramienta CASE, utilizada en un ambiente de software libre, debido a la posibilidad de ejecutarse sobre cualquier sistema operativo, por lo que se convierte en una herramienta multiplataforma, permite crear diferentes tipos de diagramas en un ambiente totalmente visual. Soporta además el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite graficar todos los tipos de diagramas de clases, generar documentación, generar código desde diagramas y código inverso. (18)

Beneficio

- Navegación intuitiva entre el modelo visual y el código
- Poderosa herramienta de generación de PDF/HTML a partir de diagramas UML
- Sincronización entre el código fuente y el modelo en tiempo real o bajo demanda
- Entorno visual de modelado superior
- Soporte para toda la notación UML
- Análisis de textos
- Sofisticados y automáticos diagramas de capas
- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación
- Capacidades de ingeniería directa (versión profesional) e inversa
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de integrarse en los principales IDEs
- Disponibilidad en múltiples plataformas

¿Por qué utilizar Visual Paradigm? (v 8.0)

Visual Paradigm para UML facilita la generación de un entorno de modelados visuales en el que se reúnen todas las necesidades tanto de software y tecnología, como de las necesidades de comunicación. Es apoyado por un conjunto de idiomas tanto en la generación del código como en la ingeniería inversa. Facilita la diagramación visual y el diseño de los proyectos. Fácil de usar y rico en características, compatible con UML 2.1 y anteriores. Intercambia diagramas UML y modelos con otras herramientas, usando representaciones industriales comunes. Respalda el ciclo de vida del software del análisis al diseño, y del diseño a la implementación. (19)

Visual Paradigm se puede integrar con el IDE eclipse lo que permite la realización de ingeniería inversa a Java y la generación de documentos. Esta última característica, unida a los beneficios mencionados anteriormete es lo que ha favorecido su elección como herramienta de modelado del sistema.

1.7. Herramientas de desarrollo

1.7.1. Entorno de desarrollo integrado (IDE)

Un ambiente de desarrollo integrado o IDE, es un programa compuesto por un conjunto de herramientas que proveen facilidades a los programadores para agilizar el proceso de desarrollo de software. (20)

Eclipse (v4.2)

Eclipse es una comunidad de código abierto cuyos proyectos se centran en la creación de una plataforma de desarrollo extensible, tiempos de ejecución y los marcos de aplicación para la construcción, despliegue y gestión de software en el ciclo de vida completo del software. Es un entorno de desarrollo integrado de código abierto multiplataforma. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador que se entrega como parte de eclipse (y que son usados también para desarrollar el mismo eclipse).

Eclipse dispone de un editor de texto con resaltador de sintaxis. La compilación es en tiempo real. Tiene pruebas unitarias con JUnit, control de versiones con CVS, integración asistentes (*wizards*) para creación de proyectos, clases, tests, etc., y refactorización. Asimismo, a través de "*plugins*" libremente disponibles es posible añadir control de versiones con subversion e integración con hibernate. (21)

1.7.2. Herramienta para el diseño de los reportes

iReport (v 5.1)

La herramienta iReport es un constructor/diseñador de informes visual de código libre para JasperReports y JasperReports Server. Permite crear diseños que contienen gráficos, imágenes, subinformes, tablas de contingencia y que los usuarios corrijan visualmente informes complejos. Los datos para imprimir pueden ser recuperados por varios caminos incluso múltiples uniones JDBC, TableModels, JavaBeans, XML, etc.Los informes pueden ser publicados en PDF, RTF, XML, XLS, CSV, HTML, XHTML, texto, DOCX, u OpenOffice. (22)

Entre las características que posee esta herramienta y que permitiron su utilización en el sistema se encuentran las siguientes:

- 100% escrito en JAVA y además OPENSOURCE y gratuito
- Maneja el 98% de las etiquetas de JasperReports
- Soporta JDBC
- Incluye *Wizard's* (asistentes) para crear automáticamente informes y para generar subreportes

El uso de esta herramienta permitió la creación de los reportes que se imprimen y forman parte del conjunto de salidas del sistema.

1.8. Sistema gestor de base de datos

El propósito general de un sistema de gestión de bases de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para un buen manejo de datos.

PostgreSQL (v 9.1)

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Algunas de sus características esenciales son: (23)

- Es una base de datos 100% ACID (acrónimo de *atomicity, consistency, isolation and durability*: atomicidad, consistencia, aislamiento y durabilidad en español)
- Múltiples métodos de autentificación
- Acceso encriptado via SSL
- Completa documentación
- Licencia BSD[2] (*Berkeley Software Distribution*)
- Multiplataforma
- Numerosos tipos de datos y posibilidad de definir nuevos tipos

1.9. Lenguaje de programación

Como lenguaje de programación se utiliza java.

Java es un lenguaje de desarrollo de propósito general, y como tal es válido para realizar todo tipo de aplicaciones profesionales. Una de las características más importantes es que los programas "ejecutables", creados por el compilador de java, son independientes de la arquitectura. Se ejecutan indistintamente en una gran variedad de equipos con diferentes microprocesadores y sistemas operativos. (24)

Características

- Su sintaxis es similar a C y C++
- No hay punteros (lo que le hace más seguro)
- Totalmente orientado a objetos
- Muy preparado para aplicaciones TCP/IP
- Implementa excepciones de forma nativa
- Es interpretado (lo que acelera su ejecución remota, aunque provoca que las aplicaciones Java se ejecuten más lentamente que las C++ en un ordenador local).
- Permite multihilos

1.10. *Frameworks*

Hibernate (v 3.0)

[2] BSD: es la licencia de software otorgada principalmente para los sistemas BSD (*Berkeley Software Distribution*). Es una licencia de software libre permisiva. Esta licencia tiene menos restricciones en comparación con otras estando muy cercana al dominio público

Hibernate es una herramienta de Mapeo Objeto-Relacional (ORM) para la plataforma java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible. Ofrece también un lenguaje de consulta de datos llamado HQL (*Hibernate Query Language*), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria") que permite crear consultas mediante la manipulación de los criterios de los objetos en tiempo de ejecución. Hibernate genera las sentencias SQL y libera al desarrollador del manejo manual de los datos que resultan de la ejecución de dichas sentencias, manteniendo la portabilidad entre todos los motores de bases de datos con un ligero incremento en el tiempo de ejecución. (25)

Swing

Swing es la parte de *JavaTM Foundation Classes* (JFC) que implementa un conjunto de componentes de interfaz gráfica de usuario con una mirada de acoplamiento activo y se siente. Es implementado en su totalidad en el lenguaje de programación Java, y se basa en el marco de la interfaz de usuario Lightweight JDKTM 1.1. Permite diseñar un único conjunto de componentes GUI (guía de interfaz de usuario, por sus siglas en inglés) que puede tener automáticamente la apariencia de cualquier plataforma de sistema operativo (*Microsoft Windows*, SolarisTM, Macintosh). (26)

Es un *framework* MVC (acrónimo de modelo vista controladora por sus siglas en inglés) para desarrollar interfaces gráficas para Java con independencia de la plataforma. Sigue un simple modelo de programación por hilos.

Un mismo objeto se ve de esas tres formas: (24)

- Modelo. Se refiere al modelo de datos que utiliza el objeto. Es la información que se manipula mediante el objeto Swing
- Vista. Es cómo se muestra el objeto en la pantalla
- Controlador. Es lo que define el comportamiento del objeto

1.11. Librerías

Jasper Reports (v 4.1)

La biblioteca Jasper Reports es un motor de informes de código abierto. Está escrito completamente en Java y es capaz de utilizar los datos procedentes de cualquier tipo de fuente de datos y producir documentos que se pueden ver, imprimir o exportar en una variedad de formatos de documentos incluyendo HTML, PDF, Excel, Open Office y Word. Permite representar información diversa de distintas maneras, pudiéndose incluir en los reportes texto, tablas, imágenes y gráficos. (22)

Esta librería se utiliza en el sistema para mostrar información estadística que apoye la toma de decisiones de la empresa y para la generación de documentos oficiales que se emiten dentro de esta tales como los submayores y el control de inventario.

1.12. Conclusiones parciales

Con la realización de este capítulo se puede afirmar que el mundo está en un proceso de constante cambio donde las tecnologías de la información están tomando un mayor auge y con ellas las empresas aumentan la producción y mejoran la gestión de sus procesos. Dentro de las empresas se hace necesario mejorar los procesos que se desarrollan por la importancia que estos revisten. Tal es el caso de la gestión de inventario. En el desarrollo del capítulo se realiza 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 empresa CAI Bartolomé Masó Márquez. Por tanto se decide realizar un software que cumpla con las necesidades de los clientes y que además se desarrolle utilizando la metodología de desarrollo OpenUP, la herramienta Visual Paradigm para modelar los procesos, el IDE Eclipse, el lenguaje de programación Java y el sistema gestor de base de datos PostgreSQL. Todas las herramientas fueron explicadas en este capítulo, incluyendo las características que poseen y que permitieron su selección para el desarrollo del presente trabajo.

Capítulo.2 Análisis y diseño

2.1. Introducción

En el presente capítulo se mencionan cada uno de los requisitos funcionales que presenta el sistema y se describen los no funcionales. Además, se enumeran y describen los actores que intervienen en el sistema, se propone una solución mediante diagramas de casos de uso y una breve descripción de la solución propuesta. Además se expone un diagrama de paquetes en el cual se agrupan casos de uso y actores del sistema en dependencia de su funcionalidad.

2.2. Propuesta de solución

Con el objetivo de obtener un mejor control de los medios y bienes con que cuenta la empresa CAI Bartolomé Masó Márquez, se ha decidido desarrollar un sistema informático que automatice los procesos de gestión de inventario que se realizan de forma manual en estos momentos. El mismo brindará facilidades para almacenar y manejar la información referente a los medios y bienes de la empresa.

Para el trabajo con el sistema se han definido tres roles fundamentales los cuales se detallan a continuación: administrador del sistema, el cual podrá crear usuarios y asignarles permisos de acuerdo al rol que ocupen, además será el encargado de la seguridad del sistema. Un grupo de usuarios que se comportarán como jefe de departamento y tendrán acceso a determinadas funcionalidades, aplicadas solo a los departamentos de los cuales ellos sean jefes, cuyas funcionalidades serán asignadas por el administrador del sistema en su debido momento. También se cuenta con el rol jefe de almacén que tiene permisos para realizar las operaciones en el almacén. La información de todos los usuarios que interactúen con el sistema se registrará y controlará en la base de datos.

El sistema permite la configuración de nomencladores, tales como categorías, departamentos, conceptos para bajas de AFT y conceptos para entradas de AFT. Se tiene control de las operaciones realizadas en el almacén mediante la gestión de informes de recepción, los vales de entrega y los movimientos de AFT. Para garantizar la seguridad del sistema se registran los usuarios que tienen acceso a este, asignándole un rol determinado de acuerdo a las funciones que realiza dentro de la empresa y a estos roles se asignarán los permisos. Cada usuario debe contar con una contraseña que será personal e intransferible.

La solución que se propone, de manera general debe permitir que se agilicen los procesos de la empresa relacionados con el inventario de AFT.

2.3. Objetivos estratégicos de la organización

Los objetivos estratégicos de la Empresa UEB CAI Bartolomé Masó Márquez son:

- Producir azúcar de alta calidad para satisfacer la demanda del mercado internacional.
- Producir derivados del azúcar (miel, urea bagacillo, miel final, cachaza y bagazo) que serán utilizadas como materia prima para obtener un elevado número de productos de alta demanda y valor agregado (tableros aglomerados, papeles, cartones, derivados de celulosa, productos químicos, alcoholes de distintos usos, proteína unicelular para uso forrajero, aminoácidos, ácidos orgánicos).
- Producir energía eléctrica sincronizada con el Sistema Electroenergético Nacional (SEN), para su comercialización a la red pública o a eventuales clientes locales.

2.4. Modelo de dominio

En la siguiente figura se representa los conceptos relevantes y sus relaciones en la descripción de solución que se propone.

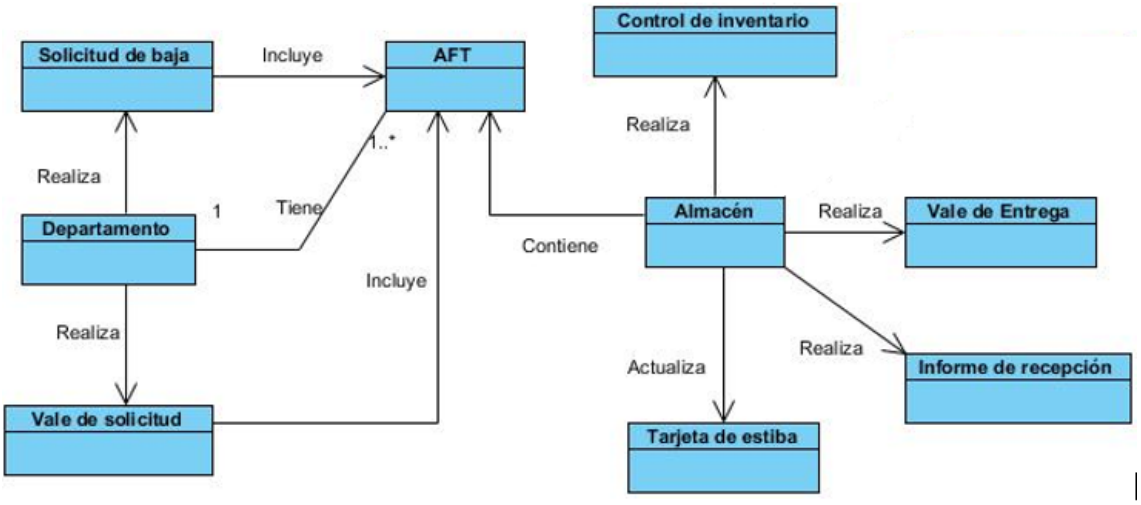


Figura 2. Modelo de dominio

2.4.1. Descripción de las clases del modelo de dominio

AFT: Activo fijo tangible (AFT) son bienes de naturaleza permanente con que cuenta la empresa.

Almacén: Local o depósito donde se almacenan los AFT.

Departamento: Sección en que está dividida la empresa. Cuenta con un jefe, que es como tal, el máximo responsable del inventario en el departamento.

Control de inventario: Acción que se realizar para verificar físicamente los medios con que cuenta la empresa. Su finalidad es llevar a cabo un registro de la existencia, cantidad, características, condiciones de uso, valor de los medios y las personas responsables de su manejo.

Informe de recepción: Documento donde se controla la entrada de AFT al almacén.

Solicitud de baja: Planilla que se llena en los departamentos y se aprueba en el almacén para darle salida a un determinado AFT.

Tarjeta de estiba: Soporte físico que registra la información de los AFT que componen el Inventario y a la vez constituyen su Control Interno, donde se registran todos los movimientos que ocurren con dicho AFT.

Vale de entrega: Planilla en el que se materializa la entrega de un determinado AFT a un departamento. Se emite desde el almacén y se firma en los departamentos.

Vale de solicitud: Planilla que se llena en los departamentos mediante el cual se pide un determinado AFT al almacén**.**

2.5. Procesos de negocio

En el proceso de negocio se realizan un conjunto de tareas relacionadas lógicamente llevadas a cabo para lograr un resultado de negocio definido. Cada uno tiene sus entradas, funciones y salidas. Las entradas son requisitos que deben tenerse antes de que una función pueda ser aplicada, esto procesos se realizan para generar productos y servicios. En el sistema propuesto se evidencia tres procesos fundamentales: informe de recepción, solicitud de AFT y solicitud de baja de AFT. A continuación se representan cada uno de estos procesos y posteriormente se realiza una breve descripción de los mismos.

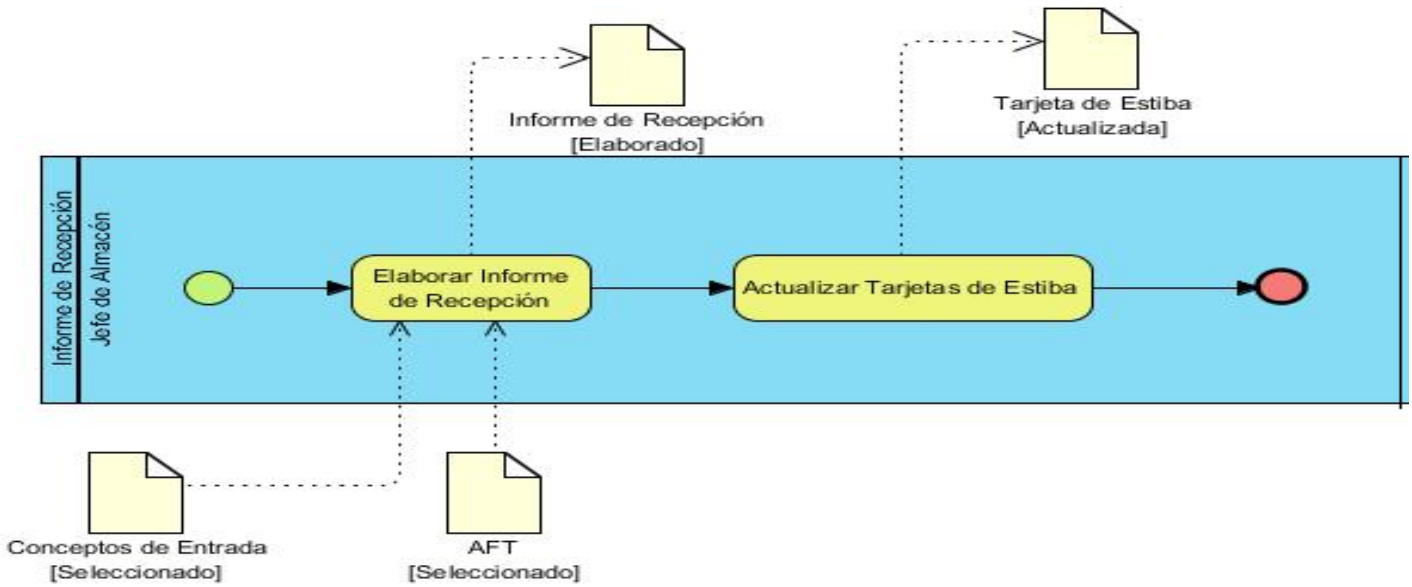


Figura 3 Mapa de procesos informe de recepción

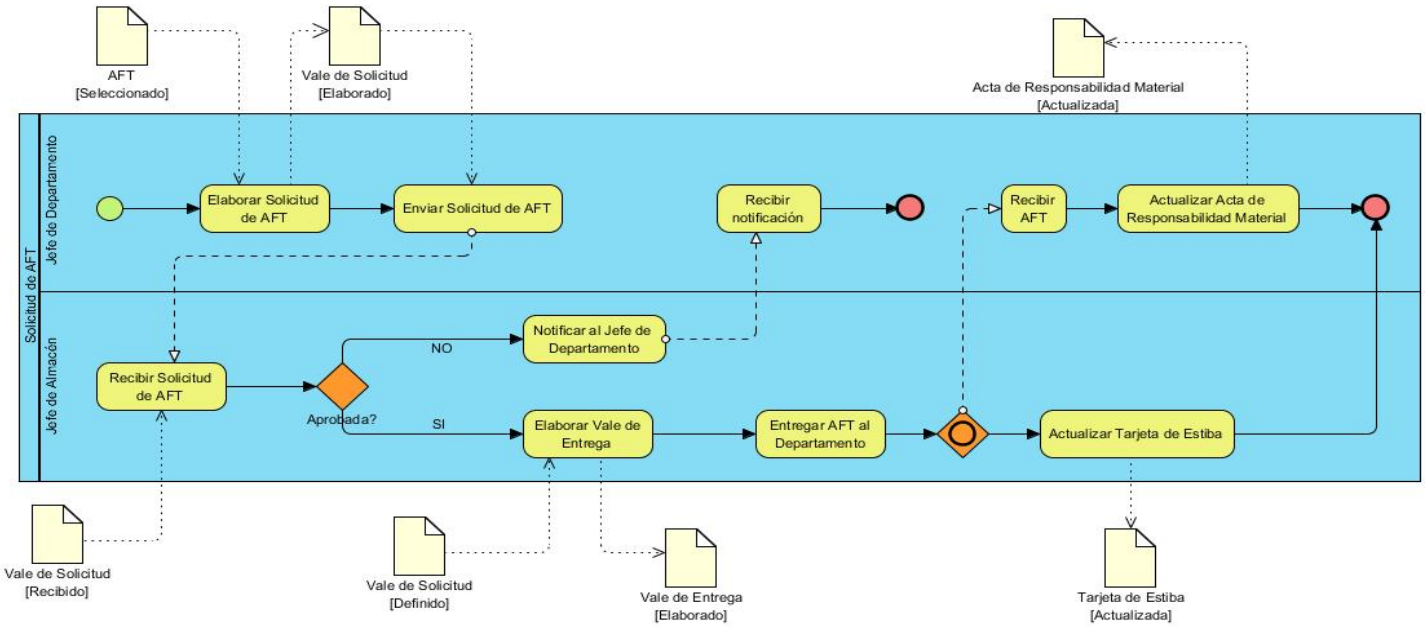


Figura 4 Mapa de procesos solicitud de AFT

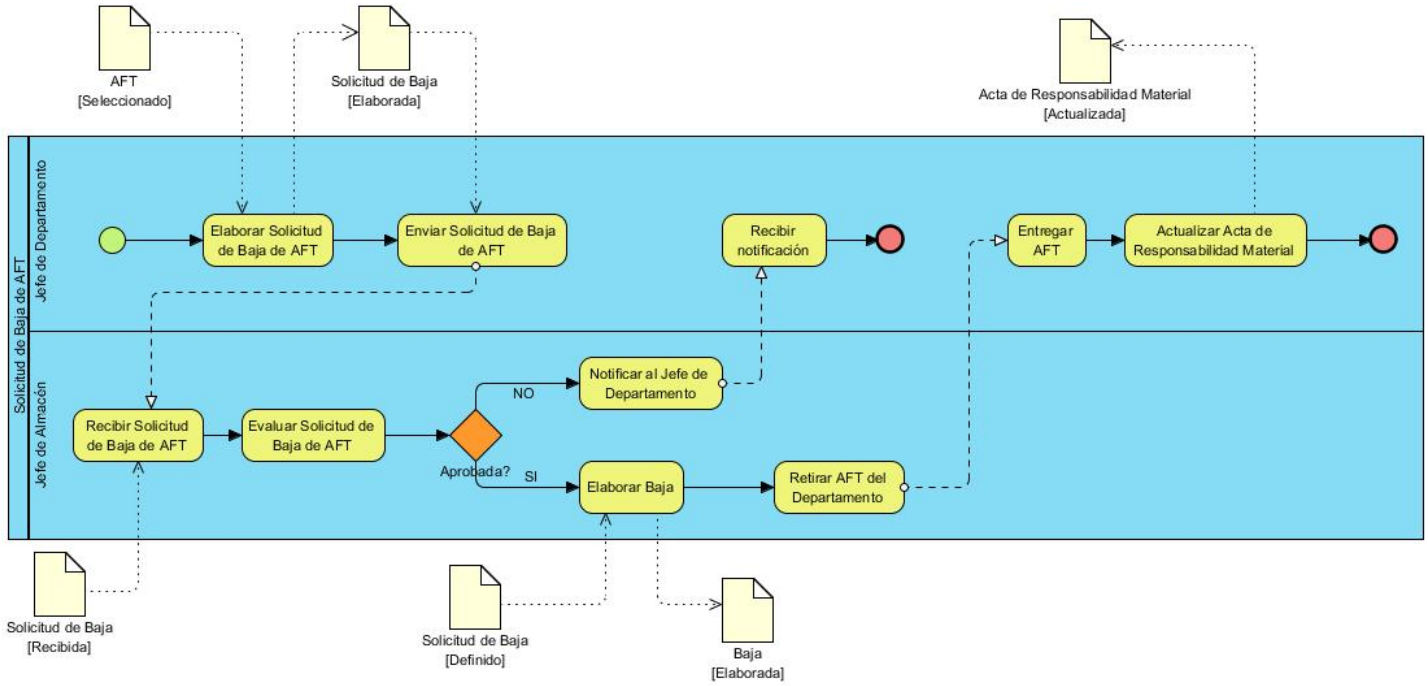


Figura 5 Mapa de procesos solicitud de baja de AFT

2.5.1. Descripción de los procesos de negocio.

Informe de recepción: El proceso comienza cuando llega un AFT al almacén. El almacenero registra el concepto de entrada de este activo. Luego se elabora el informe de recepción, con los datos requeridos para el llenado de este informe. Finalmente se realiza la actualización de la tarjeta de estiba del correspondiente AFT, culminando de esta manera el proceso.

Solicitud de AFT: El jefe de departamento llena la planilla de solicitud de un determinado AFT. Luego envía dicha planilla al jefe de almacén, el cual tiene la responsabilidad del control de los AFT en la empresa. El jefe de almacén recibe la solicitud y la evalúa para su aprobación. Si la solicitud no es aprobada se notifica al jefe de departamento. En caso positivo el jefe de almacén elabora el vale de entrega, lleva el AFT al departamento y actualiza la tarjeta de estiba. Por su parte el jefe de departamento recibe el AFT y actualiza el acta de responsabilidad material, completándose así el proceso.

Solicitud de baja de AFT: El jefe de departamento llena la planilla de solicitud de baja de un determinado AFT. Luego envía dicha planilla al jefe de almacén. El jefe de almacén recibe la solicitud y revisa el estado del medio para poder aprobar o no la solicitud de baja. Si la solicitud no es aprobada se notifica al jefe de departamento. En caso positivo el jefe de almacén elabora la panilla de baja de AFT, retira el AFT del

departamento. Por su parte el jefe de departamento actualiza el acta de responsabilidad material, completándose así el proceso.

2.6. Requisitos funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir. Son considerados característica requerida del sistema que expresa una capacidad de acción del mismo una funcionalidad; generalmente expresada en una declaración en forma verbal.

Son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (27)

A continuación se mencionan los requisitos funcionales que se utilizan en la solución propuesta. Estos requisitos están asociados a diferentes paquetes, para que sea de mejor comprensión y más adelante se explica en cada uno de ellos:

Paquete Seguridad

1. RF1 Gestionar datos para un rol de usuario
    - RF1.1 Adicionar datos para un nuevo rol de usuario
    - RF1.2 Modificar datos existentes para un rol de usuario
    - RF1.3 Consultar datos existentes para un rol de usuario
    - RF1.4 Eliminar datos existentes para un rol de usuario
2. RF2 Gestionar datos para un usuario
    - RF2.1 Adicionar datos de un nuevo usuario
    - RF1.2 Modificar datos existentes de un usuario
    - RF1.3 Consultar datos existentes de usuario
    - RF1.4 Eliminar un usuario existente
3. RF3 Consultar acciones desarrolladas en el sistema por los diferentes usuario denominadas trazas
4. RF4 Introducir datos de un usuario para poder ser autenticado
5. RF5 Introducir nueva contraseña del usuario para poder ser modificada

Paquete Configuración (Nomencladores)

1. RF6 Gestionar datos para una categoría
    - RF6.1 Introducir datos para crear una nueva categoría
    - RF6.2 Modificar datos existentes para una categoría
    - RF6.3 Eliminar una categoría existente

2. RF7 Gestionar datos de concepto para baja de AFT
   - RF7.1 Introducir datos para crear un nuevo concepto para baja de AFT
   - RF7.2 Modificar datos existentes para un concepto de baja de AFT
   - RF7.3 Eliminar un concepto para baja de AFT existente
3. RF8 Gestionar datos de concepto para entrada de un AFT
   - RF8.1 Introducir datos para crear un nuevo concepto para entrada de AFT
   - RF8.2 Modificar datos existentes para un concepto de entrada de AFT
   - RF8.3 Eliminar un concepto para entrada de AFT existente
4. RF9 Gestionar datos para un AFT
   - RF9.1 Introducir datos para crear un nuevo AFT
   - RF9.2 Modificar datos existentes de AFT
   - RF9.3 Eliminar un AFT existente
5. RF10 Gestionar datos para un departamento
   - RF10.1 Introducir datos para crear un nuevo departamento
   - RF10.2 Modificar datos existentes de un departamento
   - RF10.3 Eliminar datos existentes de un departamento
6. RF11 Gestionar datos para un centro de costo
   - RF11.1 Introducir datos para crear un nuevo centro de costo
   - RF11.2 Modificar datos existentes de un centro de costo
   - RF11.3 Eliminar un centro de costo existente

Paquete Almacén

1. RF12 Gestionar datos para un informe de recepción
   - RF12.1 Introducir datos para crear un nuevo informe de recepción
   - RF12.2 Consultar datos de un informe de recepción existente
2. RF13 Gestionar datos para un vale de entrega
   - RF13.1 Introducir datos para crear un vale de entrega
   - RF13.2 Consultar datos de un vale de entrega existente
3. RF14 Gestionar datos para movimientos de AFT
   - RF14.1 Introducir datos para realizar un movimiento de AFT
   - RF14.2 Consultar datos de un movimiento de AFT realizado
4. RF15 Gestionar datos para la solicitud de baja realizada
   - RF15.1 Consultar datos de la solicitud de baja realizada
   - RF15.2 Modificar datos de la solicitud de baja realizada

- RF15.3 Eliminar solicitud de baja realizada

Paquete Departamento

1. RF16 Gestionar datos para una solicitud de baja de AFT
   - RF16.1 Introducir datos para crear una nueva solicitud de baja de AFT
   - RF16.2 Consultar datos de una solicitud de baja de AFT existente
   - RF16.3 Modificar datos existentes de una solicitud de baja de AFT
   - RF16.4 Eliminar una solicitud de baja de AFT
2. RF17 Confirmar el recibo del AFT entregado por el almacén
3. RF18 Gestionar datos para un vale de solicitud
   - RF18.1 Introducir datos para crear un nuevo vale de solicitud
   - RF18.2 Consultar datos de un vale de solicitud existente
   - RF18.3 Modificar datos existentes de un vale de solicitud
   - RF18.4 Eliminar un vale de solicitud
4. RF19 Verificar los datos para actualizar el estado de los AFTs

Paquete Reportes

1. RF20 Visualizar los datos que presenta la tarjeta de estiba
2. RF21 Visualizar los datos del inventario de activos en el almacén
3. RF22 Visualizar los datos que presenta el inventario de activo por departamento
4. RF23 Visualizar los datos del acta de responsabilidad material

2.7. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con todas las funcionalidades requeridas, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación

Usabilidad

La aplicación será una herramienta para las personas encargadas de la gestión de inventarios de AFT en una empresa azucarera. Por tanto debe estar concebida para la facilidad de uso por personas sin experiencia previa con las computadoras y existir una pequeña documentación de usuario o ayuda.

Garantizar el acceso a todas las secciones del sistema, siempre y cuando existan los privilegios requeridos de autenticación, además de la validación de los datos entrados. Es de vital importancia el tratamiento de

errores, ya que esto facilitaría el trabajo con el sistema. Otro elemento importante es que el sistema es multiusuario, lo cual implica que varias personas pueden estar trabajando al mismo tiempo, sin poner en riesgo la integridad de los datos.

Soporte

Al sistema se le realizarán las pruebas necesarias por el administrador y los especialistas en informática de la empresa CAI Bartolomé Masó Márquez en los servidores donde se instalará el sistema, con el objetivo de detectar cualquier deficiencia. Después de instalada la aplicación se debe incluir el mantenimiento del sistema. La persona encargada de dar mantenimiento será el administrador del sistema.

Interfaz de Usuario

El sistema será usado por personas con una experiencia básica en informática, por lo que deberá estar provisto de elementos que hagan de este un programa fácil de usar, legible y con una interfaz amigable, estando confeccionada con colores poco llamativos y relajantes para la vista.

Seguridad

El sistema debe garantizar la protección de los datos almacenados. Para ello se establecerán diferentes niveles de acceso, garantizando que cada usuario tenga acceso solamente a las funcionalidades permitidas. Se aplicará en la autenticación de usuario la encriptación de la contraseña, para evitar que esta pueda ser descifrada por intrusos. Una vez que el usuario sea autenticado, se identifica su nivel de acceso, proporcionando el sistema las opciones asociadas a su permiso.

Confiabilidad.

Con el manejo de errores será posible prevenir fallos y recuperarse ante ellos. La base de datos será actualizada en tiempo real en caso de actualización, modificación o inserción de datos. Se utilizarán mecanismos de validación de los datos para que la información sea guardada de forma correcta y completa.

Es imprescindible minimizar la pérdida de datos ante roturas, fallas eléctricas y por almacenamiento físico. Por lo que se recomienda la realización de salvas periódicas de la base de datos del sistema aunque estas serán realizadas de forma manual.

Software.

- JRE 1.7 o superior
- PostgreSQL 9.1

Hardware.

- Cliente: Para uso del sistema se requerirán máquinas con 2.3 GHz de micro, tarjeta de red a 100 Mb/s o equivalente y 512 Mb de RAM y 1 Gb libres de disco duro. Se requiere de al menos una impresora para imprimir los reportes.

- Servidor de Base de Datos: Se recomienda para los servidores 40 Gb libres de disco duro, 1 Gb de RAM, 3.0 GHz de micro y tarjeta de red a 100 Mb/s o equivalente.

2.8. Diagrama de casos de uso

Los diagramas de casos de uso al igual que los requisitos funcionales se encuentran distribuidos por paquetes. El diagrama de paquetes se muestra más adelante.

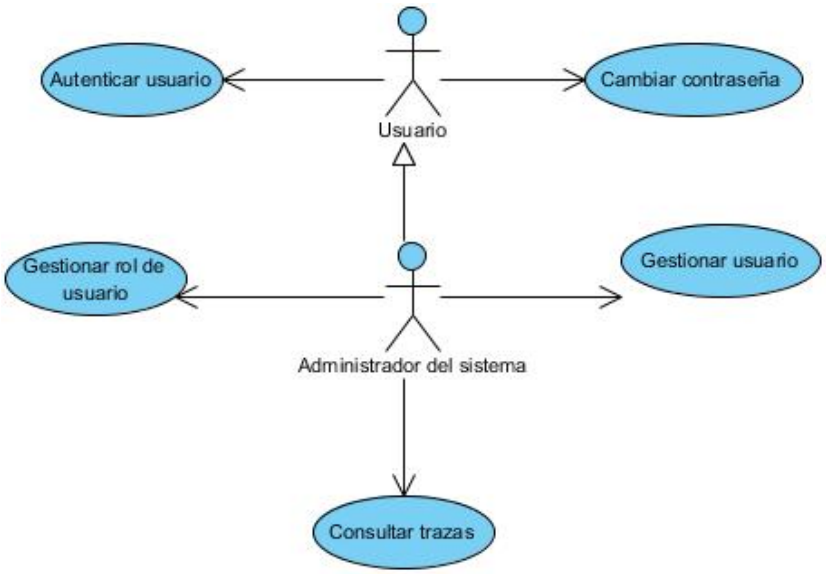


Figura 6 Diagrama de casos de uso del paquete seguridad

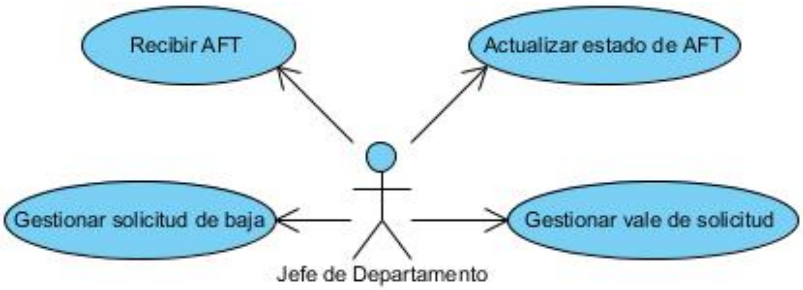


Figura 7 Diagrama de casos de uso del paquete departamento

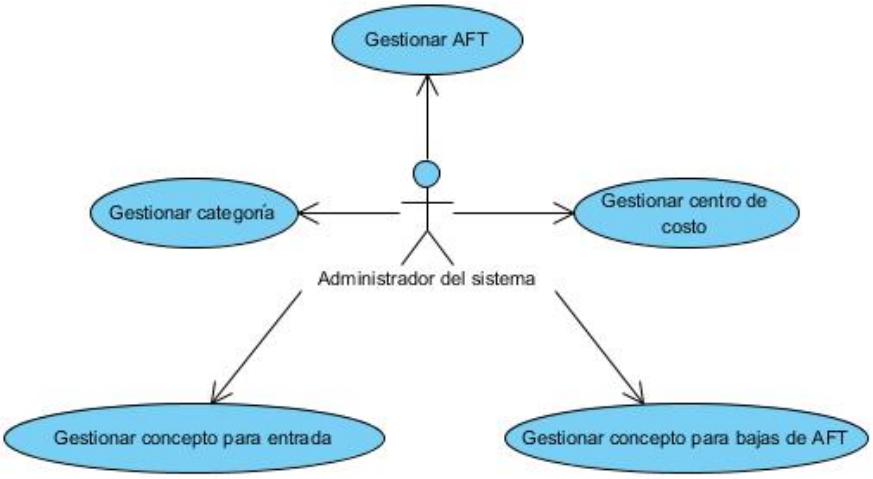


Figura 8 Diagrama de casos de uso del paquete configuración

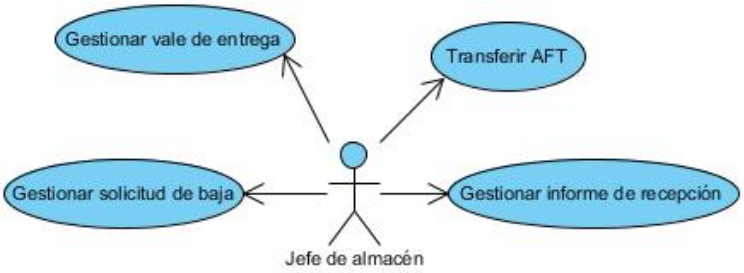


Figura 9 Diagrama de casos de uso del paquete almacén

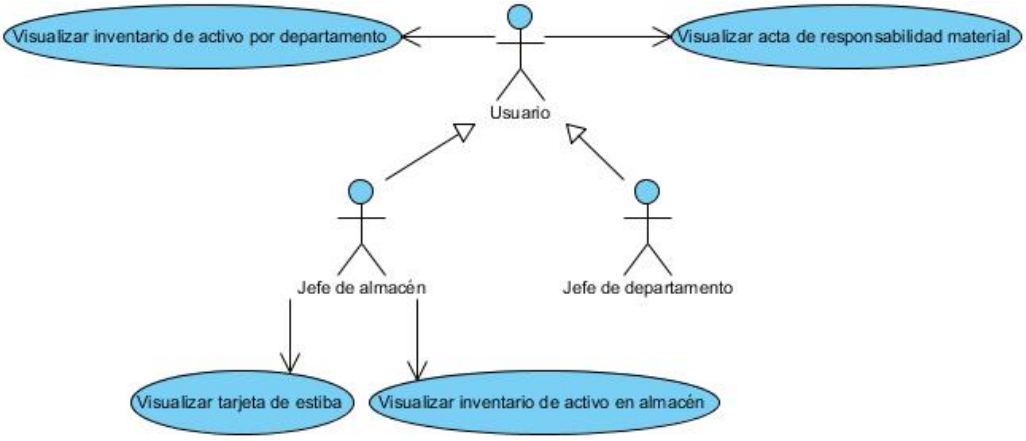


Figura 10 Diagrama de casos de uso del paquete reportes

2.9. Especificación y resumen de los casos de uso

A continuación se muestra la especificación del CU Gestionar Departamento. Consultar demás especificaciones de los casos de usos en anexo 1.

CU Gestionar departamento

Objetivo	Registrar, modificar, filtrar y eliminar el departamento
Actores	Jefe de almacén: (inicia) registrar, modificar, filtrar y eliminar los datos del departamento
Resumen	El caso de uso se inicia cuando el actor desea registrar, modificar, filtrar o eliminar el departamento. Si el actor selecciona la opción de registrar, el sistema le permite especificar los datos y registrarlos. Si desea modificar, el sistema muestra los datos y permite modificar los datos requeridos. Si desea filtrar, el sistema muestra los datos y permite consultarlos según los criterios especificados El actor podrá eliminar el departamento. Termina así el caso de uso
Complejidad	Alta
Prioridad	Alta
Precondiciones	Debe estar autenticado en el sistema y tener los permisos necesarios
Postcondiciones	Se registró, actualizó o eliminó un departamento

Flujo de eventos

	Actor	**Sistema**
	Accede al sistema y selecciona la opción gestionar departamento	
		Permite realizar las siguientes acciones sobre el departamento. • Registrar un departamento. Ver Sección 1: “Registrar departamento” • Modificar el departamento. Ver Sección 2 “Modificar departamento” • Eliminar departamento. Ver Sección 3: “Eliminar departamento”

Sección 1 “Registrar Departamento”

Flujo básico Registrar Departamento

	Actor	**Sistema**

1.	Selecciona la opción que le permite registrar el departamento	
2.		Permite introducir/seleccionar los siguientes datos del departamento: • Código • Nombre • Centro de Costo • Responsable • Descripción • Si es o no un almacén • Añadir trabajadores Muestra las siguientes opciones: • Aceptar • Cancelar
3.	Introduce los datos solicitados	
4.	Selecciona la opción que permite añadir trabajadores	
5.		Muestra una interfaz que permite añadir un trabajador solicitando los siguientes datos: • Nombre • Ocupación Muestra las siguientes opciones: • Aceptar • Cancelar
6.	Introduce los datos del trabajador	
7.	Selecciona la opción “Aceptar” para registrar el trabajador	
8.		Valida que se hayan llenados los campos para el registro de trabajador
9.		Añade los trabajadores al departamento
10.		Muestra listado actualizado de los trabajadores del departamento y permite eliminar un trabajador

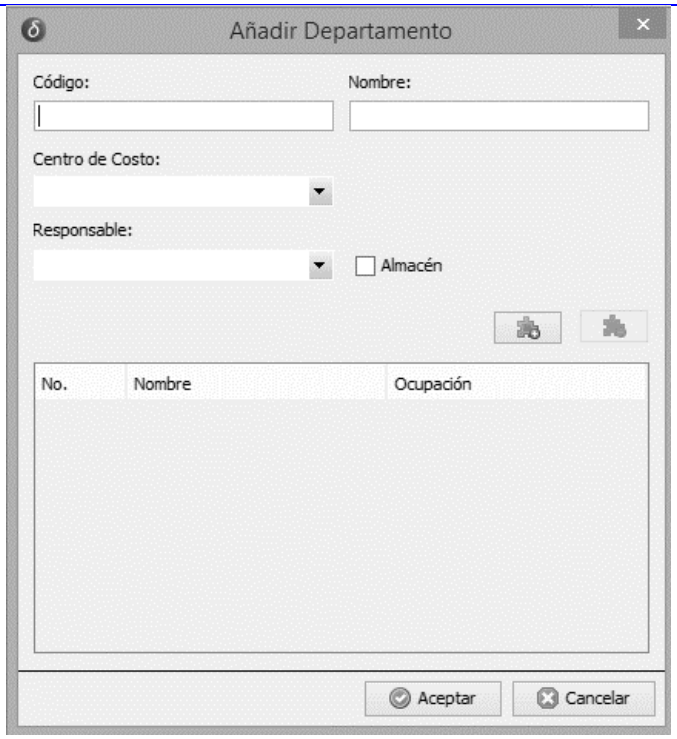
		seleccionado
11.	Selecciona la opción Aceptar para registrar el departamento	
12.		Valida que se hayan llenado los campos
13.		Registra los datos del departamento
14.		Muestra los datos del departamento
15.		Termina el caso de uso
Flujos alternos		
***a Cancelar el registro del departamento.**		
	Actor	**Sistema**
1.	Selecciona la opción que le permite cancelar el registro del departamento	
2.		Cancela las operaciones realizadas sobre el departamento
3.		Termina el caso de uso
5*a Cancelar el registro del trabajador.		
	Actor	**Sistema**
1.	Selecciona la opción que le permite cancelar el registro del trabajador	
2.		Cancela el registro del trabajador
3.		Termina el caso de uso
8 b Campos vacíos.		
	Actor	**Sistema**
1.	Presiona el botón Aceptar y existen campos vacíos para el registro del trabajador	

2.		Muestra un mensaje de error: “Todos los campos son requeridos”
3.		Regresa al paso 8 del flujo básico
10 *a Eliminar trabajador.		
	Actor	**Sistema**
1.	Selecciona el trabajador a eliminar y la opción que permite eliminar el trabajador	
2.		Elimina el trabajador seleccionado
3.		Regresa a la interfaz principal de la funcionalidad
12 a Campos vacíos.		
	Actor	**Sistema**
1.	Presiona el botón Aceptar y existen campos vacíos para el registro del departamento	
2.		Muestra un mensaje de error: “Todos los campos son requeridos”
3.		Regresa al paso 5 del flujo básico
12 b No se añade ningún trabajador		
	Actor	**Sistema**
1.	El actor selecciona la opción que le permite Aceptar el registro del departamento y no ha añadido ningún trabajador	
2.		Muestra un mensaje de error ”Debe definir al menos un trabajador”
3.		Termina el caso de uso

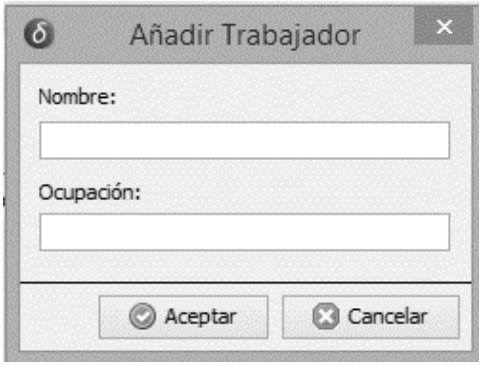
Sección 2: “Modificar departamento”		
Flujo básico Modificar departamento		
	Actor	**Sistema**
1.	Selecciona la opción que le permite modificar el departamento	
2.		Muestra los datos del departamento, y permite modificarlos, ver flujo básico: “Registrar Departamento” • Código • Nombre • Centro de Costo • Responsable • Descripción • Si es o no un almacén • Listado de trabajadores Muestra las siguientes opciones: • Aceptar • Cancelar Permite • Eliminar trabajador seleccionado
3.	Modifica los datos deseados	
4.	Selecciona la opción que le permite modificar los cambios	
5.		Valida los datos
6.		Actualiza los cambios en el departamento
7.		Muestra los datos del departamento
8.		Termina el caso de uso
Flujos alternos		
***a Cancelar la operación.**		
	Actor	**Sistema**

1.		Cancela las operaciones realizadas sobre el departamento
2.		Regresa a la interfaz principal Gestionar Departamento
2 *a Eliminar trabajador.		
	Actor	**Sistema**
	Selecciona el trabajador a eliminar y la opción que permite eliminar el trabajador	
1.		Elimina el trabajador seleccionado
2.		Regresa a la interfaz principal de la funcionalidad Modificar Departamento
Flujos alternos		
5a Campos vacíos.		
	Actor	**Sistema**
1.	Selecciona la opción “Aceptar” y existen campos vacíos	
2.		Muestra un mensaje de error: “Todos los campos son requeridos”
3.		Regresa al paso 3 del flujo básico
Sección 3: “Eliminar departamento”		
Flujo básico Eliminar departamento.		
	Actor	**Sistema**
1.	Selecciona el departamento a eliminar	
2.	Selecciona la opción que le permite eliminar un departamento	

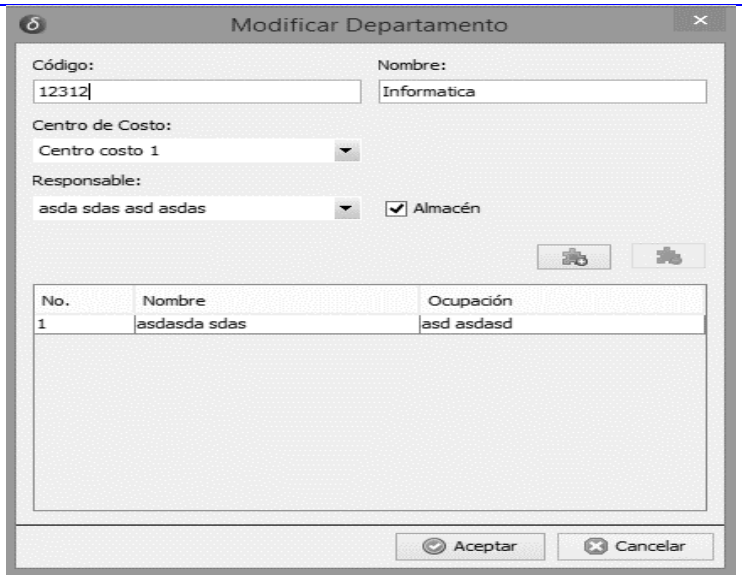
3.		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”
4.		Y permite: • Aceptar la operación • Cancelar la operación
5.	Selecciona la opción que le permite aceptar la operación	
6.		Elimina el departamento
7.		Termina el caso de uso
Flujos alternos		
***a Cancelar la operación.**		
	Actor	**Sistema**
1.	El actor selecciona la opción que le permite cancelar la operación	
2.		Sale de la vista actual sin eliminar el departamento seleccionado
3.		Regresa a la vista anterior
4.		Termina el caso de uso
Prototipo elemental de interfaz gráfica de usuario		
Registrar Departamento		



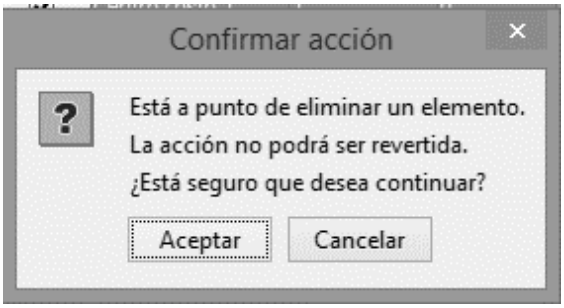
Añadir Trabajador.



Modificar Departamento.



Eliminar departamento



2.10. Patrones utilizados

Los patrones arquitectónicos capturan existencia, experiencia comprobada en el desarrollo del software y ayudan a promover buenas prácticas de diseño. Cada patrón es específico a un problema recurrente en el diseño e implementación de un sistema de software y expresa los esquemas fundamentales de la organización estructural en ellos, por lo que en este epígrafe se realiza un análisis de los patrones a utilizar en el desarrollo de la aplicación.

2.10.1. Patrones de diseño

Según Metsker (28) un patrón de diseño es:

- Una solución estándar para un problema común de programación.

- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.
- Un lenguaje de programación de alto nivel.
- Una manera más práctica de describir ciertos aspectos de la organización de un programa.
- Conexiones entre componentes de programas.
- La forma de un diagrama de objeto o de un modelo de objeto.

Objetos de Acceso a Datos (DAO)

El propósito del patrón DAO es abstraer y encapsular todos los accesos a la fuente de datos. Con esto se obtiene como ventajas fundamentales:

1) Se mitiga el problema de la transparencia
2) Se baja el nivel de acoplamiento entre clases, reduciendo la complejidad de realizar cambios
3) Se aísla las conexiones a la fuente de datos en una capa fácilmente identificable y mantenible

Su principal ventaja es que separa en una capa aparte todo el acceso a datos y abstrae la comunicación con la fuente de datos, esto hace que sea mucho más fácil una migración de fuente de datos en caso de ser necesaria. (25)

El uso de este patrón se evidencia en la utilización de las clases DAO como elemento que le entregará a la aplicación una única forma de acceso a los datos que se gestionan el la base de datos. Ejemplo de esta clase es DepartamentoDao.

Patrones generales para asignar responsabilidades (GRASP)

Los patrones GRASP describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

- **Alta cohesión:** Cada elemento del diseño debe realizar una labor única dentro del sistema.

En el sistema, poniendo en práctica este patrón, cada clase tiene una función específica y única. Por ejemplo las clases DAO se encargan solamente de la gestión de la información que se almacena en la base de datos.

- **Bajo acoplamiento:** Debe haber pocas dependencias entre las clases.

Un ejemplo de utilización de este patrón es la relación que existe entre el controlador y la fachada. La fachada se encarga de proveerle los datos necesarios al controlador para evitar las relaciones directas

con los objetos de acceso a datos. Con este débil acoplamiento entre clases, el desarrollador es libre de actualizar y/o agregar funcionalidades según sea necesario, sin afectar otras partes del sistema.

- **Controlador:** Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas.

Es usado en cada clase controladora. Cada vista tiene asociada una controladora encargada de atender las peticiones que en ella se generan. Éstas invocan a las clases de su negocio que necesitan para cumplir con la petición, recibe la respuesta y construye la vista correspondiente para mostrar finalmente la respuesta al usuario. Aporta al sistema la ventaja de que toda la gestión de peticiones se centraliza, lo que hace posible mayor agilidad en la identificación de errores y en la corrección de los mismos. Además aumenta el mantenimiento y la reusabilidad del código.

Patrones GOF (Gang of four)

Creacionales

- **Singleton (instancia única):** está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. (28)

Un ejemplo de implementación de este patrón se evidencia en la siguiente figura.

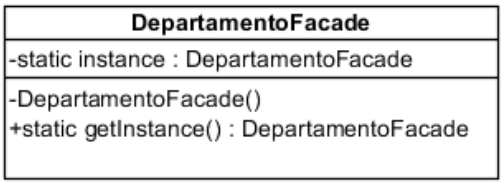


Figura 11 Ejemplo de utilización del patrón singleton

En la clase un método solo crea una instancia cuando aún no existe ninguna. Para asegurar que la clase no puede ser instanciada nuevamente se crea un método público y estático que es el único responsable de crear una instancia de la clase (+ static getInstance). Además se declara al constructor como privado y de esta forma se garantiza que el constructor no sea accesible desde fuera de la clase (- DepartamentoFacade).

Estructurales:

- **Fachada (Facade):** es un patrón de diseño que sirve para proveer de una interfaz unificada sencilla que haga de intermediaria entre un cliente y una interfaz o grupo de interfaces más complejas. Puede hacer una biblioteca de software más fácil de usar y entender, ya que implementa métodos convenientes para tareas comunes; hacer el código que usa la biblioteca más legible, por la misma razón. (25)

Este patrón es utilizado en el sistema para separar el modelo y el controlador, lo que hace posible la independencia entre ellos, por lo que un cambio en uno sería transparente para el otro.

2.10.2. Patrón arquitectónico

Modelo Vista Controlador (MVC)

El estilo arquitectónico modelo vista controladora separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario en tres clases diferentes (29):

Modelo: administra el comportamiento y los datos del dominio de aplicación, responde a requisitos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: maneja la visualización de la información.

Controlador: interpreta las acciones del ratón y del teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

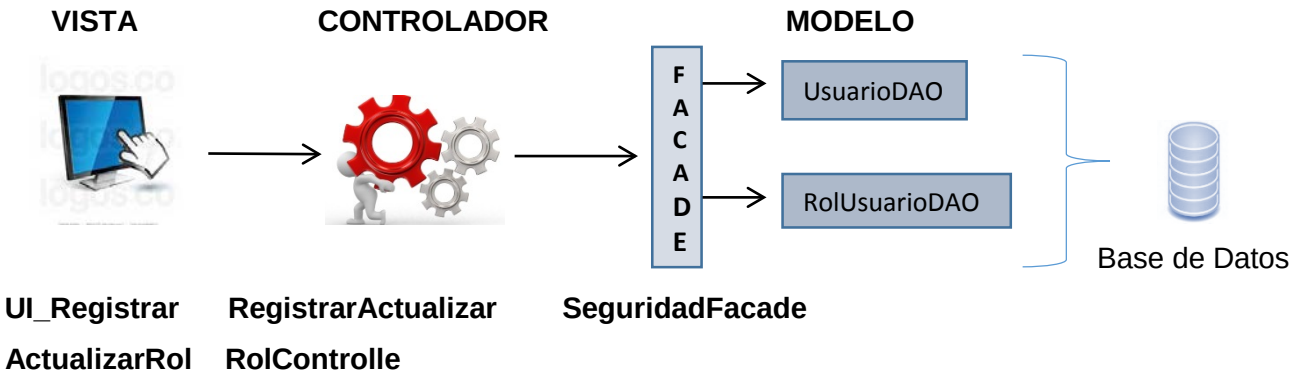
Entre las principales ventajas de hacer uso del MVC se encuentran:

- Soporte de vistas múltiples. Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.
- Adaptación al cambio. Los requisitos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo.
- Permite la escalabilidad de la aplicación en caso de que sea requerido. El diseño MVC tiene una estructura organizada y debido a la clara separación de tareas que posee, facilita la modificación para aumentar las capacidades funcionales o de almacenamiento de la aplicación, asegurando la independencia del desarrollo hecho a los cambios sufridos.
- La conexión entre el modelo y sus vistas es dinámica; se produce en tiempo de ejecución, no en tiempo de compilación.

En el sistema de gestión de inventario el patrón arquitectónico se evidencia de la siguiente forma:

- En la vista se encuentran todas las clases interfaces con las cuales interactúa el usuario, esta presentan la estructura (prefijo UI seguido por el nombre de la clase). **Ejemplo:** UI_RegistrarActualizarRol.

- En la capa controladora estarán todas las clases controladoras que presentan la estructura (nombre de la clase seguido del sufijo Controller). **Ejemplo:** RegistrarActualizarRolController.
- En la capa modelo se encuentran todas las clases de acceso a datos, así como la fachada que se encargara de unificar funcionalidades entre el controlador y el modelo, estas clases presentaran la siguiente estructura (nombre de la clase seguido de los sufijos Facade, DAO). **Ejemplo:** SeguridadFaceade, UsuarioDAO.



2.11. Modelo de diseño

El modelo de diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. En el diseño modelamos el sistema y encontramos su forma para que soporte todos los requisitos, incluyendo los requisitos no funcionales, además de que impone una estructura del sistema que debemos mantener lo más fielmente posible.

2.11.1. Diagrama de paquetes

El siguiente diagrama muestra todos los paquetes en los que fue estructurado el diseño de la solución que se propone.

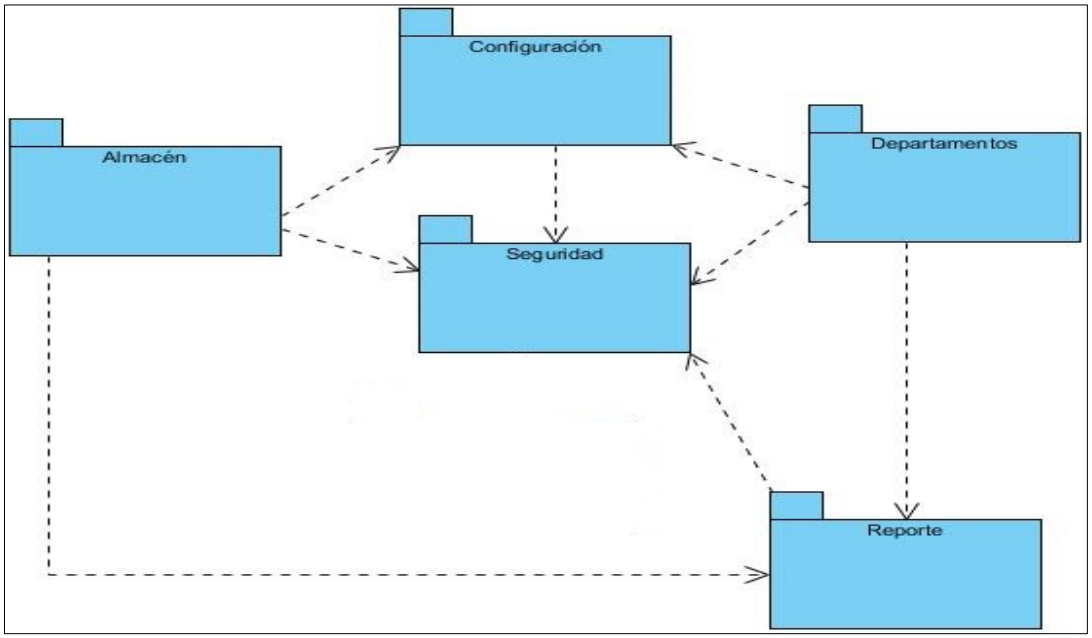


Figura 12 Diagrama de paquetes

El diagrama de paquetes expuesto anteriormente sirve para dar una vista de la organización interna de la estructura del sistema. En el mismo se denotan cinco paquetes principales: Seguridad, Configuración, Almacén, Departamento y Reportes**.** La estructura de paquetes guarda ciertas dependencias con respecto al diseño general. Es por ello que no se mostrará el propósito de cada paquete sino especificaciones de cada uno de estos en pos de su mejor entendimiento.

Paquete Seguridad: Es el paquete que tiene la responsabilidad de verificar los permisos de acceso de los usuarios del sistema, además de la creación de nuevos usuarios, cambios de contraseña y gestión los roles que estos tendrán dentro del sistema.

Paquete Configuración: Es el paquete donde se almacenan todas las clases relacionadas con la configuración del sistema como son las categorías, conceptos para bajas de AFT y departamentos.

Paquete Almacén: Este paquete contiene todos los elementos relacionados con el área de almacén. Desde estos se gestionan los informes de recepción, vale de entrega y los movimientos de AFT.

Paquete Departamento: Es el paquete que contiene los procesos relacionados con un departamentos. En estos se gestionan los Vale de solicitud y se pude actualizar el estado de los AFT.

Paquete Reportes: Recoge todos los reportes que se manejan en el sistema y que son las salidas del mismo.

2.11.2. Diagrama de Clases del Diseño

A continuación se muestra el diagrama de clases de la funcionalidad gestionar departamento.

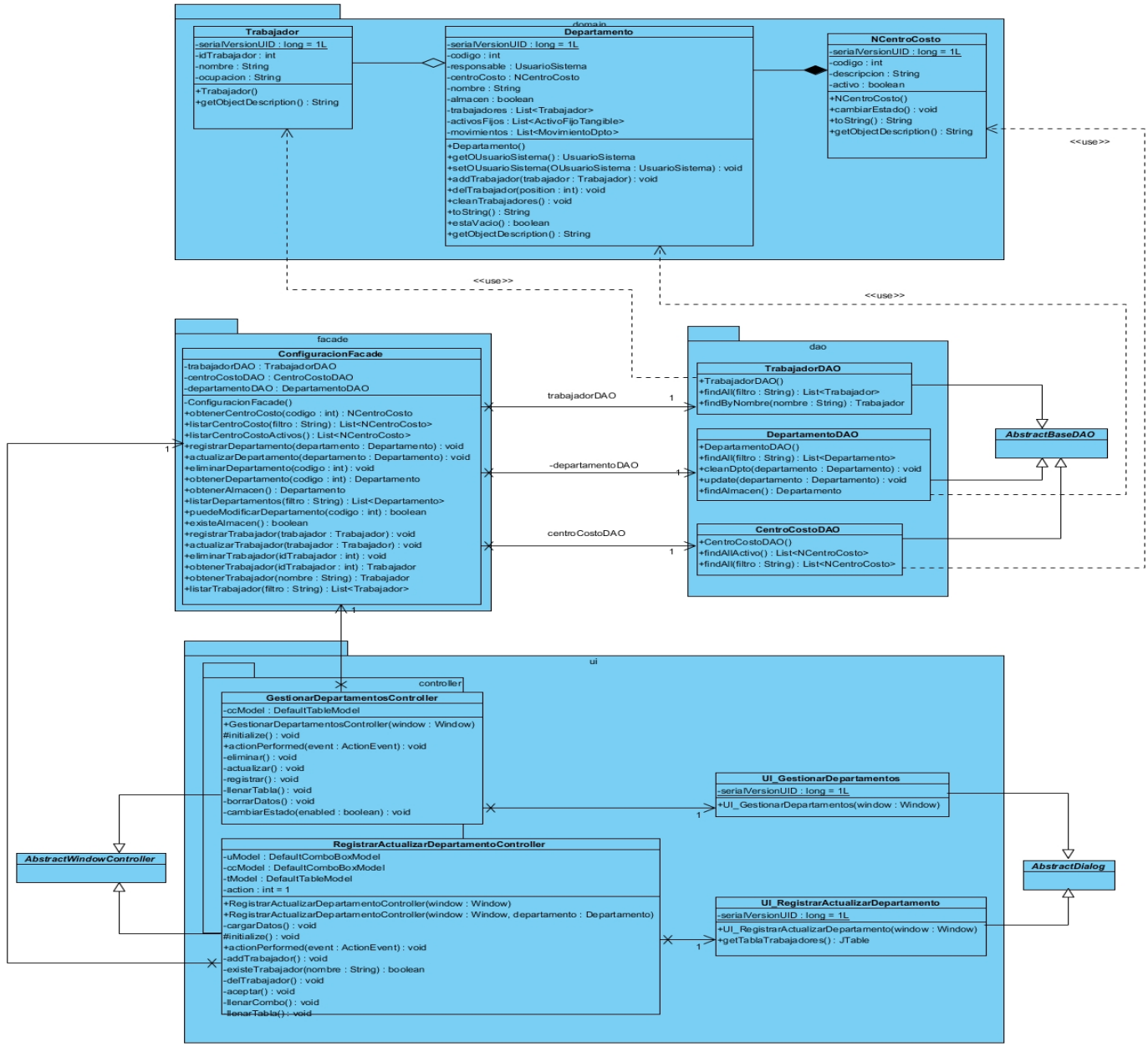


Figura 13 Diagrama de clases GestionarDepartamento

En el diagrama aparece la clase interfaz, que está identificada por el prefijo UI más el nombre de la interfaz (UI_GestionarDepartamento). De forma general y como se evidencia en la imagen cada una de ellas están asociadas a un controlador que tiene el nombre de la interfaz y termina con la palabra Controller. Los controladores están asociados a una fachada, la cual facilita la comunicación entre los controladores y los

DAO, estos últimos obtienen los datos de las clases entidades, las cuales contienen la información que se almacena en la base de datos.

2.12. Modelo de datos

Un modelo de datos es un conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Los modelos de datos comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones.

El modelo Entidad/Relación ha tenido una gran difusión en la comunidad informática dedicada a las base de datos, prueba de ello es que ha sido el modelo más extendido en las herramientas CASE de ayuda al diseño de base de datos. El modelo E/R puede ser usado como una base para una vista unificada de los datos, adoptando el enfoque más natural del mundo real que consiste en entidades y relaciones (30).

En la siguiente figura se representa el modelo de datos del sistema. Este incluye las entidades persistentes, así como los nomencladores utilizados. Uno de sus principales objetivos es garantizar un rendimiento óptimo en el acceso a datos y un mínimo acoplamiento al sistema utilizado para gestionar la base de datos.

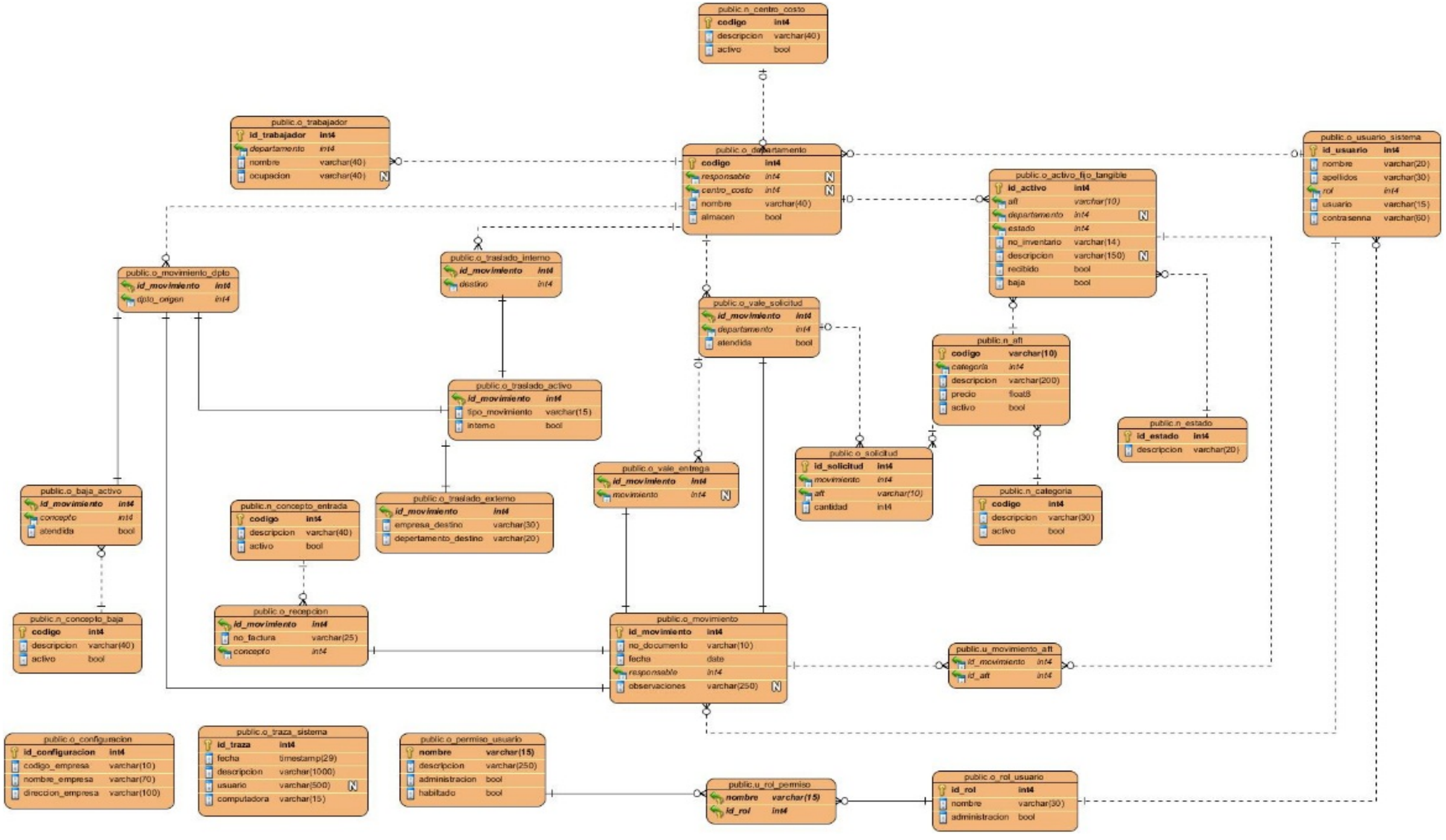


Figura 14 Modelo de datos

2.13. Diagrama de interacción del diseño

Los diagramas de interacción son modelos que describen la manera en que colaboran grupos de objetos para cierto comportamiento. Son útiles para analizar el comportamiento de un grupo de objetos en un mismo caso de uso. Los diagramas de interacción muestran cierto número de ejemplos de objetos y los mensajes que se pasan entre estos objetos dentro del caso de uso. Hay dos tipos de diagramas de interacción:

- Diagramas de secuencia: muestra una interacción ordenada según la secuencia temporal de eventos. En particular, muestra los objetos participantes en la interacción y los mensajes (llamadas a métodos) que intercambian, ordenados según su secuencia en el tiempo.

- Diagramas de colaboración: muestra cómo las instancias específicas de las clases trabajan juntas para alcanzar un objetivo común. En cierta forma, en un diagrama de colaboración se detallan las asociaciones que se muestran en un diagrama de clases, describiendo el intercambio de mensajes entre objetos y las relaciones entre los objetos, sin tomar en cuenta la oportunidad o la dimensión temporal de dichas relaciones.

A continuación se muestra el diagrama de secuencia de la funcionalidad registrar departamento.

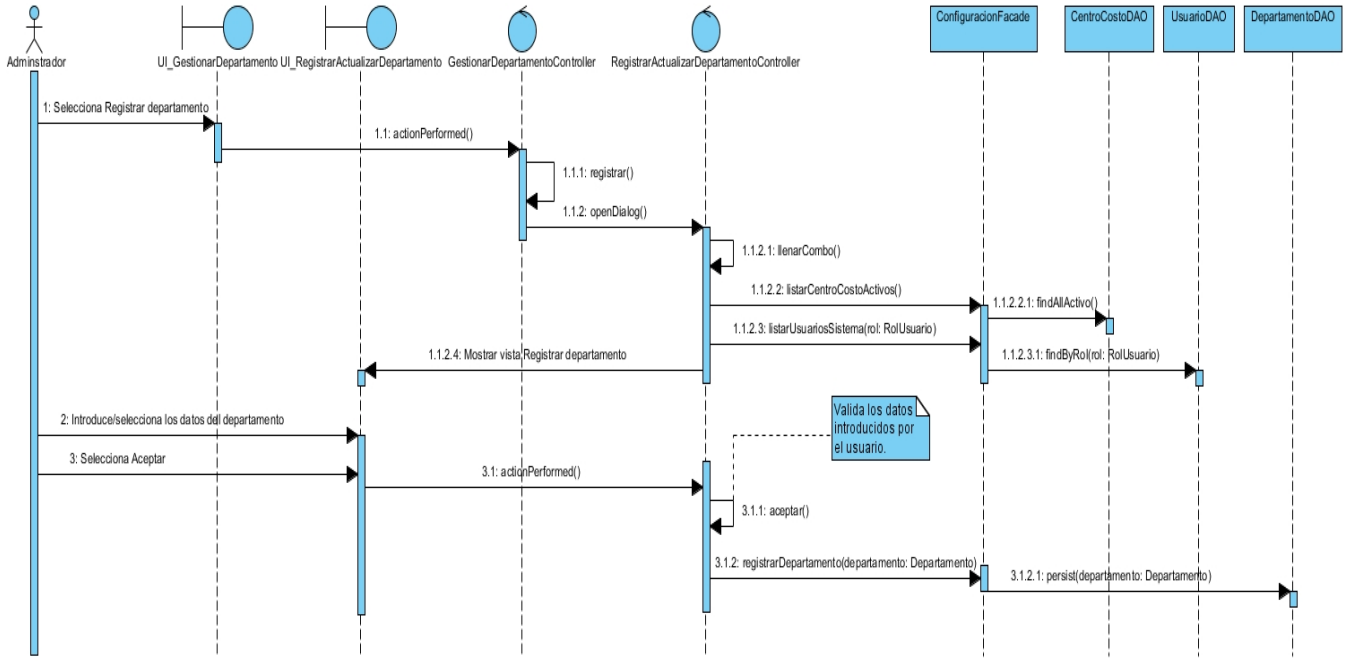


Figura 15 Diagrama de secuencia de la funcionalidad RegistrarDepartamento

2.14. Conclusiones parciales

Durante el diseño en base a la metodología OpenUP y los patrones arquitectónicos y de diseño propuestos se pudo concluir que:

- La flexibilidad de la metodología OpenUP permitió un desarrollo centrado en la implementación, a partir de la creación de solo los artefactos necesarios e ilustrativos del proceso, muy necesario a causa de un cronograma apretado y un equipo de trabajo pequeño.
- La utilización de patrones de diseño reconocidos, y el apoyo en el patrón arquitectónico MVC, evidenció la construcción de un sistema legible, reutilizable y de fácil mantenimiento, elementos que favorecen el aporte tecnológico que representa este sistema para la empresa.

Capítulo.3 Implementación y pruebas

3.1. Introducción

En el presente capítulo se reflejan artefactos como el diagrama de componentes y el diagrama de despliegue y se describen las funcionalidades implementadas más significativas para el desarrollo del sistema. Se reflejan además las pruebas realizadas al software desarrollado, con el objetivo de comprobar las funcionalidades en los diferentes escenarios para medir la calidad del producto y el grado en el que cumple con los requisitos previamente identificados.

3.2. Modelo de implementación

El modelo de implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

3.2.1. Diagrama de componente

Los diagramas de componentes modelan la vista estática de un sistema. Se representa como un grafo de componentes software unidos por medio de relaciones de dependencia (compilación, ejecución), pudiendo mostrarse las interfaces que estos soporten.

A continuación se muestra la estructura general del diagrama de componentes del sistema, teniendo en cuenta la arquitectura utilizada.

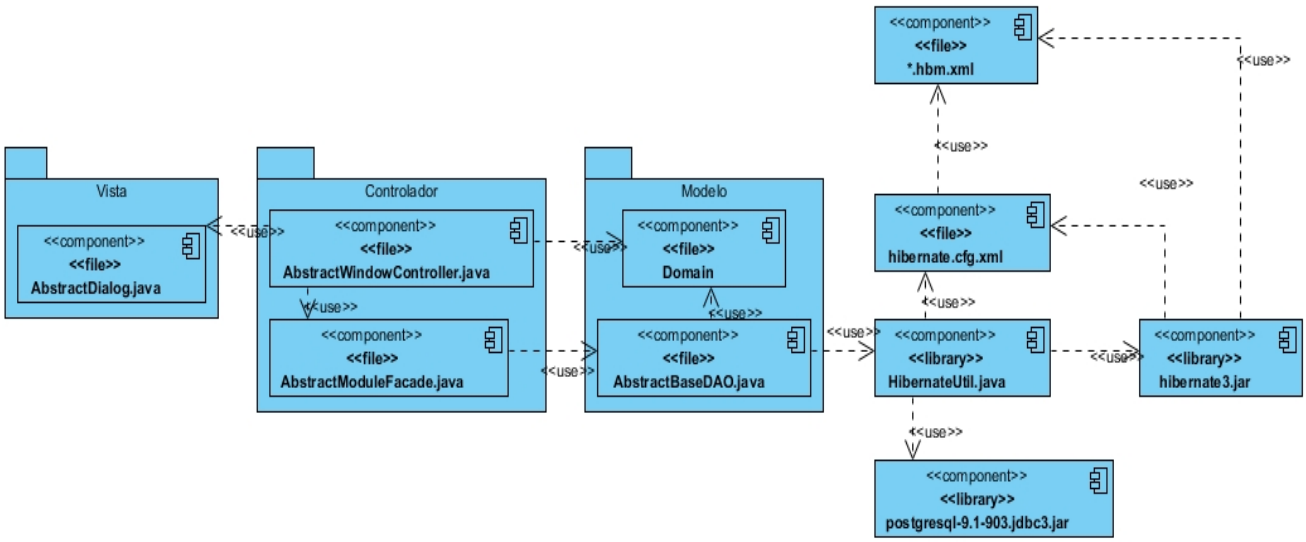


Figura 16 Estructura general del diagrama de componentes del sistema

Representación del diagrama de componentes de la funcionalidad gestionar departamento. Consultar otros diagramas en anexo 4.

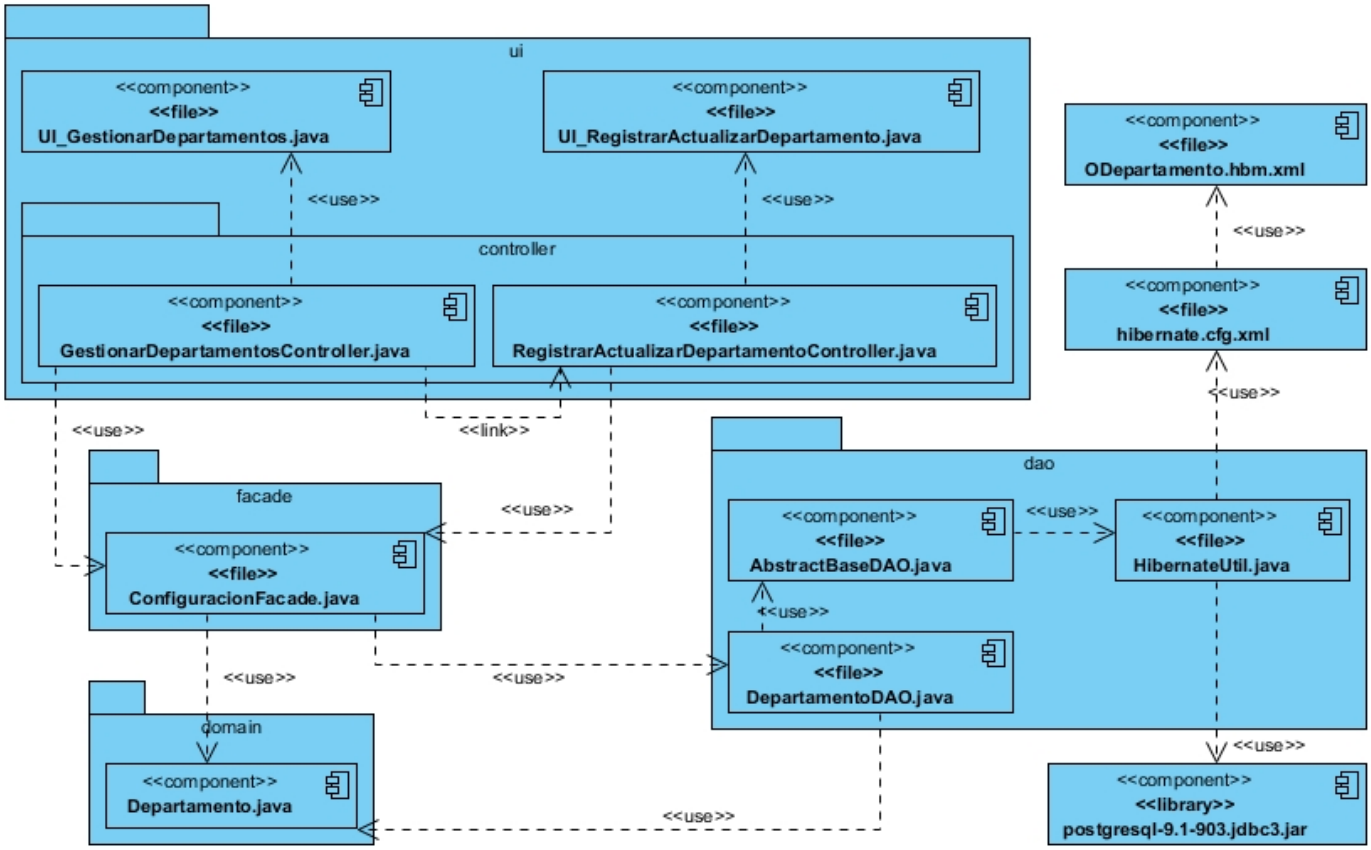


Figura 17 Estructura del diagrama de componentes GestionarDepartamento

3.2.2. Diagrama de despliegue

Un diagrama de despliegue representa la distribución física del sistema en términos de cómo se distribuirán las funcionalidades entre los nodos, cada nodo representa un recurso de cómputo, siendo estos procesadores o dispositivos hardware que se necesitarán para el despliegue del sistema.

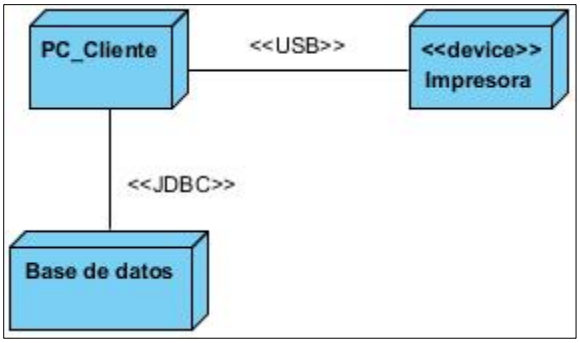


Figura 18 Diagrama de despliegue

3.3. Justificación del estándar de codificación

En general, un estándar de codificación son reglas que se siguen para la escritura del código fuente. De esta manera a otros programadores se les facilita entender un código cualquiera (como identificar las variables, las funciones o métodos, etc.).

Para el desarrollo del sistema se establecen como estándares de codificación los siguientes:

- Los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto se pondrá con minúscula. Cuando sea un nombre compuesto se utilizará la notación Pascal Casing[3]. Ejemplo: Departamento.
- Las clases controladoras se conforman con la unión del nombre de la clase y la palabra Controller. Ejemplo: DepartamentoController.
- Las interfaces de usuario se distinguirán por tener el prefijo UI seguido del nombre de la interfaz. Ejemplo: UI_RegistrarDepartamento.
- La fachada debe tener el nombre de la fachada seguido de la palabra Facade. Ej: SeguridadFacade.
- El nombre de las funciones se escribe con la primera palabra en minúscula y si es un nombre compuesto se empleará la notación lowerCamelCase[4]. Ejemplo: registrarDepartamento.
- El nombre de las variables se escribe con la primera palabra con minúscula, si es un nombre compuesto se utilizará la notación CamelCase. Ejemplo: responsable.

[3] PascalCasing es un procedimiento de programación común en el lenguaje Java y .Net. La nomenclatura está compuesta por tantas palabras como sean necesarias. La primera letra de cada una de las palabras irá siempre en mayúsculas y se obvia el uso de artículos. Ej.: GestionarDepartamentoController.

[4] CamelCase es la nomenclatura por excelencia en el mundo Java. Consistente en utilizar las mayúsculas como separadores de palabras. Si comienza con mayúscula, se denomina *UpperCamelCase* y, si no, *lowerCamelCase*.

3.4. Descripción de la implementación del sistema

Implementación de la vista

En la implementación de la vista la actividad fundamental es la creación de la interfaz de usuario ante cada petición. Para la implementación de la vista en el sistema se debe tener dominio de Swing y Jasper Reports. El *framework* Swing incluye *widgets* para interfaz gráfica de usuario tales como cajas de texto, botones, desplegables y tablas. Por su parte Jasper Reports se utiliza para la confección de los reportes que se manejan en el sistema.

Implementación del controlador

La implementación del controlador tiene dos momentos significativos: la implementación de los controladores y la implementación de la fachada. Los controladores dentro del sistema implementan una interfaz genérica, en la que se definen funcionalidades comunes que estos deberán contener.
Además serán los encargados de comunicar la vista con el modelo a través de la fachada. Son también los responsables de realizar las validaciones de los datos de entrada y dar formato a los datos de salida.
La implementación de la fachada provee una simplificación de la relación entre el modelo y el controlador, reduciendo al mínimo los objetos con los que esta última deberá interactuar. En la fachada se encuentran implementadas funcionalidades, brindándole de este modo al controlador un punto de mínimo acceso a la información en cualquier momento.

Implementación del modelo

En la implementación del modelo se realizan dos tareas fundamentales: la creación de los ficheros de mapeo de Hibernate (hbm.xml) y la implementación de las funcionalidades de los DAOs. Estos últimos hacen uso de las transacciones de Hibernate para garantizar la integridad de los datos. En los ficheros XML se realiza una configuración de manera tal que se asocie una tabla de la base de datos a un objeto java, relacionando cada campo de la tabla con un atributo del objeto. En la creación de estos ficheros juega un papel importante el *plugin* de eclipse Hibernate Tools, el cual facilita la generación de código en algunos momentos durante su confección.
Para la implementación de las funcionalidades de los DAO se usa como primera opción el API (*Aplication Programming Interface*) Criteria del framework Hibernate, que permite la creación de consultas al gestor de base de datos de gran complejidad en la que intervienen varias tablas, sin la necesidad de introducir código en lenguaje SQL. Cuando estas APIs no son suficientes para cumplir

con las funcionalidades se utiliza también el lenguaje de consultas de *Hibernate Query Languaje* (HQL).

Con la utilización de este lenguaje se obtiene independencia de la aplicación con respecto al gestor de base de datos utilizado, ya que las consultas se realizan de manera transparente, gestionadas directamente por el API provisto por Hibernate para ello.

Uno de los aspectos a tener en cuenta a la hora de desarrollar un sistema es que el mismo mantenga la integridad de la información. Uno de los procedimientos utilizados para cumplir con este requisito es el empleo de transacciones. Las mismas proporcionan cuatro garantías muy importantes: la consistencia, atomicidad, aislamiento y durabilidad.

En el sistema que se desarrolla las transacciones se realizan a nivel de DAO, por ende cada DAO que implique operaciones de escritura sobre la base de datos, se ejecuta como una transacción; por tanto si una de las operaciones falla, se cancelarán todas las demás.

3.5. Pruebas

Las pruebas constituyen un paso importante para el desarrollo de software pues permiten detectar y corregir errores desde etapas tempranas. La calidad de un sistema software es algo subjetivo que depende del contexto y del objeto que se pretenda conseguir. Para determinar dicho nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Las pruebas de software implican ejecutar una implementación del software con datos de prueba. Se examinan las salidas del software y su entorno operacional para comprobar que funcionan tal y como se requiere. Las pruebas son una técnica dinámica de verificación y validación. (27)

Nivel de Prueba

La Prueba es aplicada para diferentes tipos de objetivos, en diferentes escenarios o niveles de trabajo. Debido a que las pruebas son agrupadas por niveles que se encuentran en distintas etapas del proceso de desarrollo.

Se distinguen los siguientes niveles de pruebas:

- **Prueba de desarrollador:** Es la prueba diseñada e implementada por el equipo de desarrollo.
- **Prueba independiente:** Es la prueba que es diseñada e implementada por alguien independiente del grupo de desarrolladores.

- **Prueba de unidad:** Es la prueba enfocada a los elementos testeables más pequeño del software. El objetivo es comprobar que el módulo, entendido como una unidad funcional, está correctamente codificado.
- **Prueba de integración:** Es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar un caso de uso.
- **Prueba de sistema:** Esta prueba tiene como objetivo verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las operaciones apropiadas funcionando como un todo.
- **Prueba de aceptación:** Es la prueba final antes del despliegue del sistema. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales.

De estos niveles de prueba se realizarán las pruebas de unidad y las de aceptación. Las primeras son aplicables a componentes representados en el modelo de implementación para verificar que los flujos de control y de datos están cubiertos, y que ellos funcionen como se espera. La prueba de unidad siempre está orientada a caja blanca. Para ello se utilizará la herramienta JUnit. Esta es un conjunto de clases (*framework*) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente.

También se realizarán las pruebas de aceptación, para verificar que el cliente está satisfecho con el sistema realizado y que puede ser usado por ellos para ejecutar aquellas funciones y tareas para las cuales el software fue construido. Estas pruebas son realizas por el cliente. Son básicamente pruebas funcionales, sobre el sistema completo, y buscan una cobertura de la especificación de requisitos. Estas pruebas no se realizan durante el desarrollo, pues sería impresentable al cliente; sino que se realizan sobre el producto terminado e integrado o pudiera ser una versión del producto o una iteración funcional pactada previamente con el cliente.

Tipos de prueba

Existen varios tipos de prueba, cada una con su objetivo específico y una técnica que lo soporta.
Para este trabajo se realizarán las pruebas funcionales y dentro de ellos las siguientes:

- **Función:** Pruebas fijando su atención en la validación de las funciones, métodos, servicios, caso de uso.
- **Seguridad:** Asegurar que los datos o el sistema solamente es accedido por los actores deseados.

Métodos de prueba

Se usan los dos métodos fundamentales de prueba:

- **Método de caja negra** para demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.
- **Método de caja blanca** para comprobar los caminos lógicos del software proponiendo casos de prueba que se ejerciten conjuntos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coinciden con el esperado o mencionado.

Técnica de prueba

Dentro de las técnicas utilizadas en el método de caja negra se usará la partición de equivalencia. El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada.

Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. (30)

En el anexo 3 se muestra el caso de prueba para el CU Gestionar Departamento.

Para las pruebas de aceptación se utiliza la técnica de Prueba Beta.

Se lleva a cabo por los usuarios finales del software en los lugares de trabajo de los clientes. A diferencia de la prueba alfa, el desarrollador no está presente normalmente. Así, la prueba beta es una aplicación "en vivo" del software en un entorno que no puede ser controlado por el desarrollador. El cliente registra todos los problemas (reales o imaginarios) que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. (31)

3.5.1. Resultados de las pruebas

Prueba de aceptación

Estas pruebas fueron realizadas por el cliente. En el informe enviado habían algunas no conformidades que fueron resultas convenientemente.

Pruebas de seguridad

Para realizar las pruebas de seguridad se crearon tres usuarios que tenían diversos roles dentro del sistema. Para cada uno de ellos fueron definidas dos funcionalidades que estaban de acuerdo al nivel de seguridad de cada usuario. En la siguiente tabla se muestran los detalles de cómo fueron distribuidas las pruebas.

Usuario	Rol	Funcionalidades
Usuario 1	Administrador del sistema	• Consultar trazas del sistema. • Crear usuario.
Usuario 2	Almacenero	• Gestionar informe de recepción. • Gestionar Vale de entrega
Usuario 3	Jefe de Departamento	• Realizar solicitud de baja de AFT. • Solicitar AFT.

El resultado de esta prueba fue satisfactorio, permitiendo comprobar que el sistema mantiene una seguridad aceptable.

Pruebas de caja negra

Los casos de prueba diseñados para la aplicación de la técnica de partición equivalente del método de Caja Negra, permitió además realizar las pruebas de función. El resultado de estas pruebas se evidencia en la siguiente tabla.

Iteración	Total de no conformidades	# de no conformidades que proceden	# de no conformidades que no proceden	# de no conformidades resueltas
1era	17	15	2	15
2da	5	5	-	5
3era	-	-	-	-

Se realizaron 3 iteraciones de pruebas. Las principales causas de no conformidades detectadas fueron:

- Los números que identifican a los escenarios no seguían un orden ascendente
- Los pasos del flujo central de los casos de prueba no se encontraban enumerados
- La descripción de un escenario no está en correspondencia con su nombre
- En la descripción de variable se especifica que admite solo letras, pero además admite números, por lo que debería especificarse que admite caracteres alfanuméricos
- El caso de prueba describía algunos mensajes que la aplicación no mostraba
- Fueron detectados algunos errores ortográficos en la descripción de los casos de prueba

De manera general, las no conformidades detectadas, no fueron significativas, por lo que no afectaban la calidad de la aplicación. Estas fueron resueltas convenientemente, logrando que la tercera iteración de pruebas resultara satisfactoria en 100%.

Pruebas de caja blanca con JUnit

Las pruebas se realizaron a las clases que contienen funcionalidades que determinan el correcto funcionamiento del sistema.

Para explicar el proceso de realización de las pruebas de caja blanca con JUnit se toma como objeto de estudio el método (*findAllByDpto*) que se encuentra en la clase ActivoFijoDao.

La realización de pruebas haciendo uso de JUnit permite comprobar si el resultado obtenido luego de la ejecución de la funcionalidad a probar es el esperado. Para la realización de pruebas al método

previamente seleccionado se creó una clase la cual debe extenderla clase TestCase brindada por el framework JUnit, en este caso dicha clase se nombra *ActivoFijoDAOTest* en la cual se realizan una serie de pasos los cuales se muestran y describen a continuación.

```
public class ActivoFijoDAOTest
{
    /*Clase a probar*/
    private ActivoFijoDAO activoFijoDAO;

    /*Datos auxiliares*/
    private DepartamentoDAO departamentoDAO;

    /*Inicialización de atributos de la clase*/
    @Before
    public void setUp() throws Exception
    {
        this.activoFijoDAO = new ActivoFijoDAO();
        this.departamentoDAO = new DepartamentoDAO();
    }
```

Figura 19 Clase ActivoFijoDAOTesr empleada para la realización de las pruebas

Primeramente se crea un objeto de la clase en la cual se encuentra el método que se va a probar, luego se definen cada uno de los parámetros que recibe el método. Posteriormente en el método setUP() se inicializan todos los atributos anteriormente creados.

```
@Test //findAllByDpto**
public void testFindAllByDepartamento() throws Exception
{
    Departamento departamento = departamentoDAO.findById(22222);
    List<ActivoFijoTangible> activosFijos = activoFijoDAO.findAllByDpto(departamento);

    String [] nrosInventarioValidos = {"MB00000001", "MB00000002", "MB00000003"};
    String [] nrosInventarioResultantes  = new String[activosFijos.size()];

    int i = 0;
    for (ActivoFijoTangible activoFijoTangible : activosFijos)
        nrosInventarioResultantes[i++] = activoFijoTangible.getNoInventario();

    assertArrayEquals(nrosInventarioValidos, nrosInventarioResultantes);
}
```

Figura 20 Método testFindAllByDepartamneto de la clase ActivoFijoDaoTest

Para la realización de las pruebas al método (*findAllByDpto*) se hace uso del método *testFindAllByDpto()* perteneciente a la clase *ActivoFijoDaoTest*. Primero se ejecuta la funcionalidad a probar y los resultados son almacenados dentro de un arreglo. Luego, haciendo uso del método *assertArrayEquals (nrosInventarioValido, nrosInventarioResultantes)* el framework JUnit comprueba que cada uno de los resultados obtenidos coincide con los resultados esperados y muestra en una

ventana los resultados obtenidos. En caso de obtener resultados satisfactorios para todas las pruebas realizadas se observa una línea verde en la ventana, en caso contrario aparece una línea roja. En la figura siguiente aparecen los resultados de las pruebas realizadas.

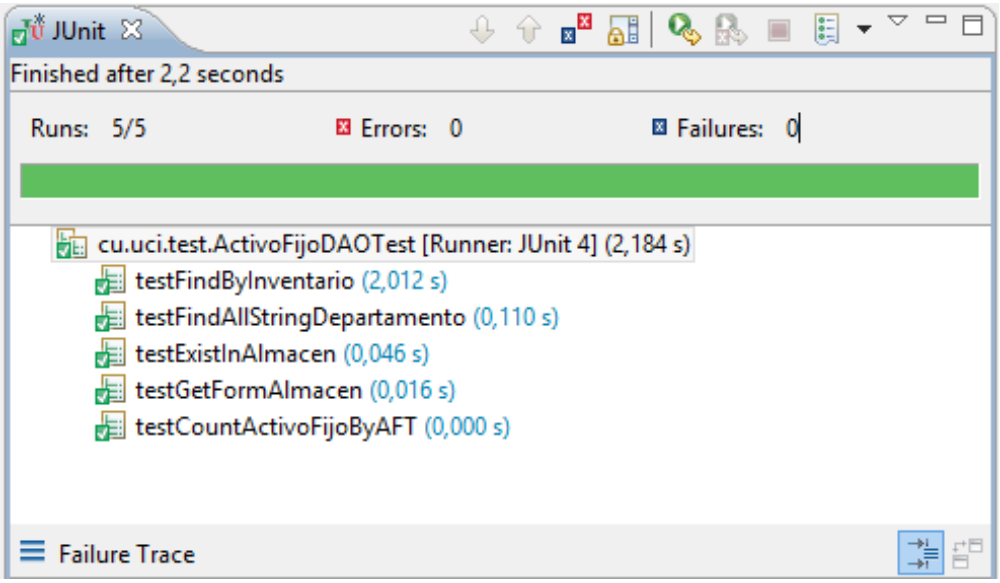


Figura 21 Resultado de las pruebas de caja blanca con JUnit

3.6. Conclusiones parciales

Luego de desarrollado el epígrafe se puede concluir que:

- La utilización de librerías y facilidades que ofrecen los *frameworks* definidos para el desarrollo del trabajo hizo más fácil la implementación del sistema ahorrando código y tiempo.
- Las pruebas realizadas fueron satisfactorias, demostrando la calidad de la solución que se propone y cumple con los requisitos funcionales definidos.
- La herramienta JUnit jugó un papel importante en la agilización de las pruebas realizadas, evidenciando un código limpio y funcional.

Conclusiones

Luego de realizado el trabajo, se arriban a las siguientes conclusiones.

- La elaboración de la fundamentación teórica de la investigación mediante la revisión documental permitió realizar un estudio sobre algunos sistemas informáticos nacionales e internacionales existentes que implementan sistemas de gestión de inventario de AFT. Se determinaron las características fundamentales de estos sistemas y las desventajas que presentan, permitiendo afirmar que ninguno de ellos constituye una solución factible para darle solución a la problemática plateada.
- Se diseñó el sistema automatizado de gestión de inventario de AFT, mediante el cual se generaron los artefactos más significativos propuestos por la metodología de desarrollo OpenUP. Para el diseño se tomaron como entrada los requisitos definidos en la etapa inicial del desarrollo de trabajo.
- Se implementaron todas las funcionalidades definidas para el sistema de gestión de inventario de AFT, utilizando para ello las herramientas y lenguajes definidos. El uso de estas herramientas permitió el ahorro de código y tiempo.
- Se realizaron pruebas a las funcionalidades del sistema de gestión de inventario de AFT, verificando la calidad de la solución que se propuso. Además se comprobó la funcionalidad del software, pudiendo constatar que este cumple con todas las funcionalidades definidas y está apto para ser puesto en explotación.

Recomendaciones

Se recomienda para futuras investigaciones:

- Añadir al sistema un módulo que permita la configuración de salvas, de forma tal que se realicen las siguientes actividades:
  - Configurar parámetros de salva
  - Realizar salva de la base de datos
  - Restaurar salvas de la base de datos

Esto permitirá que desde el propio sistema se puedan asegurar los datos ante cualquier fallo sin necesidad de realizar la tarea directamente en la base de datos. Para ello se deben establecer permisos que solo tendrá el rol de administrador del sistema, como una forma de mantener la seguridad, la confidencialidad y la integridad de los datos.

Glosario de Términos

AFT: Activo Fijo Tangible.

Activo: Bien tangible o intangible que posee una empresa o persona natural. Por extensión se considera activo a aquellos bienes o derechos que tienen un beneficio económico a futuro y se pueda gozar de los beneficios económicos que otorgue.

Activo fijo: Activo que no varían durante el ciclo de explotación de la empresa (o el año fiscal). Por ejemplo, el edificio donde una fábrica monta sus productos es un activo fijo porque permanece en la empresa durante todo el proceso de producción y venta de los productos.

Activo fijo tangible: Activo fijo que tiene vida relativamente larga y que se utiliza en la producción o venta de otros activos o servicios. Comprende las propiedades o bienes susceptibles de ser tocado, tales como los mobiliarios, la maquinaria etc.

Administrador: Persona que tiene privilegios para determinadas funcionalidades del sistema.

Caso de uso: Operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

Clase: Elemento de software que describe o caracteriza una entidad del mundo real o un entidad del espacio de implementación.

Centro de costo: Área de la empresa que tienen manejo y control sobre el consumo de recursos (material, mano de obra, etc.). En los centros de costos no se toman decisiones sobre ventas o cantidad de activos. El informe mediante el cual se evalúan los centros de costos es el Informe de costos.

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

Herramientas CASE: Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y de dinero.

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

Proceso: Conjunto de actividades relacionadas lógicamente que producen una salida o resultado.

Requisito Funcional: Requisito que especifica una acción que debe ser capaz de realizar el sistema, sin considerar restricciones físicas. Requisito que especifica comportamiento de entrada/salida de un sistema.

Requisito No Funcional: Requisito que especifica propiedades del sistema, como restricciones del entorno o de implementación, rendimiento, dependencias de la plataforma, extensibilidad o fiabilidad. Requisito que especifica restricciones físicas sobre un requisito funcional.

Referencias bibliográficas

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

2. **Tapia, Illescas, Bermeo Morejón, Andrea del Cisne y Viviana, Nube.** Gestión de inventarios en la empresa Decortextiles. *Repositorio Digital de la Universidad Politécnica Salesiana de Ecuador.* [En línea] 2007. [Citado el: 8 de Mayo de 2014.] http://dspace.ups.edu.ec/bitstream/123456789/743/4/Capitulo_III.pdf.

3. Implementación SIG, Sistema Integrado de Gestión. [En línea] [Citado el: 02 de febrero de 2014.] www.implementacionsig.com/index.php/23-noticiac/28-que-es-un-sistema-de-gestion.

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

5. **Planche El, Odalis y León Domínguez, Bernardo.** *Manual "Temas de contabilidad ".* La Habana : s.n., 2004.

6. Gobierno de España. Ministerio de Hacienda y administración pública. [En línea] [Citado el: 10 de Enero de 2014.] http://www.oficinavirtual.pap.minhap.gob.es/sitios/oficinavirtual/es-ES/CatalogoSistemasInformacion/sorolla2/Paginas/ModuloInventario.aspx.

7. PAE. Portal de Administación Electrónica. *Red Sara.* [En línea] [Citado el: 11 de mayo de 2014.] http://administracionelectronica.gob.es/ctt/verPestanaGeneral.htm?idIniciativa=redsara.

8. CS-Almacenes. Software CS-Almacenes. [En línea] [Citado el: 04 de 02 de 2014.] http://www.softwareseleccion.com/cs+almacenes-p-3197.

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

10. **Fuentes, Indira Lilled Laurencio.** *Modelado de una aplicación Web para el módulo inventario del sistema Cóndor.* La Habana. Universidad de las Ciencias Informáticas : s.n., 2008.

11. **González Brito, Henry Raúl y Lezcano Lozada, Yunieski.** *Sistema de Inventario Participativo en la UCI.* Universidad de las Ciencias Informáticas. La Habana. : s.n., 2005.

12. **Figueroa, Roberth G. y Solís, Camilo J.** Metodologías Tradicionales vs. Metodologías Ágiles. [En línea] [Citado el: 10 de diciembre de 2011.] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.

13. **José H. Canós, Patricio Letelier y María Carmen Penadés.** *Métodologías Ágiles en el Desarrollo de Software.* DSIC -Universidad Politécnica de Valencia : s.n.

14. **Amaro Calderón, Jorge Carlos Sarah Dámaris Valverde Rebaza.** *Metodologías Ágiles.* Trujillo. Perú : s.n., 2007.

15. **Eclipse Fundation.** OpenUP. [En línea] [Citado el: 20 de febrero de 2012.] http://epf.eclipse.org/wikis/openup/.

16. **Caro Salinas, Patricio.** Departamento de Ciencias de la Computación. Universidad de Chile. [En línea] [Citado el: 06 de febrero de 2014.] http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.

17. Rational Rose Enterprise. Software de IBM. [En línea] [Citado el: 04 de febrero de 2014.] http://www-03.ibm.com/software/products/es/enterprise/).

18. UML CASE Tools - Free for Learning UML, Cost-Effective for Business Solutions. [En línea] [Citado el: 01 de febrero de 2014.] www.visual-paradigm.com.

19. slideshare.net. [En línea] [Citado el: 03 de febrero de 2014.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

20. searchsoftwarequality.techtarget. [En línea] [Citado el: 06 de febrero de 2014.] http://searchsoftwarequality.techtarget.com/sDefinition/0,,sid92_gci754848,00.html..

21. Eclipse Fundation. [En línea] [Citado el: 05 de febrero de 2014.] http://epf.eclipse.org/wikis/openup/.

22. JasperSoft Comunity. [En línea] [Citado el: 7 de Mayo de 2014.] http://community.jaspersoft.com.

23. Portal en español sobre postgres SQL. [En línea] [Citado el: 06 de febrero de 2014.] http://www.postgresql.org.es/sobre_postgresql.

24. **Sánchez , Jorge.** *Java2 incluye Swing, Threads, programación en red, JavaBeans, JDBC y JSP / Servlets.* 2004.

25. **Rosabal Espinosa, Gleidis Yurinnis.** *Desarrolo del Módulo Supervisión dle Sistema de Gestión Penitenciario SIGEP.* La Haban. Universidad de las Ciencias Informáticas : s.n., 2011.

26. Oracle Documentation. *Project Swing (JavaTM Foundation Classes).* [En línea] [Citado el: 06 de febrero de 2014.] http://docs.oracle.com/javase/1.5.0/docs/guide/swing/ Project Swing.

27. **Somerville, Ian.** *Softweare Engineering.* 2006.

28. **Metsker, Steven John.** *Design Patterns Java Workbook.* s.l. : Addison Wesley, 2002.

29. **Reynoso , Carlos y Kiccillof, Nicólas .** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

30. **Zorrilla Pantaleon, Martha E.** página personal Dra. Marta Elena Zorrilla Pantaleón. *Modelo de datos.* [En línea] 2010-1011. [Citado el: 4 de 6 de 2014.] http://personales.unican.es/zorrillm/BasesDatos/ModelosdedatosER-UML-relacional.pdf.

31. Conferencia 7 ISW 2. Sobre la disciplina ingeniería de software. *Eva. Entorno Virtual de Aprendizaje.* [En línea] 2010-2011. [Citado el: 12 de mayo de 2014.] http://eva.uci.cu/mod/resource/view.php?id=9065.

32. **Pressman, Roger S.** *Ingeniería de software, un enfoque práctico.* 2002. Quinta edición.

Anexos

Anexo.1 Especificación y resumen de los casos de uso

Autenticar usuario

Objetivo	Identificar usuario para la entrada al sistema
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor especificado desea acceder a al sistema. El mismo introduce su usuario y contraseña y en vista del rol y permisos que tenga definido accede a las funcionalidades específicas según los privilegios que le fueron asignados.
Complejidad	Baja
Prioridad	Baja
Precondiciones	El usuario debe estar previamente creado en la base de datos
Postcondiciones	El usuario accede al sistema

Flujo de eventos		
Flujo básico Consultar expresiones de búsqueda		
	Actor	**Sistema**
	Accede al sistema	
		Brinda la posibilidad de introducir los datos elementales de búsqueda: • Usuario • Contraseña
		Permite: • Aceptar • Cancelar
	Introduce los datos y presiona el botón aceptar	
		Valida los datos.

		Termina el caso de uso.

Flujos alternos		
Nº *a Selecciona la opción que le permite cancelar		
	Actor	**Sistema**
4.		Regresa a la vista anterior.
5.		Termina el caso de uso

Flujos alternos		
Nº 5a Validación de campos vacíos en una consulta.		
	Actor	**Sistema**
4.		Muestra un mensaje de información: “No debe dejar campos en blanco”.
5.		Regresa al paso 4 del flujo básico.

Flujos alternos		
Nº 5b El usuario o la contraseña es incorrecta		
	Actor	**Sistema**
1.		Muestra un mensaje de información: “Datos de inicio de sesión incorrectos”.
2.		Regresa al paso 4 del flujo básico.

Prototipo elemental de interfaz gráfica de usuario

Autenticar usuario

Consultar trazas del sistema

Objetivo	Consultar trazas del sistema
Actores	Administrador del sistema
Resumen	El caso de uso se inicia cuando el actor especificado desea consultar las diferentes acciones realizadas en el sistema de la base de datos. Finaliza con la visualización de las trazas registradas.
Complejidad	Baja
Prioridad	Baja
Precondiciones	El usuario debe estar autenticado en el sistema y contar con los permisos para realizar dicha operación
Postcondiciones	
Flujo de eventos	
Flujo básico Consultar expresiones de búsqueda	

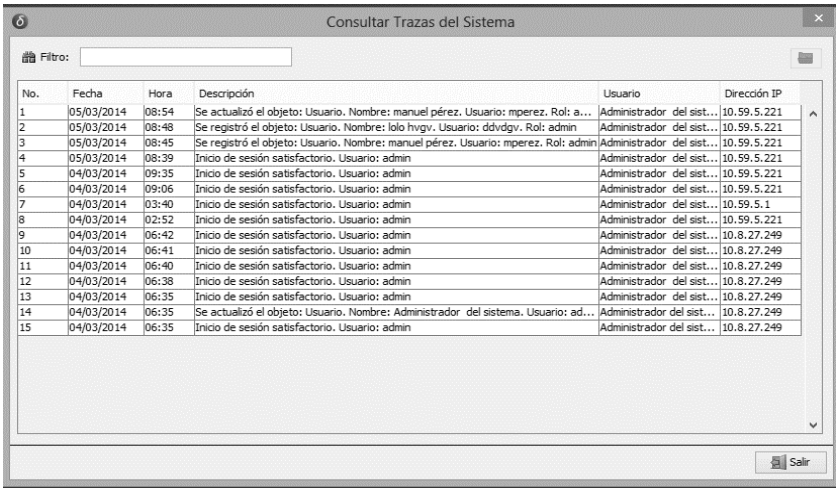
	Actor	**Sistema**
	Accede a la opción Consultar trazas	
		Muestra los elementos de las trazas: • Número • Fecha • Hora • Descripción • Dirección IP

		Permite: • Filtrar • Salir
	Introduce datos relacionados con los elementos registrados en el sistema	
		Muestra los datos que coinciden con los datos escritos.
		Termina el caso de uso.

Flujos alternos		
Nº *a Selecciona la opción que le permite salir		
	Actor	**Sistema**
6.		Regresa a la interfaz principal.
7.		Termina el caso de uso

Prototipo elemental de interfaz gráfica de usuario

Consultar trazas



Gestionar usuario

Objetivo	Registrar, modificar, filtrar y eliminar usuario.
Actores	Administrador del sistema: (Inicia) Registrar, modificar, filtrar y eliminar los datos del usuario
Resumen	El caso de uso se inicia cuando el actor especificado desea gestionar los usuarios del sistema. El caso de uso permite insertar, modificar o eliminar un usuario determinado, finalizando con la visualización del listado de los usuarios previamente gestionados.
Complejidad	Baja
Prioridad	Baja
Precondiciones	Debe haber roles registrados en el sistema.
Postcondiciones	Se registró, editó, eliminó o visualizó un usuario.
Flujo de eventos	
Flujo básico Registrar Usuario	

	Actor	**Sistema**
	Desea realizar las siguientes acciones.	
		Permite realizar las siguientes acciones sobre el usuario. • Registrar usuario • Modificar usuario. Ver Sección 1: “Modificar usuario” • Eliminar usuario. Ver Sección 2: “Eliminar usuario” • Cambiar Contraseña Ver Sección 3 “Cambiar contraseña de usuario”.
	Selecciona la opción que le permite registrar el usuario.	
		Permite introducir los siguientes datos del usuario:

		• Usuario • Nombre • Apellidos • Rol • Contraseña
	Introduce los datos del usuario.	
		Valida que se hayan llenado los campos Y permite: • Aceptar la operación • Cancelar la operación
	Selecciona la opción que le permite aceptar la operación.	
		Valida los datos.
		Registra los datos del usuario
		Muestra los datos del usuario.
		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
8.		Cancela las operaciones realizadas sobre la entidad.
9.		Termina el caso de uso.

Flujos alternos		
Nº 6a Campos vacíos.		
	Actor	**Sistema**
6.		Muestra un mensaje de error: “No debe

		dejar Campos en blanco".
7.		Regresa al paso 5 del flujo básico.

Flujos alternos		
Nº 6b Existe otro usuario con ese usuario		
	Actor	**Sistema**
8.		Muestra un mensaje de error: "El usuario que desea añadir ya está registrado en el sistema".
9.		Regresa al paso 5 del flujo básico.

Sección 1: "Modificar usuario"		
Flujo básico Modificar usuario		
	Actor	**Sistema**
9.	Selecciona la opción que le permite modificar el usuario.	
10.		Muestra los datos del usuario, y permite modificarlos, ver flujo básico: "Registrar Usuario". Y además permite: • Modificar los cambios • Cancelar la operación
11.	Selecciona la opción que le permite modificar los cambios.	
12.	Introduce los nuevos datos del usuario.	
13.		Valida los datos.
14.		Actualiza los cambios en el usuario.
15.		Muestra los datos de la usuario

16.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
3.		Cancela las operaciones realizadas sobre la entidad.
4.		Regresa al paso que le dio origen en el flujo básico.

Flujos alternos		
Nº 5a Campos vacíos.		
	Actor	**Sistema**
10.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
11.		Regresa al paso 4 del flujo básico.

Flujos alternos		
Nº 5b Existe otro usuario con ese usuario		
	Actor	**Sistema**
12.		Muestra un mensaje de error: “El usuario que desea añadir ya está registrado en el sistema”.
13.		Regresa al paso 4 del flujo básico.

Sección 2: “Eliminar usuario”

Flujo básico Eliminar usuario.		
	Actor	**Sistema**
8.	Selecciona el usuario a eliminar.	
9.	Selecciona la opción que le permite eliminar un usuario.	
10.		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”.
11.		Y permite: • Aceptar la operación • Cancelar la operación
12.	Selecciona la opción que le permite aceptar la operación.	
13.		Elimina el usuario.
14.		Termina el caso de uso.

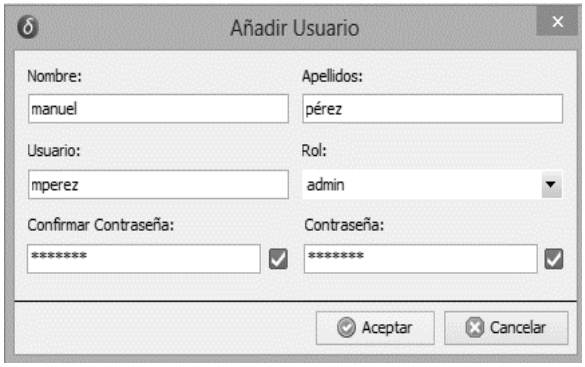
Flujos alternos		
Nº 5a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
5.		Sale de la vista actual sin eliminar el usuario seleccionado.
6.		Regresa a la vista anterior.
7.		Termina el caso de uso.

Sección 3: “Cambiar contraseña”		
Flujo básico Cambiar contraseña.		
	Actor	**Sistema**

1	Selecciona el usuario al que desea cambiar la contraseña	
2		Muestra la interfaz que permite Cambiar contraseña
3		*Ver ECU "Modificar Contraseña"*
4		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

Añadir Usuario



Modifica Usuario

Gestionar rol

Objetivo	Registrar, modificar, filtrar y eliminar rol de usuario.
Actores	Administrador del sistema: (Inicia) Registrar, modificar, filtrar y eliminar los datos del rol
Resumen	El caso de uso se inicia cuando el actor desea registrar, modificar, filtrar o eliminar el rol. Si el actor selecciona la opción de registrar, el sistema le permite especificar los datos y registrarlos. Si desea modificar, el sistema

	muestra los datos y permite modificar los datos requeridos. Si desea filtrar, el sistema muestra los datos y permite consultarlos según los criterios especificados El actor podrá eliminar el rol. Termina así el caso de uso.
Complejidad	Baja
Prioridad	Baja
Precondiciones	Debe haber permisos registrados en el sistema
Postcondiciones	Se registró, editó, eliminó o visualizó un rol de usuario.

Flujo de eventos

Flujo básico Registrar Rol

	Actor	Sistema
	Desea realizar las siguientes acciones.	
		Permite realizar las siguientes acciones sobre el rol. • Registrar rol • Modificar rol. Ver Sección 1: “Modificar rol” • Consultar rol. Ver Sección 2: “Consultar rol” • Eliminar rol. Ver Sección 3: “Eliminar rol”
	Selecciona la opción que le permite registrar el rol.	
		Permite introducir los siguientes datos del rol: • Nombre • Permisos definidos • Permisos asignados
	Introduce los datos del rol.	
		Valida que se hayan llenado los campos Y permite:

		• Aceptar la operación • Cancelar la operación
	Selecciona la opción que le permite aceptar la operación.	
		Valida los datos.
		Registra los datos del rol
		Muestra los datos del rol.
		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
10.		Cancela las operaciones realizadas sobre la entidad.
11.		Termina el caso de uso.

Flujos alternos		
Nº 6a Campos vacíos.		
	Actor	**Sistema**
14.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
15.		Regresa al paso 5 del flujo básico.

Flujos alternos		
Nº 6b No ha asignado los permisos al usuario		
	Actor	**Sistema**

16.		Muestra un mensaje de error: “Debe asignar al menos un permiso”.
17.		Regresa al paso 5 del flujo básico.

Flujos alternos		
Nº 6b Existe otro usuario con ese rol		
	Actor	**Sistema**
18.		Muestra un mensaje de error: “Existe un rol de usuario con igual nombre”.
19.		Regresa al paso 5 del flujo básico.

Sección 1: “Modificar rol”		
Flujo básico Modificar rol		
	Actor	**Sistema**
17.	Selecciona la opción que le permite modificar el rol.	
18.		Muestra los datos del rol, y permite modificarlos, ver flujo básico: “Registrar Rol”. Y además permite: • Modificar los cambios • Cancelar la operación
19.	Selecciona la opción que le permite modificar los cambios.	
20.	Introduce los nuevos datos del rol.	
21.		Valida los datos.
22.		Actualiza los cambios en el rol.
23.		Muestra los datos de la rol

24.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
5.		Cancela las operaciones realizadas sobre la entidad.
6.		Regresa al paso que le dio origen en el flujo básico.
Flujos alternos		
Nº 5a Campos vacíos.		
	Actor	**Sistema**
20.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
21.		Regresa al paso 4 del flujo básico.

Sección 2: “Consultar rol”		
Flujo básico Filtrar rol		
	Actor	**Sistema**
1.	Selecciona la opción que permite consultar el rol	
2.		Muestra los datos del rol • Nombre • Permisos definidos • Permisos asignados
3.	Selecciona la opción que permite salir de la vista actual.	
4.		Sale de la vista actual.

5.		Termina el caso de uso.
3c El actor selecciona la opción que le permite modificar el rol.		
	Actor	**Sistema**
1.		Permite modificar los datos del rol, ver Sección 1: “Modificar rol”.
2.		Termina el caso de uso.

Flujos alternos		
3c El actor selecciona la opción que le permite eliminar el rol.		
	Actor	**Sistema**
1.		Permite eliminar los datos del rol, ver Sección 3: “Eliminar rol”.
2.		Termina el caso de uso.

Sección 3: “Eliminar rol”		
Flujo básico Eliminar rol.		
	Actor	**Sistema**
15.	Selecciona el rol a eliminar.	
16.	Selecciona la opción que le permite eliminar un rol.	
17.		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”.
18.		Y permite: • Aceptar la operación • Cancelar la operación

19.	Selecciona la opción que le permite aceptar la operación.	
20.		Elimina el rol.
21.		Termina el caso de uso.

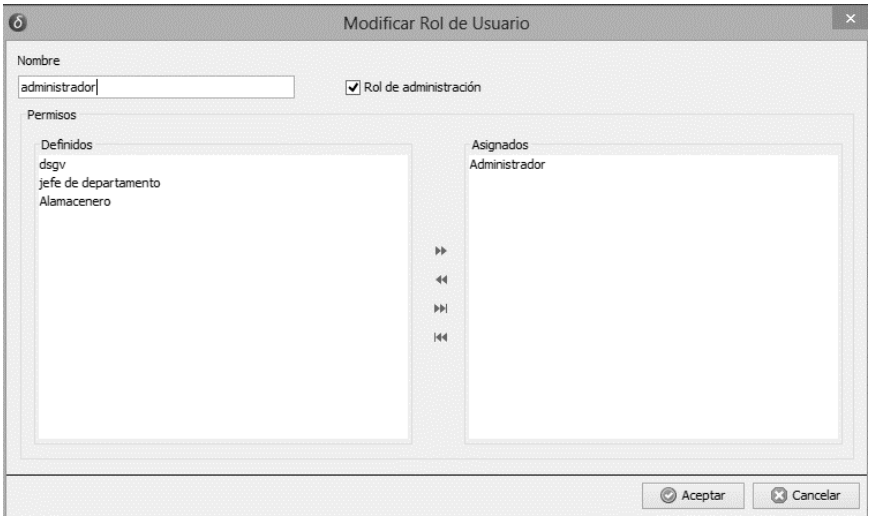
Flujos alternos		
Nº 5a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
8.		Sale de la vista actual sin eliminar el rol seleccionado.
9.		Regresa a la vista anterior.
10.		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

Añadir Rol de Usuario

Consultar Rol de Usuario

Modifica Rol de Usuario



Modificar contraseña

Objetivo	Cambiar la contraseña del usuario
Actores	Usuario
Resumen	El caso de uso se inicia cuando el actor especificado desea cambiar su contraseña. El caso de uso permite modificar la contraseña previamente definida del usuario autenticado. El caso de uso finaliza con la visualización de la ventana principal de la aplicación.
Complejidad	Baja

Prioridad	Baja
Precondiciones	El usuario debe estar autenticado en el sistema
Postcondiciones	Debe quedar almacenado en la base de datos la nueva contraseña de forma encriptada.

Flujo de eventos		
Flujo básico Consultar expresiones de búsqueda		
	Actor	**Sistema**
37.	Accede al sistema	
38.		Brinda la posibilidad de modificar los datos elementales: • Contraseña actual • Nueva contraseña • Confirmar contraseña
39.		Permite: • Aceptar • Cancelar
40.	Introduce los datos y presiona el botón aceptar	
41.		Valida los datos.
42.		Se muestra la ventana principal del sistema
43.		Termina el caso de uso.

Flujos alternos		
Nº *a Selecciona la opción que le permite cancelar		
	Actor	**Sistema**
12.		Regresa a la vista anterior.

13.		Termina el caso de uso

Flujos alternos		
Nº 5a Existen campos vacíos.		
	Actor	**Sistema**
22.		Muestra deshabilitado el botón aceptar.
23.		Permanece en la misma interfaz.

Flujos alternos		
Nº 5b Las contraseñas no coinciden		
	Actor	**Sistema**
3.		Muestra deshabilitado el botón aceptar.
4.		Permanece en la misma interfaz.

Flujos alternos		
Nº 5b La contraseña actual es incorrecta		
	Actor	**Sistema**
5.		Muestra un mensaje de error “Datos de inicio de sesión incorrectos”.
6.		Permanece en la misma interfaz.

Flujos alternos		
Nº 5b La nueva contraseña no es válida (no tiene letras y números)		
	Actor	**Sistema**
7.	Muestra deshabilitado el botón aceptar.	Muestra deshabilitado el botón aceptar.

8.		Permanece en la misma interfaz.

Prototipo elemental de interfaz gráfica de usuario

Modificar contraseña

Gestionar trabajador

Objetivo	Registrar, modificar, filtrar y eliminar el trabajador.
Actores	Jefe de Almacén: (Inicia) Registrar, modificar, filtrar y eliminar los datos del trabajador
Resumen	El caso de uso se inicia cuando el actor desea registrar, modificar, filtrar o eliminar el trabajador. Si el actor selecciona la opción de registrar, el sistema le permite especificar los datos y registrarlos. Si desea modificar, el sistema muestra los datos y permite modificar los datos requeridos. Si desea filtrar, el sistema muestra los datos y permite consultarlos según los criterios especificados El actor podrá eliminar el trabajador. Termina así el caso de uso.
Complejidad	Baja
Prioridad	Baja
Precondiciones	Debe haber registrado departamentos en el sistema
Postcondiciones	Se registró, editó, eliminó o visualizó un trabajador.
Flujo de eventos	
Flujo básico Registrar Trabajador	

	Actor	Sistema
44.	Desea realizar las siguientes acciones.	
45.		Permite realizar las siguientes acciones sobre el trabajador. • Registrar una trabajadores • Modificar el trabajador. Ver Sección 1: “Modificar trabajadores” • trabajadores. Ver Sección 2: “Filtrar trabajadores” • Eliminar trabajadores. Ver Sección 3: “Eliminar trabajadores”
46.	Selecciona la opción que le permite registrar el trabajador.	
47.		Permite introducir los siguientes datos del trabajadores: • Nombre • Apellidos • Ocupación • Departamento
48.	Introduce los datos del trabajador.	
49.		Valida que se hayan llenado los campos Y permite: • Aceptar la operación • Cancelar la operación
50.	Selecciona la opción que le permite aceptar la operación.	
51.		Valida los datos.
52.		Registra los datos del trabajador
53.		Muestra los datos del trabajador.
54.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
14.		Cancela las operaciones realizadas sobre la entidad.
15.		Termina el caso de uso.

Flujos alternos		
Nº 6a Campos vacíos.		
	Actor	**Sistema**
24.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
25.		Regresa al paso 5 del flujo básico.

Flujos alternos		
Nº 8a Código con menos de 5 dígitos.		
	Actor	**Sistema**
1.		Muestra un mensaje de error: “El código debe tener 5 dígitos”.
2.		Regresa al paso 5 del flujo básico.

Sección 1: “Modificar trabajador”		
Flujo básico Modificar trabajador		
	Actor	**Sistema**
25.	Selecciona la opción que le permite modificar el trabajador.	

26.		Muestra los datos del trabajador, y permite modificarlos, ver flujo básico: “Registrar Trabajadores”. Y además permite: • Modificar los cambios • Cancelar la operación
27.	Selecciona la opción que le permite modificar los cambios.	
28.	Introduce los nuevos datos del trabajador.	
29.		Valida los datos.
30.		Actualiza los cambios en el trabajador.
31.		Muestra los datos del trabajador
32.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
7.		Cancela las operaciones realizadas sobre la entidad.
8.		Regresa al paso que le dio origen en el flujo básico.

Flujos alternos		
Nº 5a Campos vacíos.		
	Actor	**Sistema**
26.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
27.		Regresa al paso 4 del flujo básico.

Flujos alternos		
Nº 5b Código con menos de 5 dígitos.		
	Actor	**Sistema**
3.		Muestra un mensaje de error: “El código debe tener 5 dígitos”.
4.		Regresa al paso 4 del flujo básico.

Sección 2: “Filtrar trabajador”		
Flujo básico Filtrar trabajador		
	Actor	**Sistema**
6.	Introduce en el campo filtro algún código o parte de una descripción del trabajador.	
7.		Muestra los datos del trabajador según coincidencias con los datos registrados. • Nombre • Apellidos • Ocupación • Departamento
8.	Selecciona la opción que permite salir de la vista actual.	
9.		Sale de la vista actual.
10.		Termina el caso de uso.
Flujos alternos		
3c El actor selecciona la opción que le permite modificar el trabajador.		
	Actor	**Sistema**
1		Permite modificar los datos del trabajador, ver Sección 1: “Modificar trabajadores”.

2		Termina el caso de uso.

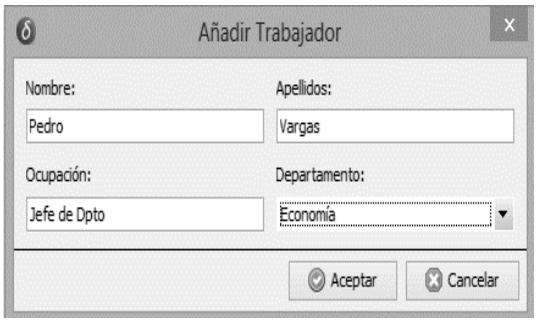
Flujos alternos		
3c El actor selecciona la opción que le permite eliminar el trabajador.		
	Actor	**Sistema**
1		Permite eliminar los datos del trabajador, ver Sección 3: “Eliminar trabajador”.
2		Termina el caso de uso.

Sección 3: “Eliminar trabajador”		
Flujo básico Eliminar trabajador.		
	Actor	**Sistema**
22.	Selecciona el trabajador a eliminar.	
23.	Selecciona la opción que le permite eliminar un proveedor.	
24.		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”.
25.		Y permite: • Aceptar la operación • Cancelar la operación
26.	Selecciona la opción que le permite aceptar la operación.	
27.		Elimina el proveedor.
28.		Termina el caso de uso.

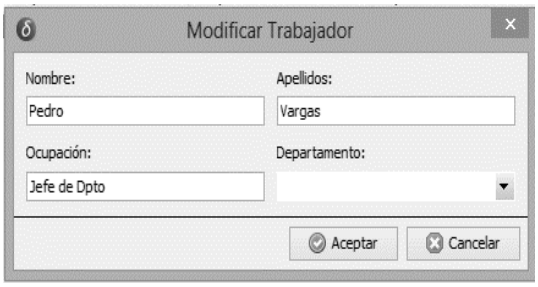
Flujos alternos		
Nº 5a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
11.		Sale de la vista actual sin eliminar el trabajador seleccionado.
12.		Regresa a la vista anterior.
13.		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

Registrar Trabajador



Modificar Trabajador



Gestionar concepto para baja de AFT

Objetivo	Registrar, modificar, filtrar y eliminar el concepto para baja de AFT.
Actores	Jefe de Almacén: (Inicia) Registrar, modificar, filtrar y eliminar los datos del concepto para baja de AFT
Resumen	El caso de uso se inicia cuando el actor desea registrar, modificar, filtrar

	o eliminar el concepto para baja de AFT. Si el actor selecciona la opción de registrar, el sistema le permite especificar los datos y registrarlos. Si desea modificar, el sistema muestra los datos y permite modificar los datos requeridos. Si desea filtrar, el sistema muestra los datos y permite consultarlos según los criterios especificados El actor podrá eliminar el concepto para baja de AFT. Termina así el caso de uso.
Complejidad	Baja
Prioridad	Baja
Precondiciones	
Postcondiciones	Se registró, editó, eliminó o visualizó un concepto para baja de AFT.

Flujo de eventos

Flujo básico Registrar Concepto para baja de AFT

	Actor	Sistema
55.	Desea realizar las siguientes acciones.	
56.		Permite realizar las siguientes acciones sobre el concepto para baja de AFT. • Registrar una concepto para baja de AFT • Modificar el concepto para baja de AFT. Ver Sección 1: “Modificar concepto para baja de AFT” • concepto para baja de AFT. Ver Sección 2: “Filtrar concepto para baja de AFT” • Eliminar concepto para baja de AFT. Ver Sección 3: “Eliminar concepto para baja de AFT”
57.	Selecciona la opción que le permite registrar el concepto para baja de AFT.	
58.		Permite introducir los siguientes datos del concepto para baja de AFT: • Código

		• Descripción
59.	Introduce los datos del concepto para baja de AFT.	
60.		Valida que se hayan llenado los campos Y permite: • Aceptar la operación • Cancelar la operación
61.	Selecciona la opción que le permite aceptar la operación.	
62.		Valida los datos.
63.		Registra los datos del concepto para baja de AFT
64.		Muestra los datos del concepto para baja de AFT.
65.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
16.		Cancela las operaciones realizadas sobre la entidad.
17.		Termina el caso de uso.

Flujos alternos		
Nº 6a Campos vacíos.		
	Actor	**Sistema**
28.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.

29.		Regresa al paso 5 del flujo básico.

Flujos alternos		
Nº 8a Código con menos de 5 dígitos.		
	Actor	**Sistema**
5.		Muestra un mensaje de error: “El código debe tener 5 dígitos”.
6.		Regresa al paso 5 del flujo básico.

Sección 1: “Modificar concepto para baja de AFT”		
Flujo básico Modificar concepto para baja de AFT		
	Actor	**Sistema**
33.	Selecciona la opción que le permite modificar el concepto para baja de AFT.	
34.		Muestra los datos del concepto para baja de AFT, y permite modificarlos, ver flujo básico: “Registrar Concepto para baja de AFT”. Y además permite: • Modificar los cambios • Cancelar la operación
35.	Selecciona la opción que le permite modificar los cambios.	
36.	Introduce los nuevos datos del concepto para baja de AFT.	
37.		Valida los datos.
38.		Actualiza los cambios en el concepto para baja de AFT.
39.		Muestra los datos del concepto para baja de AFT

40.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
9.		Cancela las operaciones realizadas sobre la entidad.
10.		Regresa al paso que le dio origen en el flujo básico.

Flujos alternos		
Nº 5a Campos vacíos.		
	Actor	**Sistema**
30.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
31.		Regresa al paso 4 del flujo básico.

Flujos alternos		
Nº 5b Código con menos de 5 dígitos.		
	Actor	**Sistema**
7.		Muestra un mensaje de error: “El código debe tener 5 dígitos”.
8.		Regresa al paso 4 del flujo básico.

Sección 2: “Filtrar concepto para baja de AFT”		
Flujo básico Filtrar concepto para baja de AFT		
	Actor	**Sistema**

11.	Introduce en el campo filtro algún código o parte de una descripción del concepto para baja de AFT.	
12.		Muestra los datos del concepto para baja de AFT según coincidencias con los datos registrados. • Código • Descripción
13.	Selecciona la opción que permite salir de la vista actual.	
14.		Sale de la vista actual.
15.		Termina el caso de uso.

Flujos alternos		
3c El actor selecciona la opción que le permite modificar el concepto para baja de AFT.		
	Actor	**Sistema**
1.		Permite modificar los datos del concepto para baja de AFT, ver Sección 1: “Modificar Concepto para baja de AFT”.
2.		Termina el caso de uso.

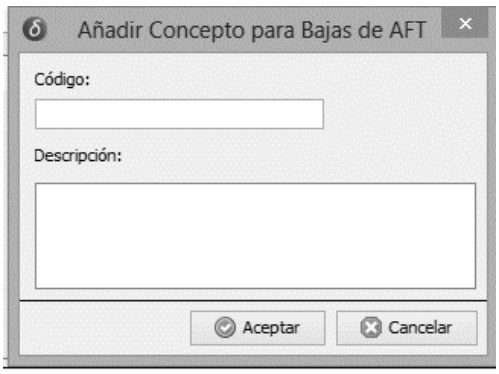
Flujos alternos		
3c El actor selecciona la opción que le permite eliminar el concepto para baja de AFT.		
	Actor	**Sistema**
1		Permite eliminar los datos del concepto para baja de AFT, ver Sección 3: “Eliminar Concepto para baja de AFT”.
2		Termina el caso de uso.

Sección 3: “Eliminar concepto para baja de AFT”		
Flujo básico Eliminar concepto para baja de AFT.		
	Actor	**Sistema**
29.	Selecciona el concepto para baja de AFT a eliminar.	
30.	Selecciona la opción que le permite eliminar un proveedor.	
31.		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”.
32.		Y permite: • Aceptar la operación • Cancelar la operación
33.	Selecciona la opción que le permite aceptar la operación.	
34.		Elimina el proveedor.
35.		Termina el caso de uso.

Flujos alternos		
Nº 5a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
14.		Sale de la vista actual sin eliminar el concepto para baja de AFT seleccionados.
15.		Regresa a la vista anterior.
16.		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

Registrar Concepto para baja de AFT



Modificar Concepto para baja de AFT

Gestionar categoría

Objetivo	Registrar, modificar, filtrar y eliminar la categoría.
Actores	Jefe de Almacén: (Inicia) Registrar, modificar, filtrar y eliminar los datos de la categoría
Resumen	El caso de uso se inicia cuando el actor desea registrar, modificar, filtrar o eliminar la categoría. Si el actor selecciona la opción de registrar, el sistema le permite especificar los datos y registrarlos. Si desea modificar, el sistema muestra los datos y permite modificar los datos requeridos. Si desea filtrar, el sistema muestra los datos y permite consultarlos según los criterios especificados El actor podrá eliminar la categoría. Termina así el caso de uso.
Complejidad	Baja
Prioridad	Baja
Precondiciones	
Postcondiciones	Se registró, editó, eliminó o visualizó una categoría.

Flujo de eventos		
Flujo básico Registrar Categoría		
	Actor	**Sistema**
66.	Desea realizar las siguientes acciones.	
67.		Permite realizar las siguientes acciones sobre la categoría. • Registrar una categoría • Modificar la categoría. Ver Sección 1: “Modificar categoría” • categoría. Ver Sección 2: “Filtrar categoría” • Eliminar categoría. Ver Sección 3: “Eliminar categoría”
68.	Selecciona la opción que le permite registrar la categoría.	
69.		Permite introducir los siguientes datos de la categoría: • Código • Descripción
70.	Introduce los datos de la categoría.	
71.		Valida que se hayan llenado los campos Y permite: • Aceptar la operación • Cancelar la operación
72.	Selecciona la opción que le permite aceptar la operación.	
73.		Valida los datos.
74.		Registra los datos de la categoría
75.		Muestra los datos de la categoría.
76.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
18.		Cancela las operaciones realizadas sobre la entidad.
19.		Termina el caso de uso.

Flujos alternos		
Nº 6a Campos vacíos.		
	Actor	**Sistema**
32.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.
33.		Regresa al paso 5 del flujo básico.

Flujos alternos		
Nº 8a Código con menos de 5 dígitos.		
	Actor	**Sistema**
9.		Muestra un mensaje de error: “El código debe tener 5 dígitos”.
10.		Regresa al paso 5 del flujo básico.

Sección 1: “Modificar categoría”		
Flujo básico Modificar categoría		
	Actor	**Sistema**
41.	Selecciona la opción que le permite modificar	

	la categoría.	
42.		Muestra los datos de la categoría, y permite modificarlos, ver flujo básico: “Registrar Categoría”. Y además permite: • Modificar los cambios • Cancelar la operación
43.	Selecciona la opción que le permite modificar los cambios.	
44.	Introduce los nuevos datos de la categoría.	
45.		Valida los datos.
46.		Actualiza los cambios en la categoría.
47.		Muestra los datos de la categoría
48.		Termina el caso de uso.

Flujos alternos		
Nº *a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
11.		Cancela las operaciones realizadas sobre la entidad.
12.		Regresa al paso que le dio origen en el flujo básico.

Flujos alternos		
Nº 5a Campos vacíos.		
	Actor	**Sistema**
34.		Muestra un mensaje de error: “No debe dejar Campos en blanco”.

35.		Regresa al paso 4 del flujo básico.

Flujos alternos		
Nº 5b Código con menos de 5 dígitos.		
	Actor	**Sistema**
11.		Muestra un mensaje de error: “El código debe tener 5 dígitos”.
12.		Regresa al paso 4 del flujo básico.
Sección 2: “Filtrar categoría”		
Flujo básico Filtrar categoría		
	Actor	**Sistema**
16.	Introduce en el campo filtro algún código o parte de una descripción de la categoría.	
17.		Muestra los datos de la categoría según coincidencias con los datos registrados. • Código • Descripción
18.	Selecciona la opción que permite salir de la vista actual.	
19.		Sale de la vista actual.
20.		Termina el caso de uso.
Flujos alternos		
3c El actor selecciona la opción que le permite modificar la categoría.		
	Actor	**Sistema**
1		Permite modificar los datos de la categoría, ver Sección 1: “Modificar Categoría”.

2		Termina el caso de uso.

Flujos alternos		
3c El actor selecciona la opción que le permite eliminar la categoría.		
	Actor	**Sistema**
1		Permite eliminar los datos de la categoría, ver Sección 3: “Eliminar Categoría”.
2		Termina el caso de uso.

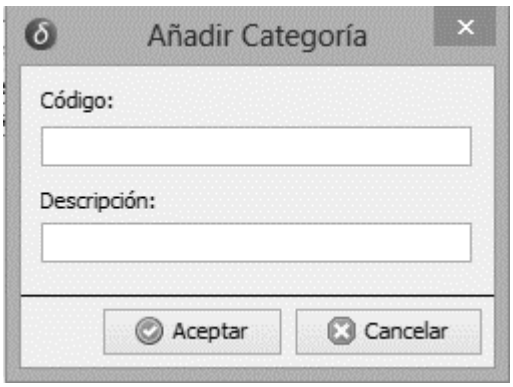
Sección 3: “Eliminar categoría”		
Flujo básico Eliminar categoría.		
	Actor	**Sistema**
36.	Selecciona la categoría a eliminar.	
37.	Selecciona la opción que le permite eliminar una categoría.	
38.		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”.
39.		Y permite: • Aceptar la operación • Cancelar la operación
40.	Selecciona la opción que le permite aceptar la operación.	
41.		Elimina la categoría.
42.		Termina el caso de uso.

Flujos alternos

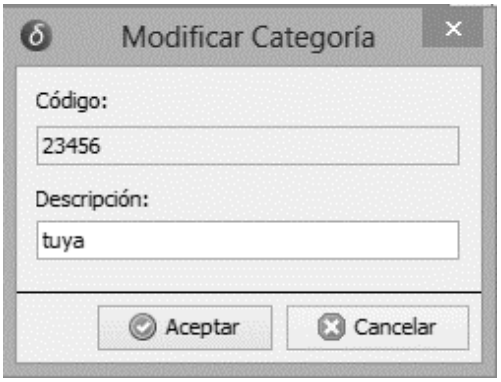
Nº 5a El actor selecciona la opción que le permite cancelar la operación.		
	Actor	**Sistema**
17.		Sale de la vista actual sin eliminar la categoría seleccionada.
18.		Regresa a la vista anterior.
19.		Termina el caso de uso.

Prototipo elemental de interfaz gráfica de usuario

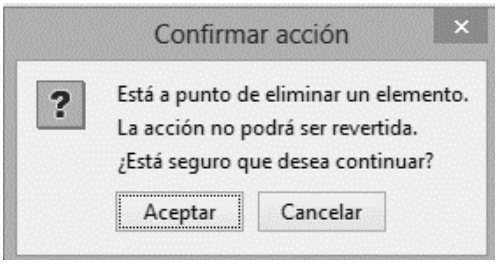
Registrar Categoría



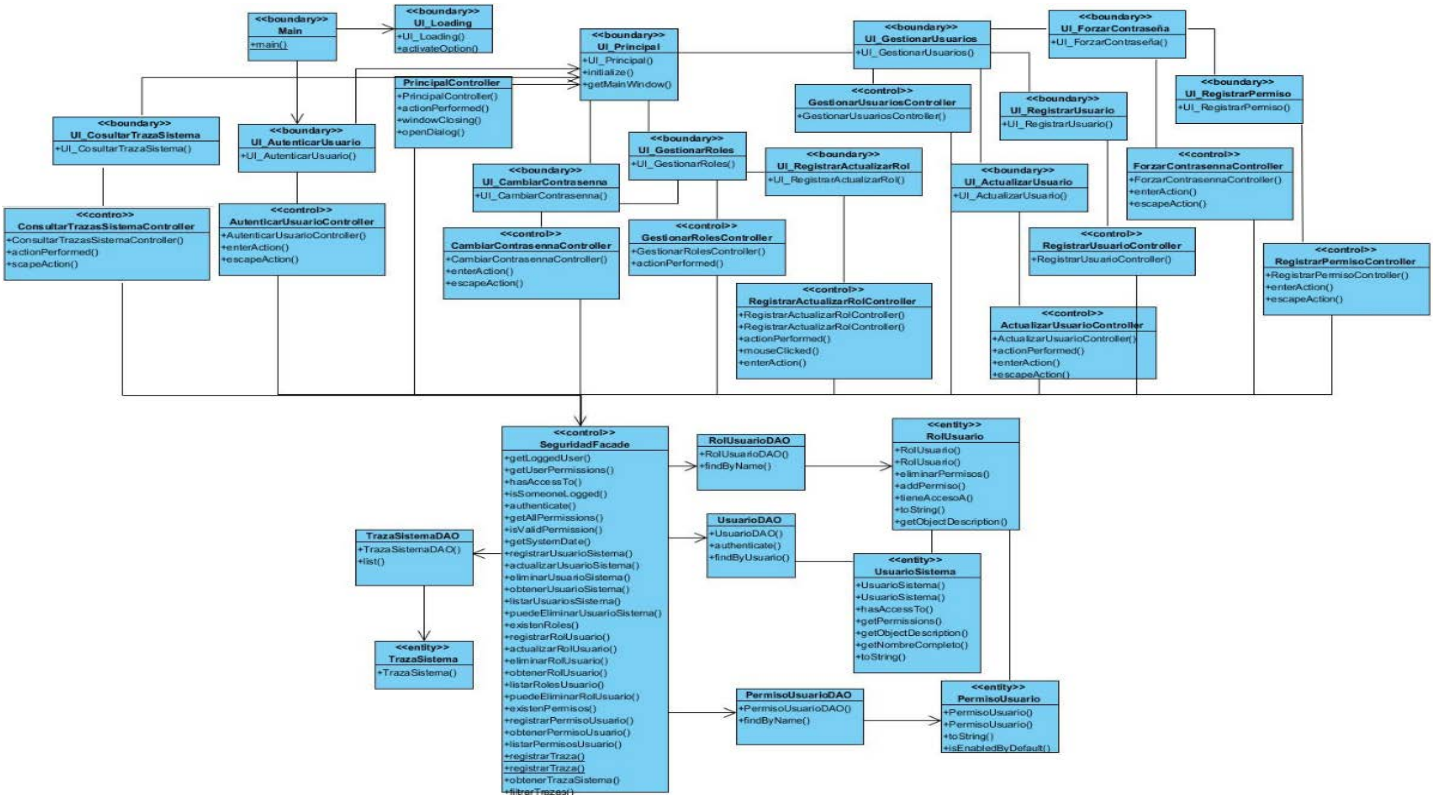
Modificar Categoría



Eliminar Categoría



Anexo 2. Diagrama de las clases de diseño del paquete Seguridad.



Anexo.3 Caso de prueba para el CU Gestionar Departamento

Descripción general

El caso de prueba es aplicado cuando se desea registrar, modificar, filtrar o eliminar el departamento. Si se selecciona la opción de registrar, el sistema le permite especificar los datos y registrarlos. Si desea modificar, el sistema muestra los datos y permite modificar los datos requeridos. Si desea filtrar, el sistema muestra los datos y permite consultarlos según los criterios especificados Finalmente se podrá eliminar el departamento.

Condiciones de ejecución

Precondiciones: El usuario debe estar anticuado y tener los permios necesarios para acceder a estas funcionalidades.

SC1 Registrar Departamento

Escenario	Descripción	Código	Nombre	Centro de costo	Responsable	Almacén	Nombre trabajador	Ocupación	Respuesta del sistema	Flujo central
EC 1.1 Registrar Departamento	Permite registrar los datos correspondientes a un departamento.	V 12345	V Economía	V Centro de costo 1	V Ramón Pérez	V Seleccionar campo Almacén	V Pepe Rosales	V Contador	Valida los datos introducidos. Registra los datos del departamento	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Introduce los datos del departamento. 5. Presiona el botón Aceptar
EC 1.2 Cancelar el registro del departamento.	Se cancela el registro del departamento.	V 12345	V Economía	V Centro de costo 1	V Ramón Pérez	V Seleccionar campo Almacén	V Pepe Rosales	V Contador	Cancela las operaciones realizadas sobre el departamento.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Introduce los datos del departamento. 5. Presiona el botón Cancelar

EC 1.3 Cancelar el registro del trabajador.	Se cancela el registro del trabajador.	NA	NA	NA	NA	NA	V Pepe Rosales	V Contador	Cancela el registro del trabajador.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Selecciona la opción que permite registrar un trabajador. 5. Introduce los datos del trabajador. 6. Presiona el botón Cancelar
EC 1.4 Campos vacíos para registrar departamento.	Permite validar que no existan campos vacíos para registrar un departamento.	I Campo vacío	I Campo vacío	I Campo vacío	I Campo vacío	I Campo vacío	I Campo vacío	I Campo vacío	Muestra un mensaje de error: “Todos los campos son requeridos”.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Presiona el botón Aceptar

EC 1.5 Eliminar trabajador.	Se elimina un trabajador.	NA	NA	NA	NA	NA	V Pepe Rosales	V Contador	Elimina el trabajador seleccionado.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Introduce los datos del departamento. 5. Selecciona la opción que permite añadir un trabajador. 6. Selecciona el trabajador que desee eliminar. 7. Selecciona la opción que permite eliminar. 8. Presiona el botón Aceptar
EC 1.6 No se añade ningún trabajador	Permite validar que al menos exista un trabajador en el departamento	NA	NA	NA	NA	NA	V Pepe Rosales	V Contador	Muestra un mensaje de error “Debe definir al menos un trabajador”.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Introduce los datos del departamento sin registrar al menos un trabajador. 5. Presiona el botón Aceptar/

SC2 Modificar Departamento

Escenario	Descripción	Código	Responsable	Descripción	Respuesta del sistema	Flujo central
EC 2.1 Modificar Departamento	Permite modificar los datos correspondientes a un departamento.	V 1E+09	V	V dato	Valida los datos introducidos. Actualiza los datos del departamento.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite actualizar el departamento. 4. Introduce los datos del departamento. 5. Presiona el botón Aceptar.
		V dato	V dato	V dato		
EC 2.2 Cancelar la actualización del departamento.	Se cancela la actualización de los datos del departamento.	NA	NA	NA	Cancela las operaciones realizadas sobre el departamento.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite actualizar el departamento. 4. Introduce los datos del departamento. 5. Presiona el botón Cancelar.

EC 2.3 Campos vacíos para modificar departamento.	Permite validar que no existan campos vacíos para modificar un departamento.	NA	NA	NA	Muestra un mensaje de error: “Todos los campos son requeridos”.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite actualizar el departamento. 5. Presiona el botón Aceptar.
EC 2.4 Eliminar trabajador.	Se elimina un trabajador.	NA	NA	NA	Elimina el trabajador seleccionado.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite actualizar el departamento. 4. Introduce los datos del departamento. 5. Selecciona el trabajador que desee eliminar. 6. Selecciona la opción que permite eliminar. 7. Presiona el botón Aceptar
EC 2.5 No se añade ningún trabajador	Permite validar que al menos exista un trabajador en el departamento	NA	NA	NA	Muestra un mensaje de error” Debe definir al menos un trabajador”.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite registrar el departamento. 4. Introduce los datos del

						departamento sin registrar al menos un trabajador. 5. Presiona el botón Aceptar/

SC3 Eliminar Departamento

Escenario	Descripción	Código	Responsable		Descripción		Respuesta del sistema	Flujo central
EC 3.1 Eliminar departamento.	Permite modificar los datos correspondientes a un departamento.	V 12458	V		V dato		Muestra un mensaje de advertencia: “Está a punto de eliminar un elemento. La acción no podrá ser revertida. ¿Está seguro que desea continuar?”. Elimina el departamento.	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona el departamento que desea eliminar. 5. Presiona el botón Aceptar.
		V dato	V dato		V dato			
EC 2.2 Cancelar la eliminación del departamento.	Se cancela la eliminación del departamento.	NA	NA		NA		No elimina el departamento y regresa a la interfaz principal Gestionar Departamento	1. Selecciona el menú Configuración. 2. Selecciona el submenú Gestionar Departamentos. 3. Selecciona la opción que le permite actualizar el departamento. 4. Introduce los datos del departamento. 5. Presiona el botón Cancelar.

Anexo.4 Diagramas de componentes.

Diagrama de componentes según las capas de la aplicación

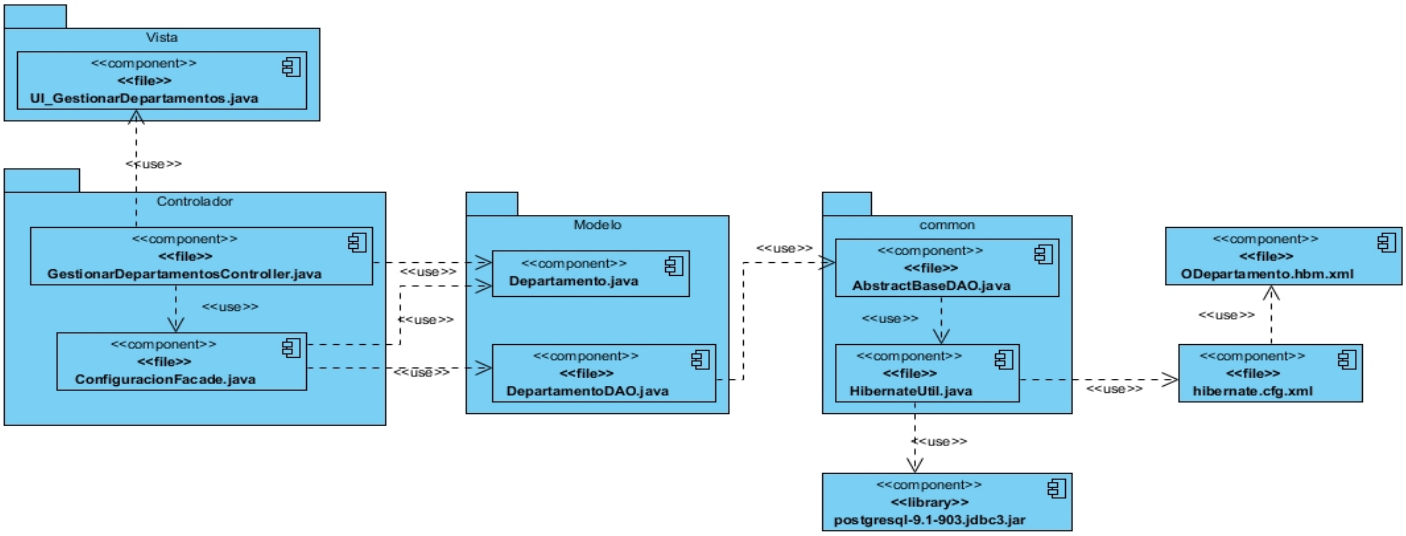


Diagrama de componentes del CU Gestionar categoría

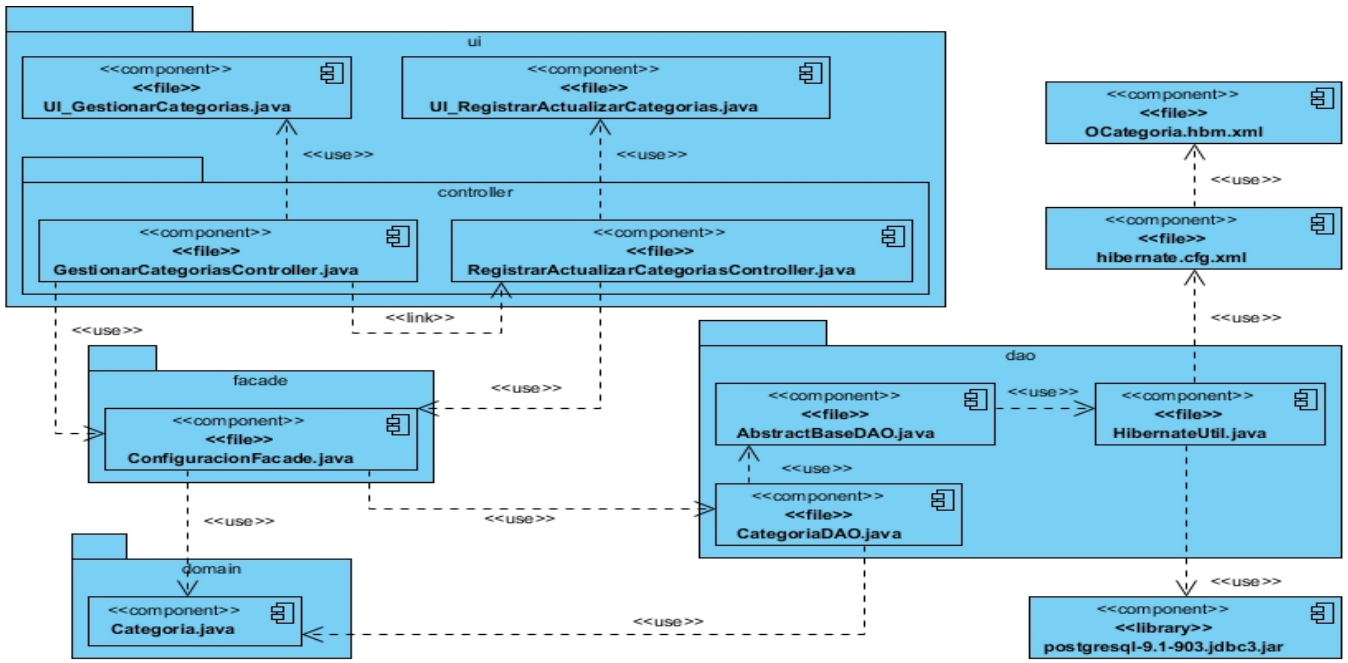
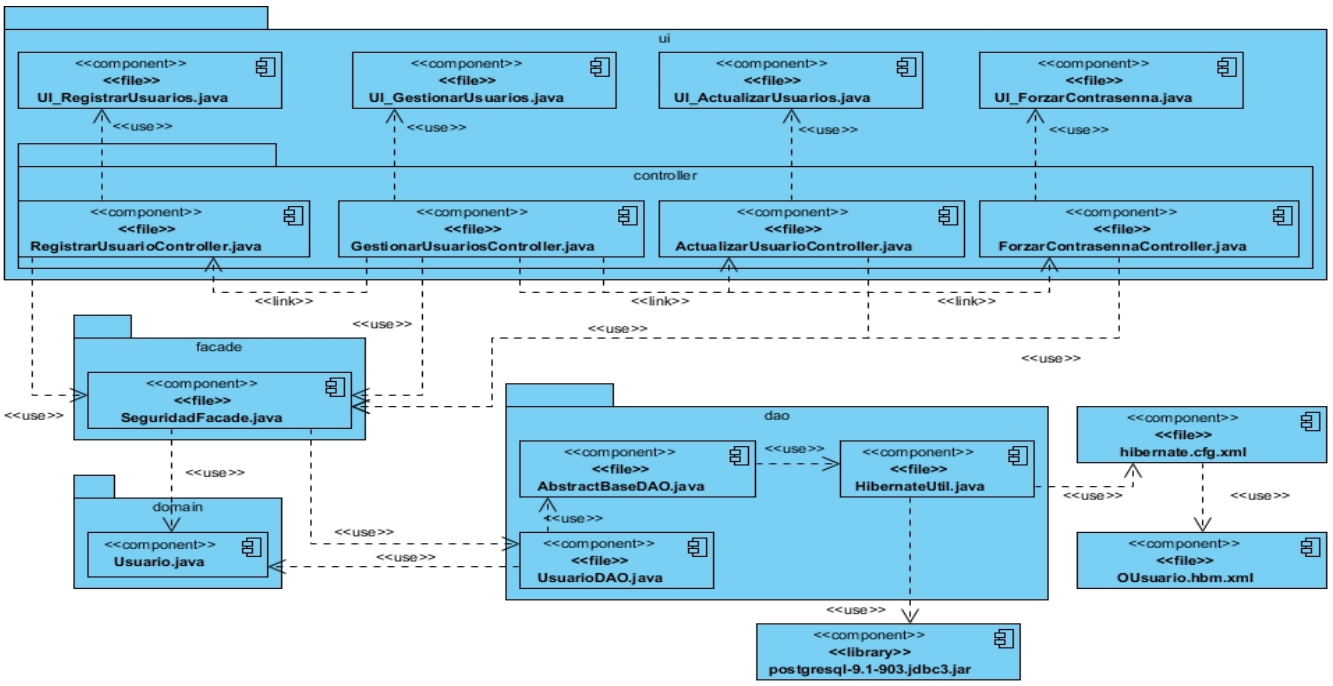


Diagrama de componentes del CU Gestionar usuario


ui
<<component>>
<<file>>
UI_RegistrarUsuarios.java
<<component>>
<<file>>
UI_GestionarUsuarios.java
<<component>>
<<file>>
UI_ActualizarUsuarios.java
<<component>>
<<file>>
UI_ForzarContrasenna.java
<<use>>
controller
<<component>>
<<file>>
RegistrarUsuarioController.java
<<component>>
<<file>>
GestionarUsuariosController.java
<<component>>
<<file>>
ActualizarUsuarioController.java
<<component>>
<<file>>
ForzarContrasennaController.java
<<link>>
facade
<<component>>
<<file>>
SeguridadFacade.java
dao
<<component>>
<<file>>
AbstractBaseDAO.java
<<component>>
<<file>>
HibernateUtil.java
<<component>>
<<file>>
hibernate.cfg.xml
domain
<<component>>
Usuario.java
<<component>>
<<file>>
UsuarioDAO.java
<<component>>
<<file>>
OUsuario.hbm.xml
<<component>>
<<library>>
postgresql-9.1-903.jdbc3.jar