Facultad 3

Título: Desarrollo de funcionalidades de los procesos de entrada y transferencias internas del subsistema inventario de Cedrux

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

Autores:

Diana Rodríguez Malagón

Diógenes Hernández Estévez

Tutores:

Ing. Javier Ramírez Hernández

Ing. Juan José Rosales Rodríguez

La Habana, Cuba, 2013

"Año del 54 Aniversario de la Revolución"

“En la carrera de la vida no importa en qué lugar llegues a la meta, lo importante es que sepas vencer cada obstáculo que se atraviesa en tu camino”.

Declaración de autoría

Declaramos que nosotros, **Diana Rodríguez Malagón y Diógenes Hernández Estévez**, somos los únicos autores de este trabajo y autorizamos a la Facultad 3 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente declaración de autoría en La Habana a los _____ días del mes de _____ del año _____.

_____ _____

Diana Rodríguez Malagón **Diógenes Hernández Estévez**

_____ _____

Ing. Javier Ramírez Hernández **Ing. Juan José Rosales Rodríguez**

Síntesis de los tutores:

Ing. Javier Ramírez Hernández:

Graduado de Ingeniero en Ciencias Informáticas en el año 2009. Se incorporó como desarrollador en la línea logística de Cedrux en CEIGE[1]. En el 2010 pasa a ocupar el rol de arquitecto. En 2011 instala y despliega el sistema de inventario para la Gran Misión Vivienda Venezuela y se desempeña como arquitecto del equipo. Ha impartido clases de Teleinformática y Debate Histórico Contemporáneo. Actualmente se desempeña como arquitecto de la línea logística, atendiendo los proyectos de inventario y activos fijos tangibles. Categoría docente: Instructor. Correo electrónico: jrhernandez@uci.cu.

Ing. Juan José Rosales Rodríguez:

Graduado de Ingeniero en Ciencias Informáticas en el año 2011. A partir de ese momento se incorpora como arquitecto en el departamento Desarrollo de Productos, del proyecto ERP[2] Cuba, en el centro CEIGE. Actualmente se desempeña como arquitecto de tecnología de los despliegues. En su trabajo de diploma desarrollo una herramienta de migración de bases de datos. Correo electrónico: jjrosales@uci.cu.

[1] Centro de Informatización de las Entidades.

[2] Enterprise Resource Planning (Planificación de recursos empresariales).

Agradecimientos

Diana Rodríguez Malagón:

…A mis padres, por existir, por darme la vida, por educarme, por hacer de mí la persona que soy hoy y por tantas cosas más que no cabrían en mil páginas solo mencionarlas.

…A todo mi familión, por apoyarme siempre, en especial a Ñaña, en ocasiones la hermana mayor de su hermana mayor.

…Al pilar fundamental de mi vida en estos 5 cursos en la universidad, mi novio, mi amigo, mi confesor, mi profesor, mi contrincante, mi compañero de tesis, Diogenito.

…A mi segunda familia, que me acogió como una hija más, Arelis, Diógenes, Arelita, Leonor, Elia y Valia.

…A mi tercera familia, la más diversa, la más alegre, mis dos grupos, el 4105 y el 3511,especialmente a Yaye, Vero, Yanet, Maikel, Doris, Lisi, Tony, Nelvis, Rada, Yadini, Karel, los Jorges, Liset, Aylén, Silvio, Luis Carlos.

…A mis amigas, mis pollitos, Yazmi, Daysi, Niuvytis, Mandita y Roci, que tanto me aconsejaron y me ayudaron.

…A los nuevos amigos, con los que he compartido estos años de alegría, tristeza, noches de sueño, a Frank, Raidel, Pedro.

…A los profes, por los conocimientos que me brindaron, por su paciencia, por su ecuanimidad, en especial a Mavi, Ana Marys, Rosalina, Hardam.

…A los tutores, por el apoyo y guía en la realización de la investigación.

…Y por último, pero no menos importante, al ejemplo de persona y dirigente que me ha guiado desde que puedo recordar, a mi eterno comandante, Fidel Castro, porque gracias a su sueño de construir una universidad mirando al progreso, es que hoy me puedo graduar como ingeniero.

Diógenes Hernández Estévez

A mis padres por apoyarme siempre, y aconsejarme en todo, por estar siempre ahí, por brindarme su amor y cariño……gracias.

A mi hermanita que la quiero mucho, es la mejor.

A mis abuelas, a mi hermana mayor, a toda la familia.

A los padres de Diana, que me acogieron como un hijo más, a Gisela y Orlando.

A mis amigos que siempre han estado en los buenos y malos ratos, en especial a Frank que es como mi tercer hermano, a Raidel que también ha estado ahí desde la primaria, a mis demás mejores amigos de la calle: el Lama, Ricardo, Pedro, Eduardo, Víctor, Rubén, Pol, Eichel, Afito, a mis amigas Débora, Laritza, Isamira, Yadini.

A mis amigos del aula, el mono, Evelio, Camilo, Alejandro, el pollo, Alain, Maikel, César, Alfredo, la china, Jany.

A mis amigos de la UCI, Luis Carlos el trompeta, Yenkiel, Karel, el kangry, Raúl, Amílcar, Revens, Jorgito.

A los del dota, que compartí mucho con ellos en los 5 años, Maoda, Overture, Sylon, Void, Buzuzima, Justice, Stark, Sylax, Executor, Bóxer, Amd, Lucky, Guate, July, z3, a los WyD, los CC, los Full, los BMS, y todos los demás rivales, a Dendy, a Maelk, a Pupey, a X, a los IG, a los Navi, a los MYM, y todos los del Batlle.

A mi tutor Juan José por aconsejarme siempre, por ser paciente, y por todo lo que me ayudó desde 4to año.

A mi tutor Javier, por ser el más exigente, por ayudarme, y brindarme conocimiento.

A todo el que aportó algo en mis 5 años aquí en la UCI.

A todos los profesores que me dieron clases y me formaron como estudiante, a Hardam, Ana Mary, Orestes, Eutimio, Yasel, Zénel.

…..Y por último y muy importante, por hacer de mí una mejor persona, a mi amiga, compañera de tesis, novia, a la que siempre estuvo a mi lado estos 5 años, a Dianita.

Dedicatoria

Diana Rodríguez Malagón:

…A mis padres, por su amor, su confianza.

…A mi hermanita del alma.

…A Dio, por ser mi vida, mis ojos y mi corazón, te amo.

Diógenes Hernández Estévez

…..a mis padres, a mis abuelas, a mi hermanita, a mi hermana mayor, a mis amigos, y muy especial a dina, te quiero mucho flaqui.

Resumen

El subsistema inventario, perteneciente al ERP cubano Cedrux, es una herramienta para realizar un buen control de los productos en los almacenes. Los componentes que lo integran manejan funcionalidades que adaptan al ERP a un entorno único, propio de las características que presentan las empresas cubanas.

En el presente trabajo de diploma se muestra el desarrollo de funcionalidades vinculadas a los procesos de entrada y las transferencias internas en almacenes, operaciones propias del subsistema antes mencionado, con el objetivo de realizar acciones o procesos que hasta el momento no se pueden realizar correctamente. Estas funcionalidades abarcan, por ejemplo, la realización de aperturas de los almacenes a través de la primera recepción de productos en los mismos y de áreas nuevas dentro de los depósitos a través de recepciones, pudiendo a su vez incluir dentro de los informes de diferencias productos que no se encuentran en el almacén y por tanto no se realizó la recepción de ninguna cantidad de ellos. Se prevé además activar y desactivar áreas, a la vez de graficar y mostrar toda la estructura de ubicación de los almacenes. Es de vital importancia además gestionar las transferencias entre áreas y entre almacenes de la entidad.

Palabras claves: almacén, apertura, área, producto, proceso, recepción, transferencia.

Índice de contenido

Índice de figuras

Índice de tablas

Introducción

El mundo empresarial se mueve constantemente con nuevos productos en diversas ramas de la ciencia y la tecnología, siempre con el objetivo fundamental de mejorar las actuales condiciones. Una de las esferas que más se ha tenido en consideración es la planificación de la economía de las empresas, pues permite determinar con antelación si estas, o alguno de sus proyectos son viables, tanto en el aspecto económico como en el financiero, a través de un análisis exhaustivo de la rentabilidad que provee cada producto y de la capacidad financiera máxima que necesitará la empresa para llevar a cabo sus actividades en cumplimiento de sus objetivos.

Un ejemplo real de como se ha tomado en cuenta este aspecto, es la creación de los sistemas para la planificación de recursos empresariales, más conocidos por sus siglas en inglés ERP. Estos consisten en conjuntos de sistemas de información que permiten la integración de ciertas operaciones de una empresa, como por ejemplo: **recursos humanos**, **contabilidad**, **planificación**, **finanzas** y **logística**. (2)

Este último contiene entre sus subsistemas al inventario, que se encarga de mantener una relación detallada de las existencias de los materiales que posee una entidad, dándole entrada y salida del almacén a los mismos, regulando el flujo de mercancías, con el fin de garantizar rentabilidad y éxito de esa organización. (2)

En busca de satisfacer la necesidad de lograr una aplicación que apoye los procesos de inventario y que permita centralizar toda la información existente en la misma, una de las organizaciones de punta que se ha sumado a la tarea es la Universidad de las Ciencias Informáticas (UCI), desarrollando un sistema integral de gestión empresarial denominado Cedrux, más conocido como el ERP cubano, el cual no es más que un paquete de soluciones integrales de gestión para entidades presupuestadas y empresariales, desarrollado siguiendo los principios de independencia tecnológica[3]. (3)

Cedrux pretende controlar la actividad económica, financiera y contable del país, y está diseñado para modelar e informatizar la mayoría de los procesos en las entidades. Este software une la información de la empresa en un lugar único, de forma tal que cualquier hecho es visible inmediatamente, posibilitando la toma de decisiones más rápida y segura, acortando los ciclos productivos. (4)

[3] Desarrollo de herramientas basadas en software libre.

Uno de los procesos más importantes para validar este software es la certificación, el cual no es más que un conjunto de acciones mediante las cuales se garantiza la calidad y/o las características del producto final, en este caso el software Cedrux, según lo establecido en normas específicas por exigencia del Ministerio de Finanzas y Precios para cualquier software contable que vaya a ser utilizado en Cuba. (5)

Este proceso de certificación es necesario para el despliegue del sistema a nivel nacional. Como parte del mismo se revisó el subsistema inventario, obteniéndose resultados satisfactorios. No obstante, en dicho escenario se detectaron aspectos que si bien no son indispensables para el control de los materiales en los almacenes, sí inciden en la calidad, rapidez y facilidad con que éste se ejecuta. De esta forma se identificaron problemas tales como:

- ✓ No puede realizarse la apertura de un almacén a través de la primera recepción de productos en los mismos.
- ✓ No puede realizarse la apertura de nuevas áreas dentro de los depósitos a través de recepciones.
- ✓ El sistema no brinda la opción de, según su necesidad, activar o desactivar un área determinada.
- ✓ No existe forma de saber gráficamente como está estructurado un almacén.
- ✓ Las transferencias entre áreas o entre los almacenes no se realizan debidamente, lo cual demora y/o dificulta el proceso.
- ✓ No existe forma de incluir dentro de los informes de diferencia productos que no se encuentran en el almacén y que debían llegar en alguna entrega.

Dada la situación problemática definida anteriormente se plantea el siguiente **problema a resolver**: la gestión de los procesos de entrada y transferencias internas en el subsistema inventario de Cedrux dificulta la calidad con que se realiza el control de los materiales de los almacenes en las entidades cubanas.

Para ello el **objeto de estudio** de la investigación se centra en los procesos de entrada y transferencias internas vinculados al control de los materiales en los almacenes, y donde el **campo de acción** definido para la misma será el conjunto de procesos de entrada y transferencias internas del subsistema inventario de Cedrux. Para guiar la investigación se plantea como **objetivo general** desarrollar funcionalidades de los procesos de entrada y transferencias internas del subsistema Inventario de Cedrux para mejorar el control de

los materiales de los almacenes en las entidades cubanas, y en aras de alcanzarlo se plantean los siguientes **objetivos específicos**:

- Establecer el marco teórico referencial sobre los procesos de entrada y de transferencias internas de productos en los almacenes.
- Gestionar los procesos de entrada y de transferencias internas de productos en los almacenes.
- Validar la solución propuesta.

Para lograr un mejor desarrollo de este trabajo se ha planteado la siguiente **idea a defender**: si se desarrollan las funcionalidades para los procesos de entrada y transferencias internas en Cedrux se logrará mejorar la calidad con que se realiza el control de los materiales en los almacenes de las entidades cubanas.

Los siguientes métodos teóricos sustentan el trabajo de investigación:

- Modelación. Se crean abstracciones que explican la realidad, por ejemplo, el modelo conceptual, el mapa de procesos, modelo de datos, los diagramas de clases de los componentes de transferencias, diagramas de interacción, el modelo de diseño, el diagrama de componentes, etc.
- Histórico-Lógico. Su empleo permitió el crecimiento cognoscitivo acerca de sistemas ERP y de los procesos que se realizan en el subsistema inventario, a través de la investigación realizada, expresada en los diversos materiales bibliográficos.

Capítulo 1: Fundamentación Teórica

Introducción

El presente capítulo está destinado a sentar las bases teóricas para la investigación que continúa. Para ello se citan una serie de conceptos básicos relacionados con los procesos de entrada y transferencias internas del subsistema Inventario de Cedrux, para guiar de forma organizada la lectura del documento. También dentro del propio capítulo, se hace un resumen de las características fundamentales de las herramientas y tecnologías, utilizadas según el modelo de desarrollo propuesto por CEIGE, de acuerdo a las necesidades de los proyectos del mismo, por lo que se realizará una descripción detallada de las mismas con el objetivo de conocer sus múltiples usos y ventajas.

1.1. Conceptos generales

Para brindar una breve panorámica de la investigación, a continuación se relacionan conceptos fundamentales a tener en consideración para el mejor entendimiento de la misma. Dichos conceptos fueron extraídos según términos del manual de usuario del módulo inventario en su versión 1.0.0.

Inventario: En términos generales el inventario es una relación detallada de las existencias de materiales. Se encarga de inspeccionar las existencias de productos, dándole entrada y salida del almacén a los mismos, regulando el flujo de mercancía en este, con el fin de hacer más rentable su posesión y garantizar en cierto grado el éxito de la organización. (6)

Almacén: El almacén es el lugar donde se llevan a cabo los movimientos de productos: entradas, salidas e inventario de los mismos. (6)

Apertura: Este es el proceso enmarcado en el momento en que es habilitado un nuevo almacén y comienza a recibir productos. También es utilizado al implantarse el sistema en un almacén en funcionamiento. Tiene como objetivo fijar el inventario inicial de los productos existentes en cada área del almacén y se realiza una sola vez para un área dentro de un depósito. (6)

Cantidad: Es un atributo del producto cuando se registra este en un documento. Puede ser de entrada al almacén, salida del almacén, contada en un inventario o bien ajustada mediante un ajuste de inventario. (6)

Recepción de productos: La recepción de productos comienza con la entrega por parte del proveedor o el transporte de aprovisionamiento de la empresa, el cual debe entregar el producto y la documentación que ampare dicha adquisición. (6)

Formato ubicación: Formato específico que debe de configurarse por niveles y cada nivel tendrá un código asignado por el usuario. (6)

Ubicación: Es un lugar dentro del área comprendida del almacén que puede ser sección-estante-anaquel-casilla o como lo tenga estructurado el almacén en cuestión. (6)

Tipos de documentos de Inventario

Informe de diferencias: Este documento se realiza con la intención de llevar el control de todas las diferencias que se generan durante las recepciones de mercancía en el almacén. (6)

Informe de recepción: Este es un documento que por su naturaleza tiene la responsabilidad de formalizar la recepción en el almacén de los productos adquiridos de otras entidades. (6)

Transferencias entre almacenes: Ampara las transferencias entre almacenes a través de tres procesos fundamentales, despacho, documento de salida y recepción, los cuales generan diferentes documentos, de forma tal que una vez comience el primer proceso este generará una autorización de entrega o plan de distribución, que propiciará pasar al segundo proceso, donde se generará a partir del documento realizado, un documento de salida, que a su vez se le hará la recepción por el almacén receptor, culminando el tercer proceso y la transferencia entre ambos almacenes. (6)

Transferencias entre áreas: Es el documento que ampara las transferencias entre áreas de un mismo depósito. (6)

1.2. Estudio de sistemas anteriores

Como parte del estudio preliminar que se realizó sobre los procesos de inventario, enfocado principalmente en el funcionamiento de las transferencias entre áreas y almacenes, así como las entradas de productos a estos últimos, se analizó la manera en que se llevaban a cabo los mismos en otros sistemas ERP, para comprender qué particularidades de estos aportarían cierto grado de optimización a los ya mencionados procesos para ponerlos en práctica en las entidades cubanas. A continuación se relacionan algunas características fundamentales de los sistemas estudiados.

1.2.1. Sistemas internacionales

A nivel mundial existen varios sistemas informáticos que apoyan los procesos de inventario en los cuales se centra la investigación, como la recepción, la transferencia de productos entre áreas y almacenes, así como la forma de estructurar la ubicación de los productos dentro de estos últimos. Ello les proporciona la seguridad y calidad de un control eficiente.

Dan fe de ello sistemas como SAP[4], el cual cuenta con un módulo logístico llamado MM (Material Management), que ofrece una solución a todas las fases de gestión de materiales de una empresa, incluyendo la entrada de mercancías. (7)

Este sistema realiza la recepción de productos contra un pedido y mantiene actualizado diariamente las entradas de los mismos, y a través de la aptitud de los materiales realiza diferentes transferencias de los mismos a lugares determinados por la organización, siendo estas diariamente indicadas en los documentos pertinentes y realizando además un listado diario que indica los traspasos de almacén a fábrica. A su vez se mantiene un control riguroso del lugar en que se encuentra ubicado cada producto. (7)

Otro sistema que maneja igualmente estos procesos es el OpenBravo, contando con un módulo de gestión de almacenes que recopila diferentes procesos tales como movimientos de productos entre almacenes, regulaciones de inventario, entradas de material, etc. Entre las funcionalidades de este módulo es importante destacar que permite definir la estructura de los almacenes de la empresa y las ubicaciones físicas donde se almacenará el material dentro de los mismos y permite realizar un movimiento de productos entre almacenes, actualizando las cantidades almacenadas de cada producto en ellos. Además provee una integración del proceso de la recepción en el almacén para reducir errores y tiempo invertido, integrando datos de ventas e inventario a tiempo real desde las tiendas para optimizar las compras y los niveles de almacenamiento. (8)

1.2.2. Sistemas nacionales

En el caso de Cuba, uno de los ejemplos de sistemas ERP que apoyan los procesos de inventario es el Versat Sarasola, abreviadamente Versat. Éste es un paquete de gestión económico-financiera, que permite registrar, controlar y analizar los hechos económicos,

[4] Sistemas, Aplicaciones y Procesamiento de datos.

que acontecen en una entidad, a partir de la información contenida en los documentos primarios. (9)

Versat cuenta con un módulo dedicado al inventario de los productos en los almacenes, que se encarga de registrar todos los documentos primarios que tradicionalmente generan las entradas y salidas en los almacenes; además tiene incorporada una gran cantidad de opciones que permiten ejecutar los conteos físicos y el tratamiento de los diferentes tipos de inventarios, que para el subsistema se llaman categorías. (10)

1.2.3. Análisis de las soluciones existentes.

En el breve estudio realizado a estos sistemas se apreció entre otras limitaciones el alto costo de implementación en el caso de OpenBravo a pesar de ser de código abierto. A su vez impide el uso del ERP SAP su categoría de software privativo. A ello se le suma que estos sistemas no contemplan similares aspectos a tener en cuenta en el control del inventario debido a la gran brecha que surge entre las diferentes economías.

En el caso del sistema cubano Versat Sarasola además de ser una aplicación de escritorio, los costos de implementación son altos, debido a que el personal no está adecuadamente preparado, presentándose como un factor de peso en contra la resistencia al cambio que presentan las empresas de forma general. Por todos los factores antes mencionados, se decide que los sistemas estudiados no brindan la posibilidad de ser utilizados como referencia a la hora de realizar los procesos y requisitos pertinentes en la presente investigación.

1.3. Modelo de desarrollo de software

El Modelo de Desarrollo, de acuerdo con los requisitos necesarios por el proyecto, es el descrito por el centro CEIGE, el cual plantea que para el ciclo de vida de cada producto y posterior especificación se tienen en cuenta las fases y actividades por áreas de procesos, que plantea el nivel dos de CMMI[5] establecido en la universidad. Las fases propuestas por CMMI abarcan las acciones que se realizan en las distintas líneas de desarrollo para la elaboración del servicio o producto final, como son Estudio Preliminar, Modelado del Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas Internas y Pruebas de Liberación. (11)

[5] Integración de modelos de madurez de capacidades

En cada una de estas fases se realizan diferentes actividades, de las que se obtienen como resultado diversos artefactos. Por ejemplo, en la primera fase, Estudio Preliminar, las actividades que en esta se desarrollan tienen que ver de cerca con la planeación del proyecto a un alto nivel, así como el estudio enfocado en la organización cliente, que permite obtener información fundamental acerca del alcance del proyecto y realizar estimaciones de tiempo, esfuerzo y costo.

Por su parte, la fase Modelado del Negocio, está destinada a comprender los procesos de negocio de la organización. Se comprende cómo funciona el negocio que se desea automatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Una actividad fundamental es, precisamente, modelar el negocio. Como resultado se obtiene la descripción de requisitos del cliente a través de artefactos como el modelo conceptual, el modelo de procesos de negocio, el mapa de procesos de negocio, etc.

La tercera fase, Requisitos, también desarrolla un conjunto de actividades, entre las que se destacan desarrollar y administrar requisitos. A partir de estos se obtienen artefactos tales como los documentos de especificación de requisitos de software.

La siguiente fase, Análisis y diseño, se encarga del modelado del sistema para que soporte todos los requisitos. Esto contribuye a una arquitectura sólida y estable. (20)

Una de las principales actividades que define es el ddiseño de los componentes. Como artefactos genera diagramas de clases del diseño de interacción (secuencia o colaboración).

Otras actividades importantes son el análisis y diseño de la base de datos, de la cual se obtiene el modelo de datos y el script de la misma.

A partir de los resultados del análisis y diseño se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ejecutables y similares, o sea, la fase de Implementación. Entre sus principales actividades se encuentran:

- Implementar funcionalidades en términos de componentes: como resultado se obtienen los ficheros de código y algoritmos de implementación.
- Elaborar el manual de usuario.

La penúltima fase, Pruebas Internas, desarrolla las pruebas del grupo de calidad del centro verificando el resultado de la implementación.

La fase final, Pruebas de Liberación, está destinada a la aplicación de pruebas a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

1.4. Arquitectura

La arquitectura de software de un sistema es la estructura del mismo, lo cual abarca componentes de software, las propiedades visibles externamente de esos componentes, y las relaciones entre ellas. De esta manera, la arquitectura de software permite representar de forma concreta la estructura y funcionamiento interno de un sistema. (12)

1.5. Herramientas y tecnologías utilizadas

Las herramientas que se tienen en consideración para el desarrollo práctico de las funcionalidades requeridas son las definidas por el marco de trabajo del centro CEIGE. A continuación se muestra una síntesis de cada una:

1.5.1. Herramienta de modelado

Visual Paradigm para UML[6] 8.3.8

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una rápida construcción de aplicaciones, mejores y a un menor coste. Permite diseñar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (7)

1.5.2. Servidor web

Apache 2.2

Este servidor web de software tiene el objetivo de servir o suministrar páginas web a los clientes web o navegadores que las solicitan. Proporciona como principales características:

- Soporte multiprotocolo.

Cuenta con algunas partes de la infraestructura en lugares que sirven de soporte para servir múltiples protocolos.

- Nueva API[7].

[6] Unified Model Language (Lenguaje unificado de modelado).

La API para módulos ha sido cambiada significativamente para esta versión. Muchos de los módulos de ordenamiento y prioridad de problemas desde la versión 1.3 han sido resueltos y los módulos de ordenamiento están realizados para permitir más flexibilidad. (8)

1.5.3. Gestor de base de datos

PostgreSQL 8.3.8

Es un sistema de gestión de base de datos relacional orientada a objetos de software libre. Tiene una arquitectura probada, por lo que ha ganado una sólida reputación para la fiabilidad, integridad y exactitud de los datos. Funciona en todos los principales sistemas operativos. Soporta el almacenamiento de grandes objetos binarios, incluyendo imágenes, sonidos o vídeo. Tiene interfaces de programación en los lenguajes C/C++, Java, Net, Perl, Python, Ruby, etc. Cuenta con versiones para una amplia gama de sistemas operativos, entre ellos: Linux, Windows, Solaris y otros más. (9)

1.5.4. Cliente o explorador web

Mozilla Firefox 3.6 o superior

Firefox 3.6 está desarrollado sobre la plataforma de dibujo web Gecko 1.9.2 de Mozilla, y contiene mejoras para desarrolladores web y para usuarios. Entre sus características fundamentales se encuentran:

- Rendimiento mejorado de JavaScript, respuesta general del navegador y tiempo de inicio.
- Implementación de nuevos atributos CSS[8] como degradados, tamaño de fondos, y eventos de puntero.
- Implementación de nuevas especificaciones DOM[9] y HTML5[10], que permitirán páginas web más interactivas.

---

[7] Application Programming Interface (Interfaz de la programación de la aplicación).

[8] Cascade Style Sheet (Hojas de estilo en cascada)

[9] Document Object Model(Modelo de documento de objetos)

Se utiliza esta versión ya que en un principio el sistema ERP Cedrux estaba pensado para soportar dicha versión. (10)

1.5.5. Herramienta de desarrollo colaborativo

RapidSVN 1.6.17

RapidSVN es un cliente gráfico multiplataforma que se distribuye bajo la Licencia Pública General de GNU. Es bastante simple, pues proporciona una interfaz fácil de usar para las características del sistema de control de versiones **Subversion**; eficiente, debido a su simplicidad para los principiantes, pero lo suficientemente flexible como para aumentar la productividad para los usuarios con experiencia. Es portable, rápido, multilingüe y posee soporte completo para el estándar de codificación denominado Unicode. (11)

1.5.6. IDE[11] de desarrollo

Netbeans 7.2

Es un entorno de desarrollo, una herramienta práctica para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero es adaptable para cualquier otro lenguaje de programación. Es un producto libre y gratuito sin restricciones de uso. (12)

1.5.7. Marco de trabajo

Doctrine 0.0.7

Doctrine es un sistema ORM[12], o sea, implementa la técnica de programación que nos permite convertir datos entre el sistema de tipos utilizados en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, de forma tal que las tablas de la base de datos pasan a ser clases y los registros, objetos que podemos manejar con mayor facilidad, para PHP 5.2 o superior. Además de las ventajas que conlleva un ORM, uno de sus puntos fuertes es su lenguaje de consulta DQL[13],

---

[10] HyperText Markup Language (Lenguaje de marcado de hipertexto)

[11] Integrate Development Environment (Entorno de desarrollo integrado).

[12] Object Relational Mapping (Mapeo de Objeto-Relacional).

[13] Doctrine Query Language (Lenguaje de consulta de Doctrine).

inspirado en el HQL[14] de Hibernate, ya que provee potentes capacidades de consulta sobre el modelo de objetos.(13)

ExtJS 2.2

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX[15], DHTML[16] y DOM[17]. Incluye un alto rendimiento, interfaces de usuario personalizables, bien diseñado y extensible modelo de componentes, una interfaz intuitiva y fácil de utilizar con licencias de código abierto y comercial. Brinda soporte para construir interfaces gráficas complejas y dinámicas y comunicar datos de forma asíncrona con el servidor. (14)

Para esta versión específica contiene características primordiales como las siguientes:

- Soporte completo para la versión 3.0 de Firefox.
- Adición del componente Ext.History.
- Extensivos adelantos en la documentación.
- Varios ficheros locales de actualización.

Zend_Framework 1.4

Zend_Framework (ZF) es un marco de trabajo de código abierto para el desarrollo de aplicaciones web y servicios con PHP 5, brindando soluciones para construir sitios web modernos, robustos y seguros. Está implementado usando 100% de código orientado a objetos. El componente de estructura es algo único; cada componente es diseñado con pocas dependencias de otros componentes. Esta arquitectura poco acoplada permite a los desarrolladores usar los componentes individualmente. (15)

1.5.8. Lenguaje del lado del servidor

PHP 5.3.10

[14] Hibernate Query Language (Lenguaje de consulta de Hibernate).

[15] Asynchronous JavaScript and XML (JavaScript asíncrono y XML).

[16] Dynamic Hypertext Markup Language (Lenguaje de Marcado de Hipertexto Dinámico).

[17] Document Object Model (Modelo de Objetos del Documento).

PHP 5 es una versión que presume un mejor soporte para la Programación Orientada a Objetos en comparación con versiones anteriores. Presenta mejoras de rendimiento y mejor soporte para MySQL con extensión completamente reescrita, así como para XML[18]. Presenta un soporte nativo para SQLite e integrado para SOAP[19]. Es muy eficiente con el manejo de excepciones. (16)

1.5.9. Lenguaje del lado del cliente

JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con Java Script se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (17)

1.5.1. Cliente-Servidor

La arquitectura cliente/servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes. En este modelo las aplicaciones se dividen de forma que el servidor contiene la parte que debe ser compartida por varios usuarios, y en el cliente permanece sólo lo particular de cada usuario. (20)

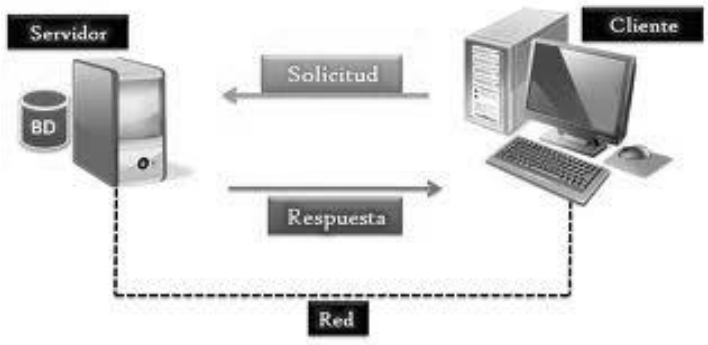

Figura 1: Arquitectura Cliente-Servidor.

[18] eXtensible Markup Language (Lenguaje de marcado extensible)

[19] Simple Object Access Protocol (Protocolo de acceso a objetos simples)

Entre las principales características de esta arquitectura se pueden destacar las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

1.5.2. Modelo-Vista-Controlador

Es un modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. (20)

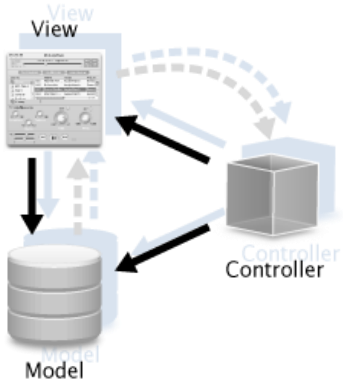

Figura 2: Patrón Modelo-Vista-Controlador.

- **Modelo**: Es la representación específica de la información con la cual el sistema trabaja. Es responsable de la lógica de negocio de una aplicación. Se encapsula el acceso a los almacenes de datos y proporciona una biblioteca de clases reutilizables.
- **Vista**: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario. Es quien controla la apariencia de los datos y proporcionan funcionalidades para recoger los de los usuarios.
- **Controlador**: Responde a eventos, usualmente acciones del usuario, e invoca peticiones al modelo y, probablemente, a la vista. Es la encargada de unir la capa de las vistas con la del modelo, así como de recibir la información recibida por la

vista y manejar el flujo de ejecución de la aplicación, es la encargada de llamar a las funcionalidades del modelo y retornar estos datos a una vista. (20)

Conclusiones parciales

En este capítulo, con el análisis referente a los sistemas ERP, se llegó a la conclusión de que resulta necesario desarrollar las funcionalidades necesarias siguiendo la línea del sistema Cedrux bajo la filosofía de software libre, para que así pueda ser utilizado por el estado cubano ya que el pago de licencias suele ser engorroso en temas económicos para Cuba. Con el estudio del modelo de desarrollo de software se podrán seguir detenidamente cada una de las fases establecidas por CEIGE para obtener un producto que responda a las necesidades del cliente y que tenga la calidad requerida, utilizando las herramientas y tecnologías establecidas desde la primera fase de desarrollo del sistema Cedrux.

Capítulo 2: Análisis y Diseño

Introducción

En el presente capítulo se describen los flujos de los procesos del negocio relacionados con el control de los materiales en los almacenes. Se enumeran además los requisitos funcionales, que abordan los elementos que debe tener el sistema propuesto, lo que permite tener una concepción del mismo. Se identifican algunos artefactos fundamentales que describen los procesos de negocio, como el mapa de procesos, el diagrama de componentes, la relación y descripción de los propios procesos principales, así como el diseño del sistema y su validación, basado en el uso de patrones en la solución y el diseño del modelo de datos.

2.1. Mapa de procesos del negocio

El mapa de procesos del negocio tiene como propósito brindar una breve descripción de cada uno de los procesos para la mejor comprensión de los mismos. (21)

En este caso específico, los procesos a modelar son los siguientes:

- **Gestionar transferencias entre áreas**
- **Gestionar transferencias entre almacenes**

A continuación se presenta el mapa de procesos de forma general y por separado a partir de cada proceso:

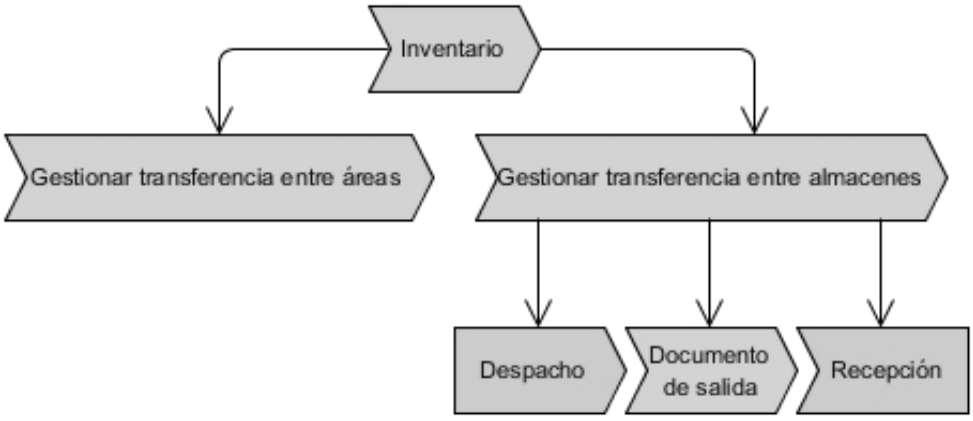

Figura 3: Mapa de procesos general.

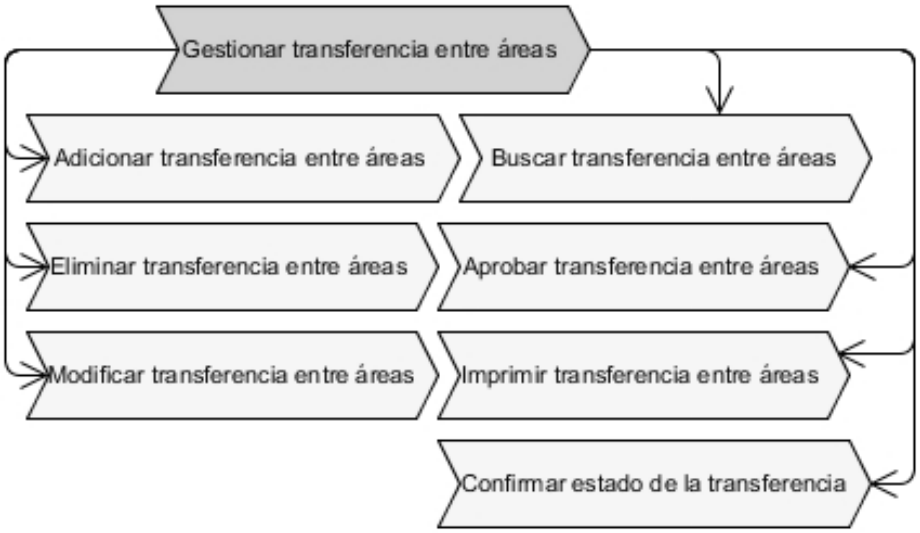

Figura 4: Mapa de procesos. Gestionar transferencia entre áreas.

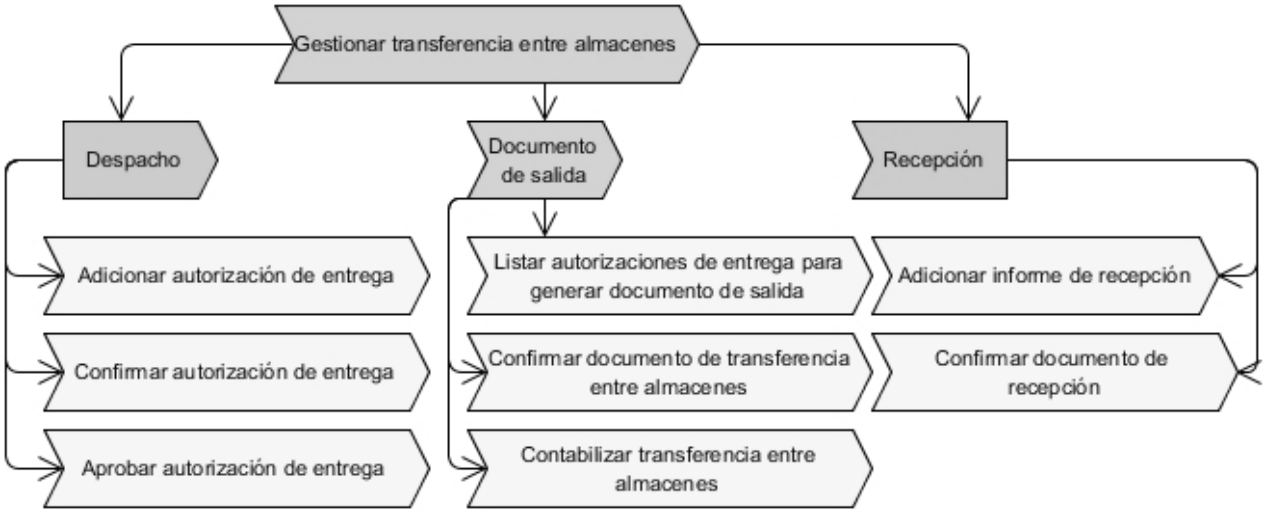

Figura 5: Mapa de procesos. Gestionar transferencia entre almacenes.

2.1.1. Descripción del proceso de negocio: Gestionar transferencia entre áreas

Es el proceso mediante el cual se realizan los movimientos de recursos o productos de un área determinada hacia otra dentro del mismo almacén de una entidad. (5)

Tabla 1: Descripción del proceso: Gestionar transferencia entre áreas.

Objetivos	**Realizar transferencia de productos entre áreas dentro de un almacén.**
Entradas	Información (Datos del producto a transferir y del

	área de origen y la de destino).
Precondiciones	Se debe verificar que exista el área a la que se pretende transferir y que esté activada.
Post-condiciones	Se debe actualizar la existencia del producto en el área a la que se transfirió y el cambio en el área de donde vino.
Salidas	Documento de la transferencia.

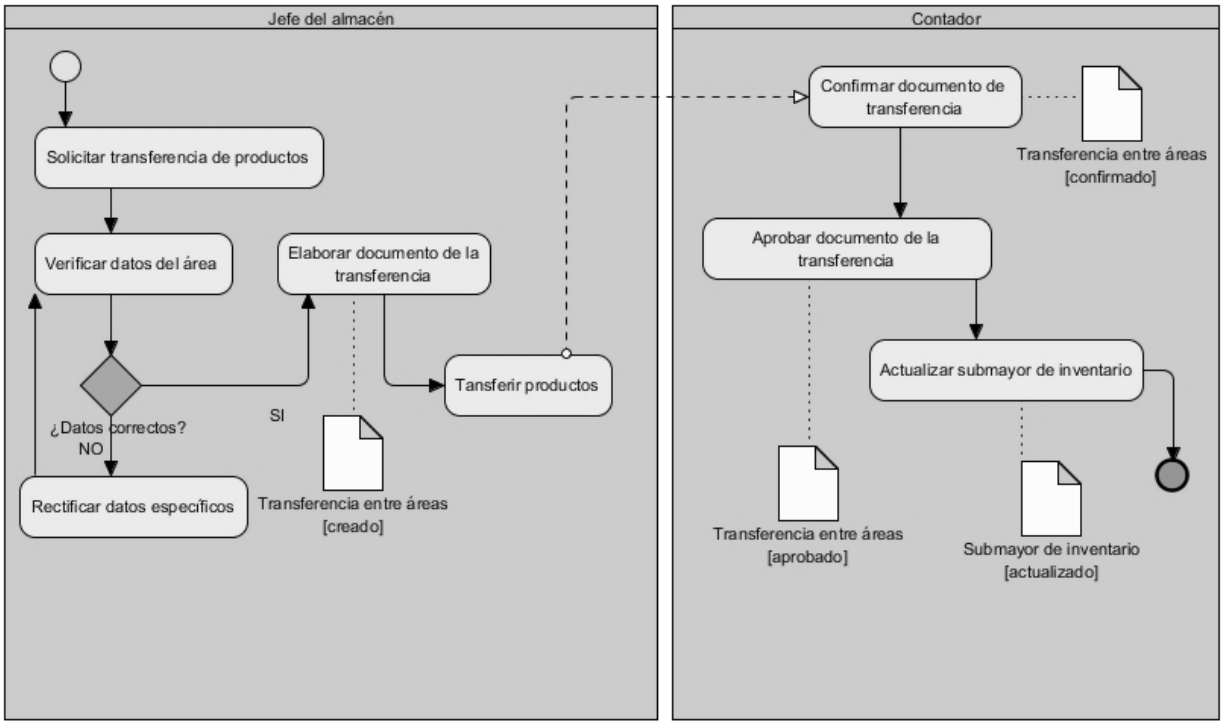

Figura 6: Diagrama del proceso Gestionar transferencia entre áreas.

2.1.2. Descripción del proceso de negocio: Gestionar transferencia entre almacenes

Es el proceso mediante el cual se realizan los movimientos de productos de un almacén a otro a través de tres procesos, comenzando por el despacho, del cual como resultado se obtiene una autorización de entrega, a partir de la cual se genera un documento de salida en el segundo proceso a efectuar, seguido de la confección de un informe de recepción, realizado por parte del almacén al cual se transfieren los productos, siendo el tercer y último proceso. (5)

Tabla 1: Descripción del proceso de negocio: Gestionar transferencia entre almacenes (Despacho y Documento de salida).

Objetivos	**Gestionar el documento de salida de los productos del almacén a través de la autorización de entrega.**
Entradas	Datos de la autorización de entrega o plan de distribución. Datos del producto.
Precondiciones	Se debe verificar que exista el almacén.
Post-condiciones	Se debe actualizar la existencia del producto en el almacén al que se transfirió.
Salidas	Documento de salida.

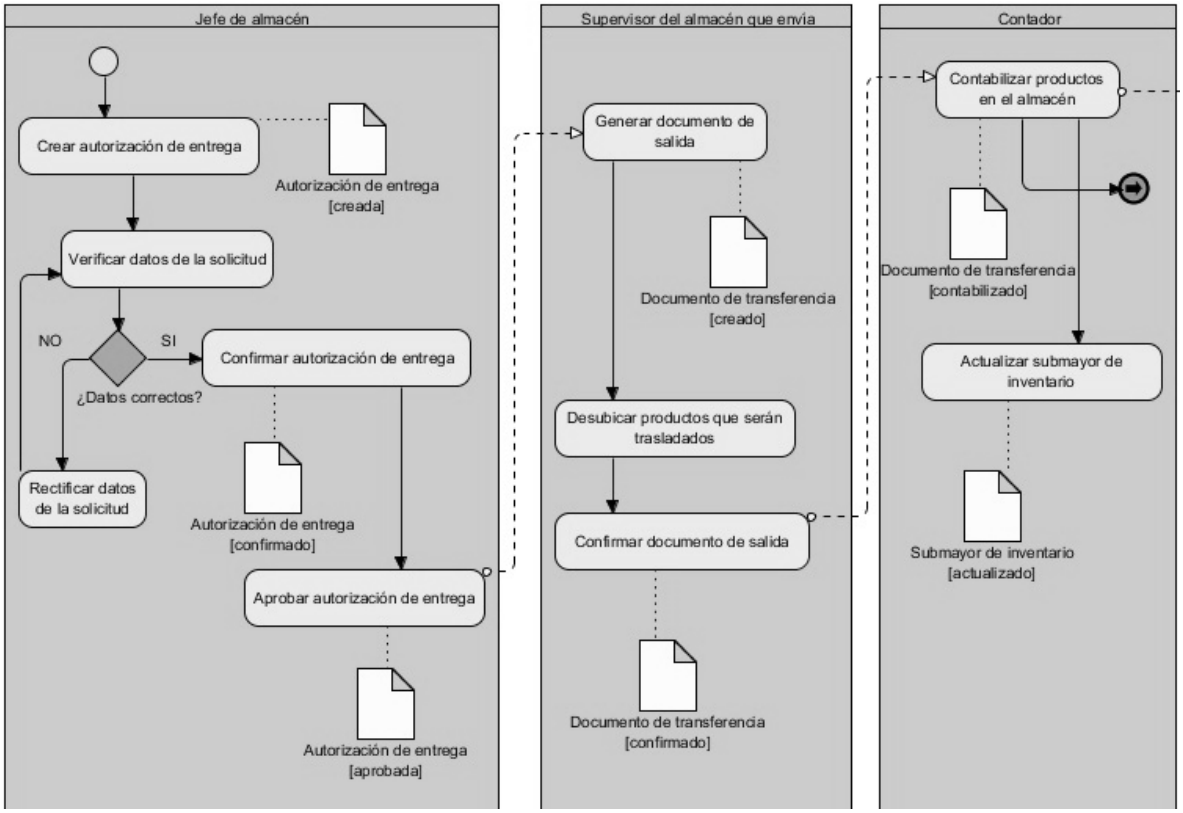

Figura 7: Diagrama del proceso Gestionar transferencia entre almacenes (Parte 1).

Tabla 2: Descripción del proceso de negocio: Gestionar transferencia entre almacenes (Recepción).

Objetivos	**Gestionar informe de recepción para almacenar los nuevos productos.**
Entradas	Datos del producto.
Precondiciones	Se debe verificar la llegada de productos al almacén, teniendo en cuenta que se corresponda la cantidad de estos con la cantidad documentada en el documento de salida.
Post-condiciones	Se debe actualizar la existencia del producto en el almacén al que se transfirió.
Salidas	Informe de recepción.

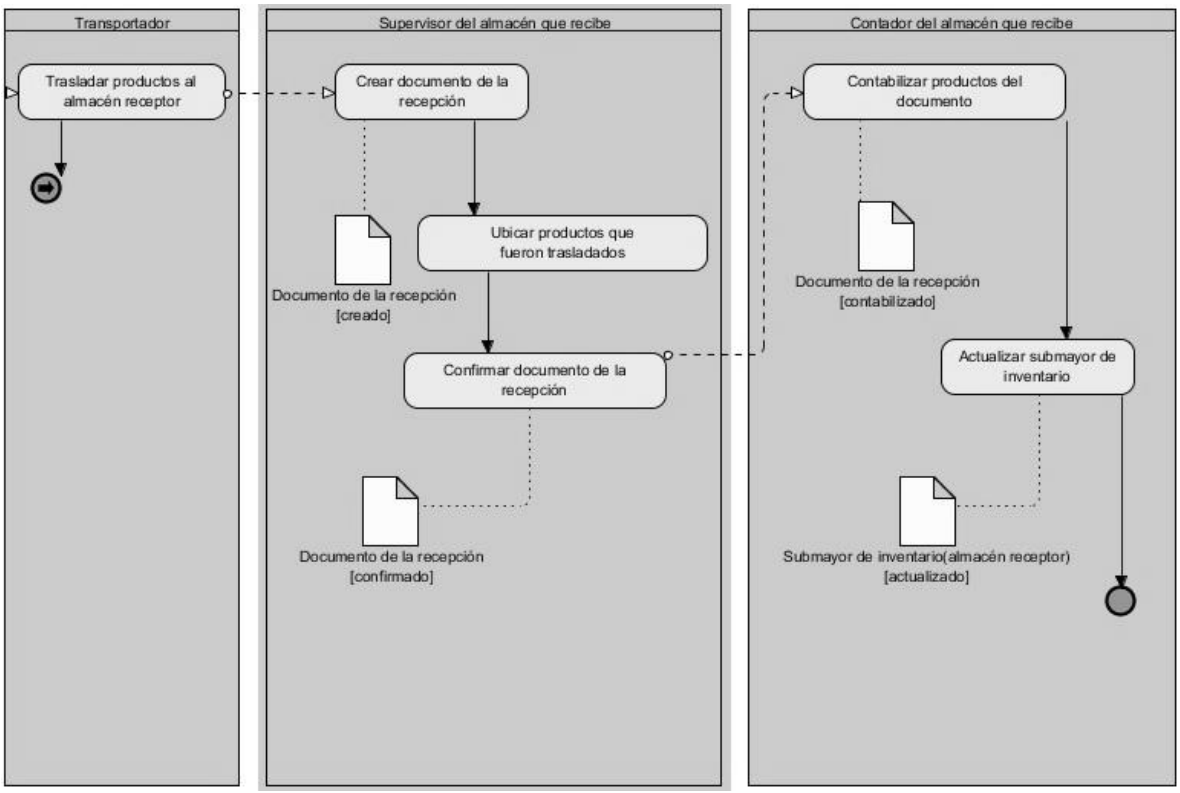

Figura 8: Diagrama del proceso Gestionar transferencia entre almacenes (Parte 2).

2.2. Modelo Conceptual

El modelo conceptual ilustra conceptos (objetos) en el dominio del problema. No es más que la representación de cosas del mundo y está representado en este caso en un diagrama de estructura estático (notación UML). (20)

En este caso específico se identifican los principales elementos físicos o lógicos del negocio para ayudar a entender el problema, representados a través de clases como se muestra a continuación:

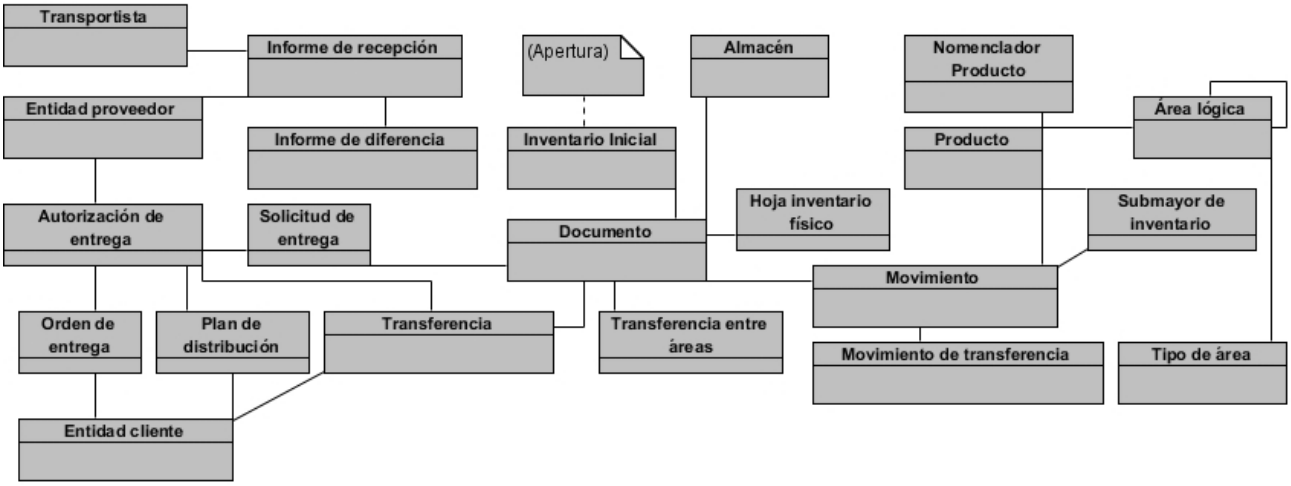

Figura 9: Modelo Conceptual.

2.3. Requisitos del sistema

2.3.1. Requisitos funcionales

Los requisitos funcionales definen el comportamiento interno del software, en éste caso específico, son funcionalidades que deben ser agregadas para proveer mejoras al funcionamiento del módulo inventario. (20)

El requisito siguiente es incorporado a raíz de implementar un nuevo proceso: gestionar transferencia entre áreas. La especificación de los requisitos que incluye el antes mencionado se puede encontrar en los anexos.

RF1. Gestionar transferencias entre áreas.

RF1.1 Adicionar transferencias entre áreas.

El sistema debe permitir adicionar una transferencia entre áreas de ser necesario, introduciendo los datos necesarios para que sea válida la misma. Ver especificación (anexo 1) al final del documento.

RF1.2 Modificar transferencias entre áreas.

El sistema debe permitir modificar una transferencia entre áreas de ser necesario, introduciendo los datos necesarios para que sea válida la misma. Ver especificación (anexo 2) al final del documento.

RF1.3 Eliminar transferencias entre áreas.

El sistema debe permitir eliminar una transferencia entre áreas, pidiendo siempre confirmación de la orden antes de ejecutar la acción. Ver especificación (anexo 3) al final del documento.

RF1.4 Buscar transferencias entre áreas.

El sistema debe permitir buscar una transferencia entre áreas, en correspondencia con los datos entrados para la misma (número, estado, año). Ver especificación (anexo 4) al final del documento.

RF1.5 Realizar búsqueda avanzada.

El sistema debe permitir buscar una transferencia entre áreas, en correspondencia con los datos entrados para la misma (número del documento, año, estado del documento, área origen, área destino, depósito destino, creado por, aprobado por, rango de fecha (desde - hasta)). Ver especificación (anexo 5) al final del documento.

RF1.6 Imprimir modelo de transferencia entre áreas.

El sistema debe permitir la opción de imprimir un modelo de la transferencia entre áreas. Ver especificación (anexo 6) al final del documento.

RF1.7 Imprimir transferencias entre áreas.

El sistema debe permitir la opción de imprimir todas las transferencias entre áreas realizadas en un almacén determinado. Ver especificación (anexo 7) al final del documento.

RF1.8 Confirmar documento de la transferencia entre áreas.

El sistema debe permitir confirmar el documento una vez que se añaden a este los productos de la transferencia y datos necesarios de la misma. Ver especificación (anexo 8) al final del documento.

RF1.9 Aprobar documento de la transferencia entre áreas.

El sistema debe permitir aprobar el documento una vez que se ha confirmado el mismo. Ver especificación (anexo 9) al final del documento.

RF1.10 Listar transferencias entre áreas.

El sistema debe permitir la opción de listar todas las transferencias de un almacén determinado. Ver especificación (anexo 10) al final del documento.

RF1.11 Cancelar estado del documento de la transferencia.

El sistema debe permitir la opción de cancelar el documento en caso de errores. Ver especificación (anexo11) al final del documento.

El requisito siguiente es incorporado a raíz de implementar un nuevo proceso, que es gestionar transferencia entre almacenes, que se realiza a partir de tres procesos ya implementados, que son: despacho, documento de salida y recepción.

RF2. Gestionar transferencias entre almacenes.

Despacho:

RF2.1 Adicionar autorización de entrega.

El sistema debe permitir la opción de adicionar una autorización de entrega cuyo tipo de transferencia es entre almacenes. Ver especificación (anexo 12) al final del documento.

RF2.2 Confirmar autorización de entrega.

El sistema debe permitir la opción de confirmar una autorización de entrega cuyo tipo de transferencia es entre almacenes. Ver especificación correspondiente en la documentación de la investigación.

RF2.3 Aprobar autorización de entrega.

El sistema debe permitir la opción de aprobar una autorización de entrega cuyo tipo de transferencia es entre almacenes. Ver especificación correspondiente en la documentación de la investigación.

Documento de salida:

RF2.4 Listar autorizaciones de entrega para generar documento de salida.

El sistema debe permitir mostrar una lista de autorizaciones de entrega cuyo tipo de transferencia es entre almacenes para generar el documento. Ver especificación (anexo 13) al final del documento.

RF2.5 Confirmar documento de transferencia entre almacenes.

El sistema debe permitir confirmar el documento de salida cuando se encuentre en estado "Elaboración" y contenga cantidades de los productos, así como debe tener todos los demás datos requeridos. Ver especificación correspondiente en la documentación de la investigación.

RF2.Contabilizar transferencia entre almacenes.

El sistema debe permitir contabilizar los productos del documento de la transferencia cuando el documento se encuentre en estado "Preparado". Ver especificación correspondiente en la documentación de la investigación.

Recepción:

RF2.6 Adicionar informe de recepción.

El sistema debe permitir agregar un informe de recepción en el almacén receptor, al que llegan los productos de la transferencia entre almacenes, tal como se genera un documento de salida para tramitar la transferencia y validando la igualdad de los datos de los productos en ambos. Ver especificación correspondiente en la documentación de la investigación.

RF2.7 Confirmar documento de recepción.

El sistema debe permitir confirmar el documento de recepción a partir de que se encuentre en estado de "Elaboración" y tenga productos asociados y cantidades asociadas al mismo. Ver especificación correspondiente en la documentación de la investigación.

Los siguientes requisitos funcionales están implicados en procesos anteriormente implementados, y están dirigidos a mejorar ciertos aspectos que dificultan el manejo de los productos.

Dentro de la Recepción:

RF3. Realizar apertura de almacenes a través de la primera recepción de productos.

El sistema debe permitir, a partir del proceso de la recepción, que se realice la apertura de un almacén que no ha comenzado a recibir productos. Ver especificación correspondiente en la documentación de la investigación. Ver especificación (anexo 17) al final del documento.

RF4. Realizar apertura de áreas nuevas dentro de los almacenes a través de recepciones.

El sistema debe permitir, a partir del proceso de la recepción, la apertura de nuevas áreas dentro de los almacenes. Ver especificación correspondiente en la documentación de la investigación. Ver especificación (anexo 18) al final del documento.

RF5. Adicionar al informe de diferencias productos que no existen en el almacén.

El sistema debe permitir a partir del informe de reclamación o diferencias, incluir productos que no llegaron en la entrega pactada y no se hizo recepción de ninguna cantidad de ellos. Ver especificación correspondiente en la documentación de la investigación.

Dentro de Estructura de ubicación:

RF6. Activar áreas.

El sistema debe permitir la activación de áreas que no se encuentren habilitados por diversos motivos. Ver especificación correspondiente en la documentación de la investigación. Ver especificación (anexo 15) al final del documento.

RF7. Desactivar áreas.

El sistema debe permitir la desactivación de áreas que no sean necesarias o presenten algún problema dentro de los almacenes, teniendo en cuenta que éstas no tengan productos ubicados y que estén activadas. Ver especificación correspondiente en la documentación de la investigación. Ver especificación (anexo 16) al final del documento.

RF8. Generar gráficos de estructura de ubicación de los almacenes.

El sistema debe permitir la opción de poder generar un gráfico que muestre la estructura de ubicación escogida. Ver especificación (anexo 14) al final del documento.

2.3.1.1. Validación de requisitos funcionales

La validación de requisitos permite definir que los requerimientos especificados realmente detallan el sistema que el cliente necesita. Verifica que las especificaciones de requisitos no presentan omisiones, conflictos y ambigüedades, además de que sean correctas las interpretaciones por parte del equipo de desarrollo de software. Entre las técnicas utilizadas para validar los requisitos identificados se utilizaron las siguientes. (20)

- **Revisiones**: Esta técnica se utiliza para corregir cualquier error existente en la documentación o modelado de los requisitos, con el objetivo de encontrar conflictos en el producto, y poder trazar alternativas de solución.

En el capítulo 3, en el epígrafe dedicado al diseño de casos de prueba (3.2.3.3), se relacionan los resultados de las revisiones efectuadas.

- **Prototipos**: Es utilizado como modelo a escala de la solución final, para brindarle al cliente una visión más clara de cómo quedaría el producto que va a recibir. Mediante los prototipos se puede verificar si las especificaciones han sido construidas de acuerdo a los requisitos del sistema. Teniendo como resultado un modelo para el desarrollo del producto atendiendo a las necesidades específicas del cliente.

En el anexo número 1 al final del presente documento se muestra un ejemplo de prototipo de interfaz de usuario.

2.4. Diseño de la solución

A continuación se muestra como quedó el diseño de la solución propuesta a través de los diferentes artefactos generados, como el diagrama de componentes, de diseño, así como el modelo de datos, que representa las tablas de la base de datos.

2.4.1. Diagrama de componentes

El diagrama de componentes se utiliza para modelar la vista estática del sistema. Los componentes pueden ser ejecutables, código fuente, librerías, bases de datos físicas, documentos, etc. A continuación se muestra el diagrama de componentes, que muestra la relación de estos a través del uso de los servicios que prestan unos para el consumo de otros, del subsistema inventario, entre los que se destacan despacho, documento de salida, recepción y transferencia entre áreas, los cuales se alimentan de los servicios que les proveen los componentes documentos, movimientos, productos, útiles y configuración. (20)

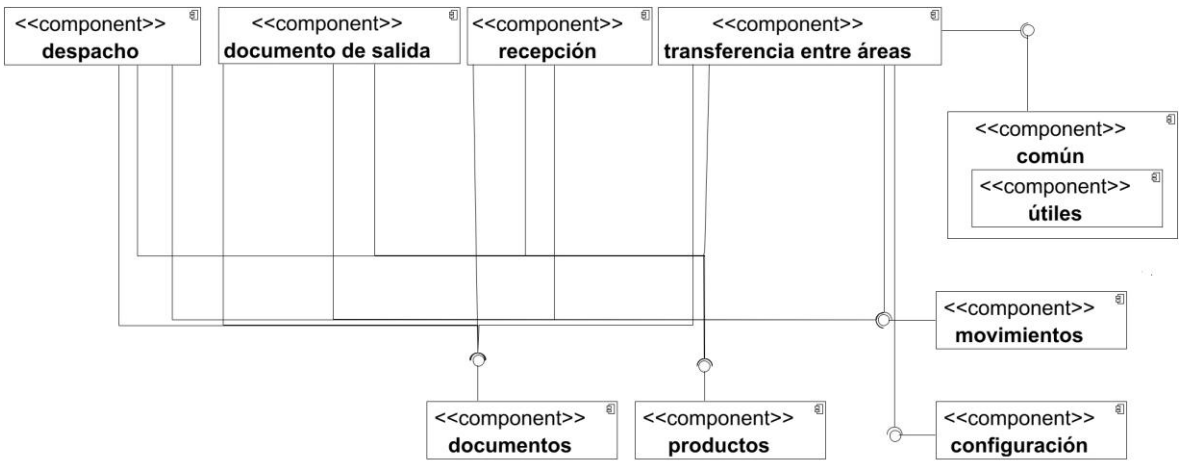

Figura 10: Diagrama de componentes.

Cada vez que se realiza un documento de recepción, despacho, documento de salida o una transferencia, implica movimiento de productos, lo cual se refleja en la estructura de la aplicación a través del diagrama.

2.4.2. Diagrama de clases. Gestionar transferencias entre áreas.

A continuación se muestra como quedó el diseño de la solución propuesta a través del diagrama de diseño para los procesos gestionar transferencias entre áreas y gestionar transferencias entre almacenes.

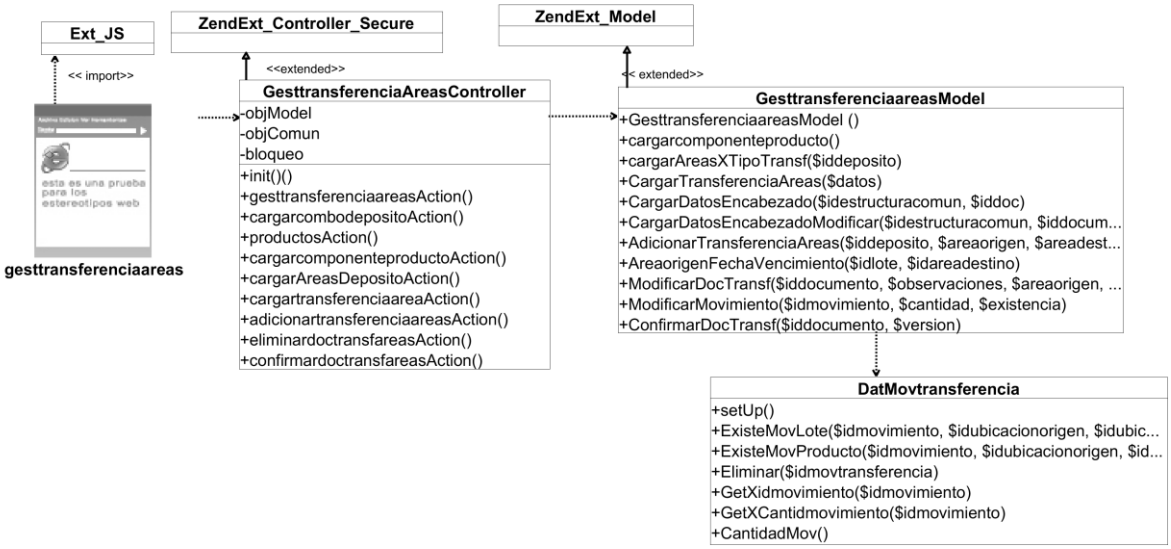

Figura 11: Diagrama de clases del proceso transferencia entre áreas.

Tabla 3: Descripción de clases: GesttransferenciaareasModel.

Nombre de la clase: GesttransferenciaareasModel.

Tipo de clase: (Modelo)	
Para cada responsabilidad:	
Nombre	**Function** GesttransferenciaareasModel ().
Descripción	Constructor de la clase GesttransferenciaareasModel ().
Nombre	**Function** cargarAreasXTipoTransf ($iddeposito).
Descripción	Obtiene las áreas dado un depósito determinado.
Nombre	**Function** CargarTransferenciAareas ($datos).
Descripción	Obtiene los datos de una transferencia entre áreas.
Nombre	**Function** CargarDatosEncabezado ($idestructuracomun, $iddoc).
Descripción	Obtiene los datos de encabezado de una transferencia entre áreas.
Nombre	**Function** CargarDatosEncabezadoModificar ($idestructuracomun, $iddocumento, $idtipotransferencia).
Descripción	Obtiene los datos de encabezado de una transferencia entre áreas a modificar.
Nombre	**Function** AdicionarTransferenciaAreas ($iddeposito, $areaorigen, $areadestino, $tipotransferencia, $observaciones).
Descripción	Adiciona una transferencia entre áreas.
Nombre	**Function** ModificarDocTransf ($iddocumento, $observaciones, $areaorigen, $areadestino).
Descripción	Modifica un documento de una transferencia.
Nombre	**Function** ModificarMovimiento ($idmovimiento, $cantidad, $existencia).
Descripción	Modifica un movimiento de una transferencia.

Nombre	**Function** ConfirmarDocTransf ($iddocumento, $versión).
Descripción	Confirma un documento de una transferencia.
Nombre	**Function** AprobarDocTransf ($iddocumento, $version).
Descripción	Aprueba un documento de transferencia en estado de preparado.
Nombre	**Function** CancelarEstadoDocTransf ($iddocumento, $version).
Descripción	Cancela un documento de transferencia en estado de preparado.
Nombre	**Function** RevertirTransfEliminar ($iddocumento).
Descripción	Revierte la eliminación de un documento de transferencia.
Nombre	**Function** AddProductos ($iddocumento, $productos).
Descripción	Adiciona productos para una transferencia.
Nombre	**Function** CargarProductosDocs ($iddocumento, $datos).
Descripción	Obtiene los datos de los productos de un documento determinado.
Nombre	**Function** EliminarDocTransfarea ($iddocumento, $version, $idareao).
Descripción	Elimina un documento de transferencia entre áreas.
Nombre	**Function** EliminarProducto ($prod, $doc, $idareao='').
Descripción	Elimina un producto de una transferencia.
Nombre	**Function** Cargararbolorigen ($objtransf).
Descripción	Carga los elementos que contiene el área origen.
Nombre	**Function** Cargararboldestino ($objtransf).
Descripción	Carga los elementos que contiene el área destino.

Nombre	**Function** CrearMovimientoTransf ($idmovimiento, $idubicacionorigen, $idubicaciondestino, $cantidad, $idproducto, $idlote, $cantvieja, $idarea).
Descripción	Crea un movimiento de productos de una transferencia, de una ubicación origen a un destino.
Nombre	**Function** actualizarMovimientoTransf ($obj).
Descripción	Actualiza un movimiento de productos de una transferencia y muestra un mensaje cuando se revierte el traspaso de productos, cuando se realiza el traspaso o cuando no se puede realizar el traspaso.
Nombre	**Function** MostrarCantidad ($idmovimiento, $idlote, $idubicacionorigen, $idubicaciondestino).
Descripción	Muestra la cantidad de productos de un movimiento.
Nombre	**Function** obtenerAreas ($iddeposito).
Descripción	Obtiene las áreas de un depósito determinado.
Nombre	**Function** Cargarcombo_Aprobadospor ().
Descripción	Carga los usuarios que han aprobado las transferencias.
Nombre	**Function** cargargriddocs ($idestructuracomun, $iddocumento, $estadodoc, $limit, $start).
Descripción	Carga todos los documentos de transferencias realizados.

2.4.3. Diagrama de clases. Gestionar transferencias entre almacenes.

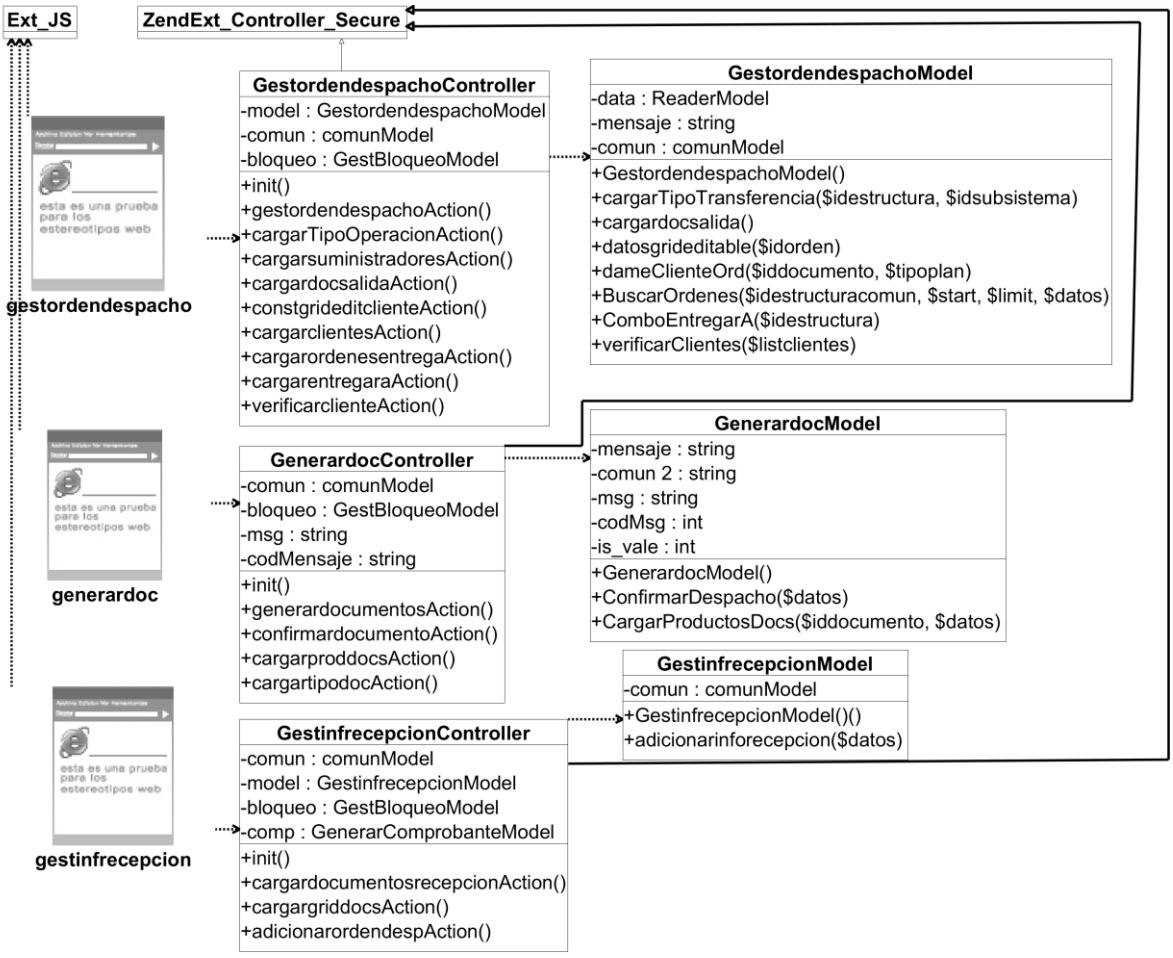

Figura 12: Diagrama de clases del proceso transferencia entre almacenes.

Descripción de las clases principales:

Tabla 4: Descripción de clases: GestordendespachoModel

Nombre de la clase: GestordendespachoModel()	
Tipo de clase: (Modelo)	
Atributo	Tipo
mensaje	String
común	comunModel
Para cada responsabilidad:	

Nombre	**Function** GestordendespachoModel ().
Descripción	Constructor de la clase GestordendespachoModel ().
Nombre	**Function** cargarTipoTransferencia ($idestructura, $idsubsistema).
Descripción	Carga los datos y descripción de los tipos de transferencia.
Nombre	**Function** cargardocsalida ().
Descripción	Carga los documentos de salida.
Nombre	**Function** dameClienteOrd ($iddocumento, $tipoplan).
Descripción	Devuelve los clientes de las órdenes de despacho (autorización o plan).
Nombre	**Function** BuscarOrdenes ($idestructuracomun, $start, $limit, $datos).
Descripción	Devuelve las órdenes de despacho (autorización o plan).
Nombre	**Function** ComboEntregarA ($idestructura).
Descripción	Carga para seleccionar en un combo los depósitos destino de la transferencia.
Nombre	**Function** verificarClientes ($listclientes).
Descripción	Verifica si existe la lista de clientes deseada.

Tabla 5: Descripción de clases: GenerardocModel.

Nombre de la clase: GenerardocModel	
Tipo de clase: (Modelo)	
Atributo	Tipo
mensaje	String

comun2	String
msg	String
codMsg	Entero
is_vale	Entero
Para cada responsabilidad:	
Nombre	**Function** GenerardocModel ().
Descripción	Es el constructor de la clase GenerardocModel ().
Nombre	**Function** ConfirmarDespacho ($datos).
Descripción	Confirma los datos del documento de transferencia generados por despacho.
Nombre	**Function** CargarProductosDocs ($iddocumento, $datos).
Descripción	Carga los productos del documento de salida deseado.

Tabla 7: Descripción de la clase GestestrucubicacionModel

Nombre de la clase: GestestructubicacionModel	
Tipo de clase: (Modelo)	
Para cada responsabilidad:	
Nombre	**function** activarAreas()
Descripción	Se encarga de activar un área dentro de un almacén, teniendo en cuenta que esta no esté ya activa.
Nombre	**function** desactivarAreas()
Descripción	Se encarga de desactivar un área dentro de un almacén, teniendo en cuenta que esté ya activa y no posea productos

	almacenados.
Nombre	**Function** graficarEstructuraUbicacion ().
Descripción	Se encarga de generar gráficos de cada estructura de ubicación del almacén.

2.4.4. Modelo de datos

El modelo de datos permite describir los elementos que intervienen en un problema dado y la forma en que se relacionan dichos elementos entre sí. (20)

Enfocado a cumplir con las exigencias propuestas para el correcto desarrollo del conjunto de funcionalidades que deben realizarse, se confeccionó el siguiente Diagrama Entidad-Relación:

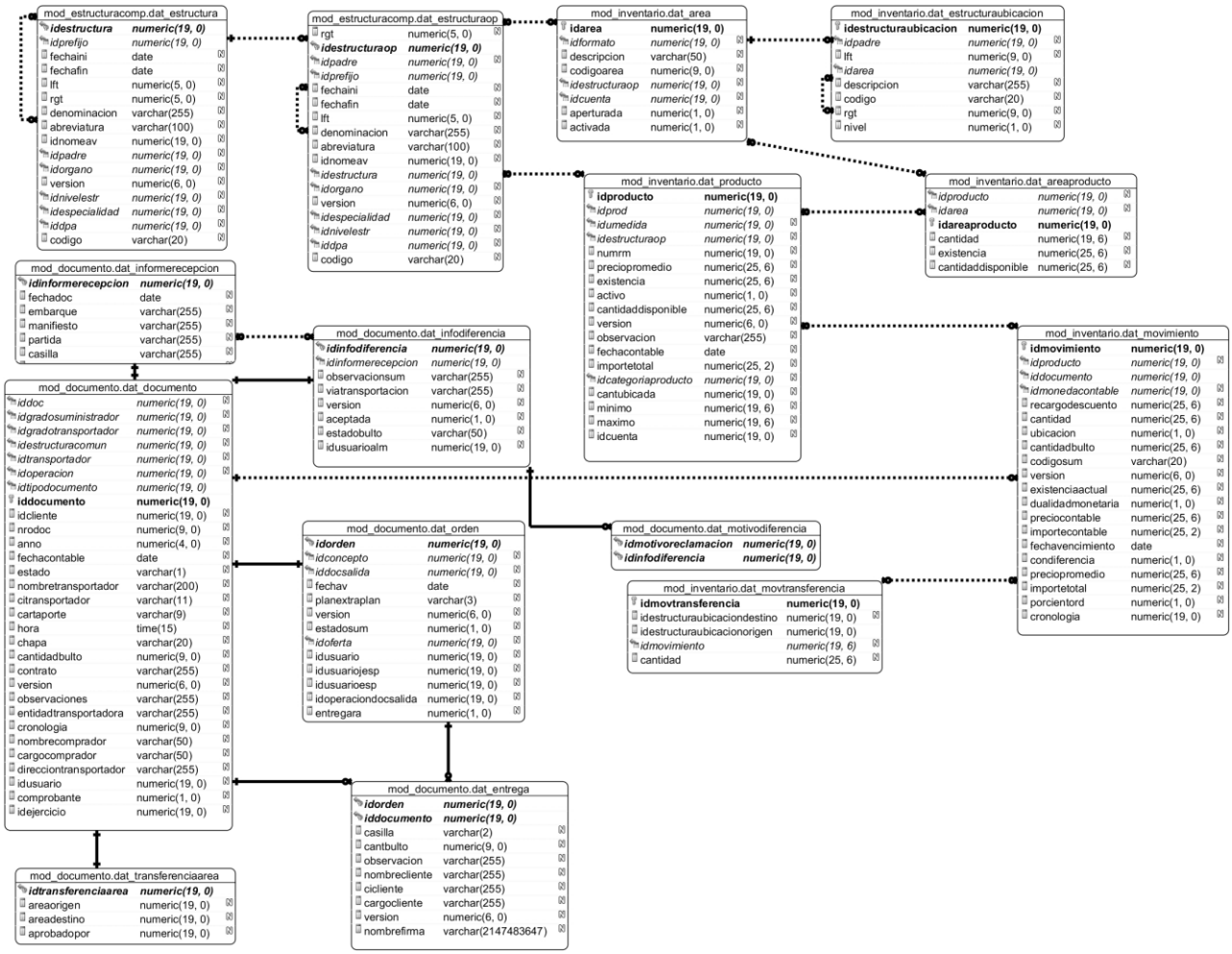

Figura 13: Modelo de datos del componente Transferencia entre áreas.

2.4.4.1. Descripción de las tablas del modelo de datos.

Tabla 8: Descripción de la tabla dat_area

Nombre: dat_area		
Descripción: Almacena los datos referente a las áreas de cada almacén.		
Atributo	**Tipo**	**Descripción**
idarea	numeric	Es el identificador del área.
idformato	numeric	Es el identificador del formato.
descripcion	varchar	Contiene la descripción del área.
codigoarea	numeric	Contiene el código del área.
idestructuraop	numeric	Identificador que proviene de la tabla de las entidades (nom_filaestruc) del esquema de estructura y composición.
idcuenta	numeric	Es el identificador de la cuenta.
aperturada	numeric	Indica si a un área se le realizó la apertura o no.
activada	numeric	Indica si un áreas esta activada o no.

Tabla 9: Descripción de la tabla dat_estructuraubicacion

Nombre: dat_estructuraubicacion
Descripción: Almacena los datos referente a la estructura de ubicación de las áreas dentro de un almacén.

Atributo	Tipo	Descripción
idestructuraubicacion	numeric	Es el identificador de la estructura de ubicación.
idpadre	numeric	Es el identificador de la estructura de ubicación padre.
idarea	numeric	Identificador del área donde está la estructura dentro de un almacén.
lft	numeric	Es el nodo más a la izquierda de la estructura del árbol.
rgt	varchar	Es el nodo más a la derecha de la estructura del árbol.
codigo	varchar	Código de la estructura.
descripcion	numeric	Atributo que describe la estructura de ubicación.
nivel	numeric	Atributo que guarda el nivel de la rama del árbol.

Tabla10: Descripción de la tabla dat_transferenciaarea.

Nombre: dat_transferenciaarea		
Descripción: Almacena los datos referente a las transferencias entre áreas de un mismo almacén.		
Atributo	**Tipo**	**Descripción**
idtransferenciaarea	numeric	Es el identificador de la transferencia entre área.

areaorigen	numeric	Identificador del área de origen.
areadestino	numeric	Identificador del área de destino.
aprobadopor	numeric	Identificador del usuario que aprueba el documento.

2.5. Patrones

Los patrones son modelos que nos brindan soluciones eficientes y que ayudan a estudiar el comportamiento, desarrollo y estructura de diversos componentes. Son guías que describen de alguna forma un proceso o procedimiento, así como a entender otros basándonos en el conocimiento previo de estos. (22)

2.5.1. Patrones de diseño

2.5.1.1 Patrones GOF (Gangs of Four)

Fachada: El patrón fachada viene motivado por la necesidad de estructurar un entorno de programación y reducir su complejidad con la división en subsistemas, minimizando las comunicaciones y dependencias entre éstos. Se aplica cuando se necesita proporcionar una interfaz simple para un subsistema complejo, o cuando se quiera estructurar varios subsistemas en capas, ya que las fachadas serían el punto de entrada a cada nivel. En la aplicación se pone de manifiesto con el uso de las clases “Services” en cada componente, pues estas proporcionan el acceso a los diferentes servicios sin tener que referenciar cada funcionalidad en cada clase.

2.5.1.2. Patrones de asignación de responsabilidades (GRASP)

Experto: Se puso en práctica en el componente de transferencia entre áreas, con el uso de clases que poseen responsabilidades específicas a cumplir de acuerdo con la información que manejan, el componente cuenta con clases controladoras, modelos y entidades que poseen funciones concretas de acuerdo con la información que gestiona. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Ejemplo: La clase controladora es la encargada de manejar las vistas, ya que cuenta con la mayor información acerca de estas.

Controlador: Sugiere que la lógica de negocios debe estar separada de la capa de presentación, para aumentar la reutilización de código y a la vez tener un mayor control. Se utiliza cuando la aplicación es muy extensa, de esta forma, en el módulo en vez de tener un solo controlador y saturarlo, se tienen clases ***Controllers***, que son controladores más pequeños especializados en las funcionalidades de cada componente. Un ejemplo de ello es la clase **GesttransferenciaAreasController.**

Bajo acoplamiento: Busca tener las clases lo menos relacionadas entre sí posible, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. Las clases del componente de transferencia entre áreas fueron diseñadas bajo este principio, para que solamente se establezcan las relaciones necesarias entre ellas.

Alta cohesión: Existe afinidad entre cada clase y los métodos que implementan. Éstas poseen responsabilidades vinculadas acordes a la información que controlan; y colaboran con otros objetos para compartir el esfuerzo si la tarea es grande, facilitando su mantenimiento y reutilización.

2.6. Métricas de diseño

Un aspecto importante a tener en cuenta en la evaluación del diseño, es la creación de métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objetos. Las métricas del diseño permiten medir de forma cuantitativa la calidad de los atributos internos del software. (23) Los atributos de calidad que tienen en cuenta estas métricas son los siguientes:

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

Complejidad de implementación: Es la complejidad de implementación que posee una estructura de diseño de clases.

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

Acoplamiento: Es el grado de dependencia o interconexión de una clase o estructura de clase, con otras.

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

Cantidad de pruebas: Es el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado. (23)

Las métricas seleccionadas como instrumento para medir la calidad del diseño de la herramienta son las siguientes:

Tamaño Operacional de Clase (TOC): Está dado por el número de métodos asignados a una clase y evalúa los siguientes atributos de calidad (23):

Tabla 11: Atributos de calidad evaluados por la métrica TOC.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Aumento del TOC provoca aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Aumento del TOC provoca aumento de la complejidad de implementación de la clase.
Reutilización	Aumento del TOC provoca disminución del grado de reutilización de la clase.

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

Tabla 12: Criterios de evaluación para la métrica TOC.

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

implementación	Alta	>2*Promedio.
Reutilización	Baja	>2*Promedio.
	Media	Entre Promedio y 2*Promedio.
	Alta	<=Promedio.

Relaciones entre Clases (RC): Está dado por el número de relaciones de uso de una clase con otra y evalúa los siguientes atributos de calidad (23):

Tabla 13: Atributos de calidad evaluados por la métrica RC.

Atributo de calidad	**Modo en que lo afecta**
Acoplamiento	Aumento del RC provoca aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Aumento del RC provoca aumento de la complejidad del mantenimiento de la clase.
Reutilización	Aumento del RC provoca disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Aumento del RC provoca aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

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

Tabla 14: Criterios de evaluación para la métrica RC.

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

Cantidad de pruebas	Baja	<=Promedio.
	Media	Entre Promedio y 2*Promedio.
	Alta	>2*Promedio.

A continuación se muestra los resultados que arrojaron los indicadores que mide la métrica TOC:

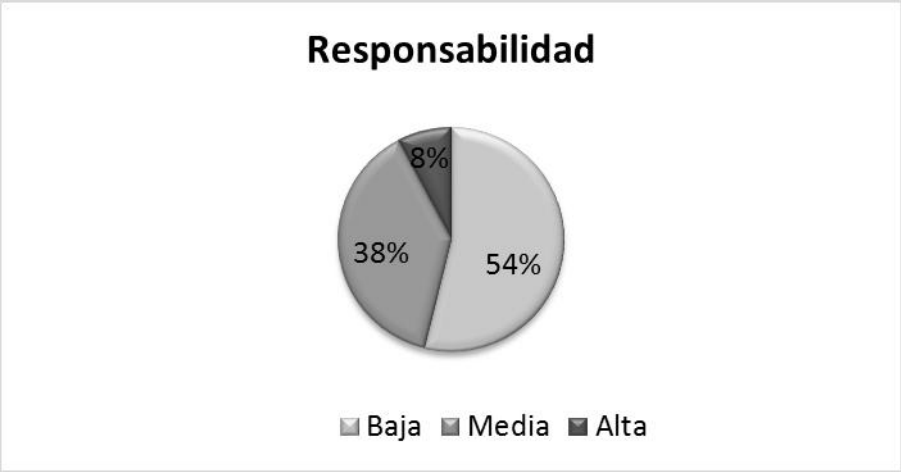

Figura 14: Representación de responsabilidad de la métrica TOC.

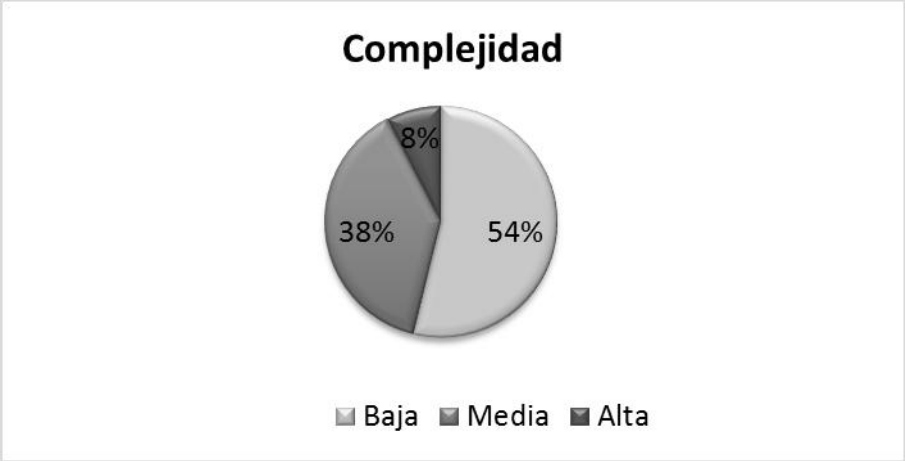

Figura 15: Representación de la complejidad de la métrica TOC.

Figura 16: Representación de la reutilización de la métrica TOC.

Después del análisis realizado basándose en los atributos de calidad definidos se llega a la conclusión de que existe una baja responsabilidad de las clases validando que un 54% de los procedimientos son menores que el promedio de procedimientos por clases, esto influye en que exista una baja complejidad de implementación contando solo con alta complejidad el 8% de los procedimientos, teniendo solamente con poca capacidad de reutilización un 8% de los procedimientos, lo que influye en que los atributos de calidad no se vean afectados en gran medida.

A continuación se muestran los resultados de la métrica RC:

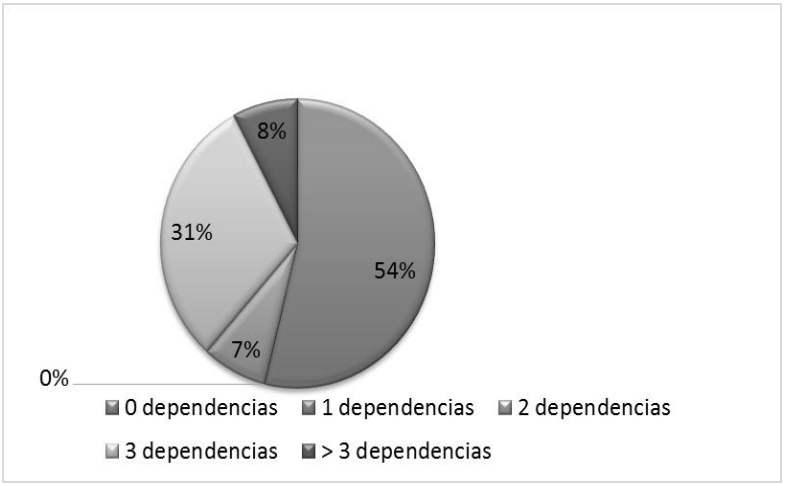

Figura 17: Representación de la dependencia entre clases.

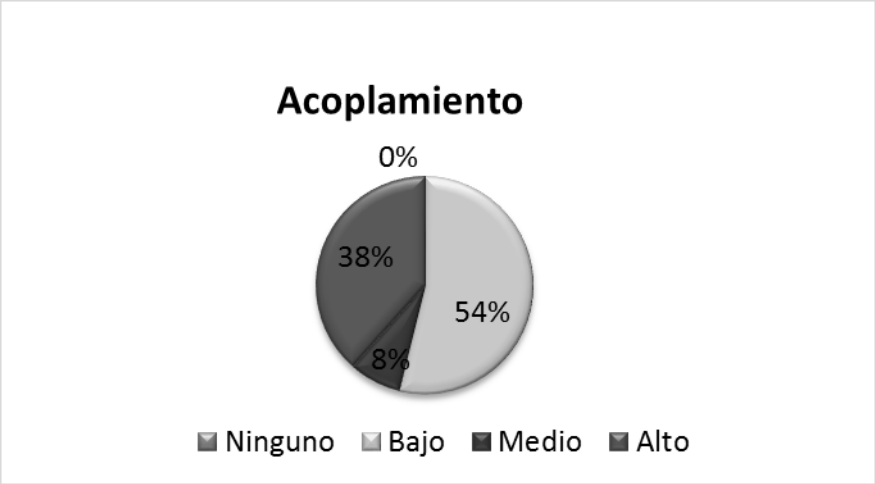

Figura 18: Representación del acoplamiento.

Figura 19: Representación de la complejidad de mantenimiento.

Figura 20: Representación de la cantidad de pruebas.

Figura 21: Representación del atributo reutilización.

Basado en el principio de independencia funcional que expresa que esta es una clave para el buen diseño y este a su vez una clave para lograr calidad del software (24), se puede concluir que, contando con aproximadamente un 61% de clases con 2 o menos dependencias (Fig. 21), la calidad del software es aceptable.

Conclusiones

En este capítulo se analizaron los procesos necesarios para realizar las transferencias de productos entre áreas y almacenes, permitiendo así visualizar cómo funcionan los mismos dentro del ERP Cedrux. El análisis de la arquitectura de Cedrux y del propio módulo realizado en este capítulo, facilitó la comprensión de la estructura de ambos, así como un mejor entendimiento del diseño de forma general. Todo el material expuesto en este capítulo conforma las bases a seguir para lograr una implementación acorde a la estructura que posee actualmente el sistema en explotación e incorporando las necesidades surgidas para dar solución al problema en cuestión.

Capítulo 3: Implementación y Pruebas

Introducción

El presente capítulo tiene como objetivo explicar, a través del código originado de la aplicación terminada, cómo se realizó la mejora del control de los materiales en los almacenes. Se presentan además los diagramas de despliegue y componentes, así como el conjunto de pruebas escogidas que se realizan a la aplicación para validar la propuesta de solución.

3.1. Estructura del marco de trabajo

A continuación se precisa la forma en que se define la estructura donde se ubican cada uno de los elementos dentro de la aplicación, de forma tal que facilite el trabajo y la búsqueda.

La carpeta denominada ***apps*** es donde se almacenan los controladores y modelos de cada uno de los componentes correspondientes a los subsistemas. En la figura 21 se muestra dicho contenido para el componente transferencia entre áreas del subsistema inventario.

Figura 22: Carpeta de aplicación correspondiente al subsistema inventario.

El componente transferencia entre áreas va a contener un conjunto de paquetes que serán especificados a continuación:

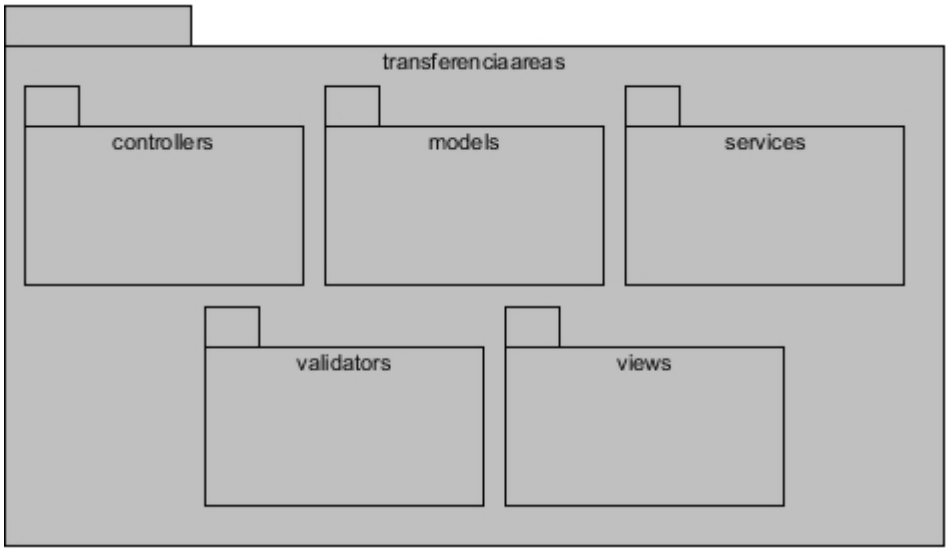

Figura 23: Paquete correspondiente al componente transferencia entre áreas.

En el paquete ***controllers*** se encontrarán las clases controladoras, encargadas de gestionar las funcionalidades del sistema. El paquete ***models*** estará estructurado de la siguiente forma:

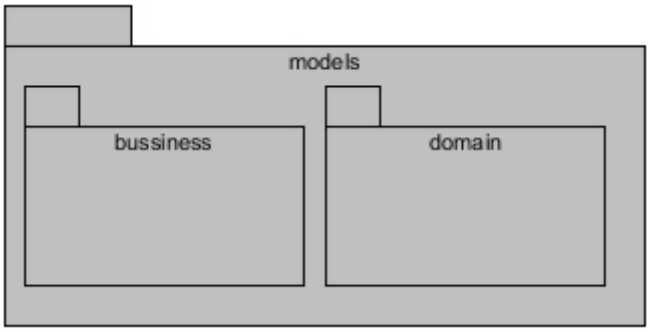

Figura 24: Paquete model correspondiente a la carpeta de aplicación.

Contiene dos paquetes para agrupar clases: ***business y domain.*** El primero contendrá las clases necesarias para acceder a los datos que persisten en la base de datos y el segundo debe contener las clases generadas por el ORM Doctrine a partir de cada una de las tablas existentes en la base de datos. Cada una de estas clases heredará de una clase generada igualmente por el Doctrine, las cuales se ubican en otro paquete dentro de este llamado ***generated***.

El paquete ***views*** contendrá el paquete scripts. Se encarga de contener las páginas clientes (archivo phtml), o sea, contiene las direcciones de los archivos JavaScript, el título de la página, etc.

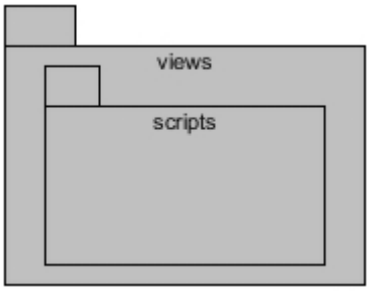

Figura 25: Paquete views correspondiente a la carpeta de aplicación.

Al igual que la carpeta ***apps,*** mencionada anteriormente, debe existir además una carpeta que contiene las vistas de los subsistemas y componentes. Dicha carpeta se denomina ***web***.

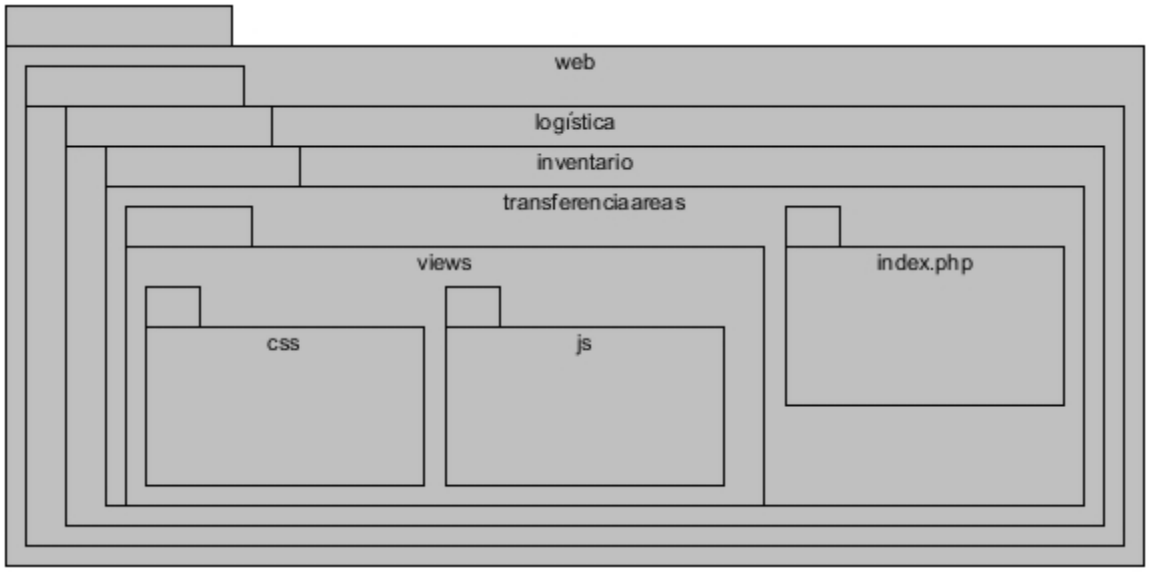

Figura 26: Carpeta de diseño correspondiente al componente transferencia entre áreas.

Index.php: Este fichero incluye la dirección del archivo de configuración y a través de este inicializa la aplicación para que se carguen en la misma un conjunto de componentes necesarios para su funcionamiento. Su código permanece igual para todos los componentes.

La carpeta ***view*** contendrá los ***css*** y los ***js*** que se explicarán a continuación. El paquete ***css*** incluirá las clases necesarias para estructurar gráficamente el componente. El paquete ***js*** comprenderá las clases JavaScript necesarias para que el usuario interactúe con el sistema y obtenga los resultados necesarios. Está compuesto por paquetes con los nombres de las funcionalidades del componente, como se muestra a continuación.

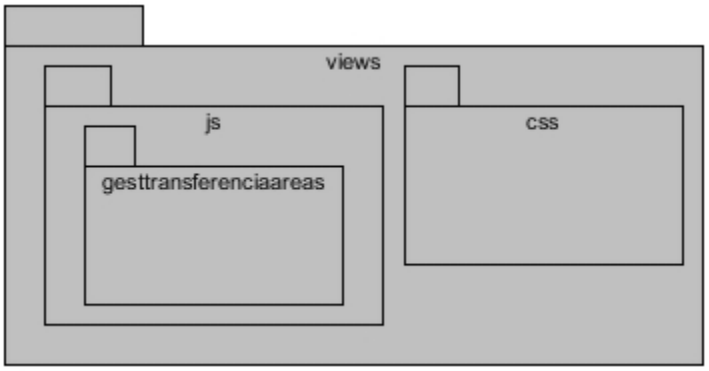

Figura 27: Paquete views correspondiente a la carpeta de diseño.

3.1.1. Nomenclatura de las clases

Clases vistas: Se definió para las clases que se encuentran dentro de Views que el nombre se escribirá sin ningún sufijo o prefijo. Los nombres se escribirán en notación camello (Camel Case). Ejemplo: **gesttransferenciaareas.js.**

Clases controladoras: Se definió para las clases controladoras que después del nombre se les colocará como sufijo la palabra: “Controller”. Ejemplo: GesttransferenciaAreasController.php

Clases modelos:

Business (Negocio). Se definió para las clases que se encuentran dentro de Business que después del nombre se les colocará como sufijo la palabra: “Model”. Ejemplo: GesttransferenciaareasModel.php.

Domain (Dominio) y Bases del Dominio (Generated). Se definió para las clases que se encuentran dentro de Domain que el nombre se escribirá con el prefijo “Dat”. Ejemplo: DatMovtransferencia.php. Para las clases que se encuentran dentro del Generated se definió que el nombre se escribirá con el prefijo “BaseDat”. Ejemplo: BaseDatMovtransfeencia.php.

3.1.2. Nomenclatura de las funciones

Se definió que el nombre de las funciones se escribe con la primera palabra en minúscula, para el caso, específico de que sea un nombre compuesto se empleará la notación CamelCasing (notación camello) y de forma tal que al leerlo se conozca el objetivo de la misma. Ejemplo: adicionarTransferencia.

En caso particular que la función sea una acción de la clase controladora u otra se escribe después del nombre de la función la palabra “Action” como sufijo. Ejemplo: gesttransferenciaareasAction ().

3.1.3. Nomenclatura de las variables

Se definió que el nombre a emplear para las variables se escribe en minúscula y en caso de ser un nombre compuesto se escriben ambas palabras en minúscula separadas por un guión bajo. Ejemplo: *variable, variable_compuesta.*

3.2. Validación de la solución

A partir del planteamiento efectuado en el primer capítulo del presente documento, que establecía que con la adición, a partir del desarrollo de funcionalidades, al subsistema inventario de Cedrux, se podría mejorar la calidad con que se realiza el control de los materiales en los almacenes de las entidades cubanas, erradicando así algunas carencias que presenta el producto funcional, realizado para la gestión de dichos procesos, a continuación se proponen algunos aspectos a tener en cuenta para la validación de la propuesta de solución.

Ante todo, la **calidad** es la aptitud que posee cualquier producto para satisfacer las necesidades del cliente. El sistema, hasta el momento en que se realiza el estudio preliminar de la investigación, muestra un alto grado de aceptación por parte de este, pero aún existen complicaciones a la hora de realizar algunas acciones de forma eficiente, como transferir productos entre áreas y almacenes, no se conoce la estructura de un almacén específico; en caso de necesitar productos que no se encuentren en el almacén, si en la entrega que se planifica no arriba ninguna cantidad de estos, no existe forma de hacer una reclamación donde se incluyan los mismos, no posee funcionalidades que le brinden activar o desactivar áreas, en caso de que se tenga que reparar, ampliar, etc. En resumen, aún persisten disfunciones en el entorno de la aplicación que hacen que algunos procesos se realicen de forma engorrosa y por tanto no cuentan con el grado de calidad que satisface en su totalidad al cliente. En aras de cambiar esta situación, se implementan funcionalidades que proveen una solución a los problemas que anteriormente se destacan.

Tabla 15: Comparación de la realización de procesos antes y después de la investigación.

Procesos antes de la investigación	**Procesos después de la investigación**

Transferencia entre áreas. (Se realizaba a través de ajustes, por baja en el almacén que transfería el producto, y por sobrante al que se traslada el mismo, siendo difícil y confuso el proceso).	Transferencia entre áreas. (Se realiza en un solo componente, siendo más rápido y efectivo el proceso.
Transferencia entre almacenes. (Se realizaba a través de ajustes, por baja en el almacén que transfería el producto, y por sobrante al que se traslada el mismo, siendo difícil y confuso el proceso).	Transferencia entre áreas. (Se realiza a través de 3 componentes pero el proceso es más rápido y efectivo.
Recepción de productos no existentes en el almacén. (Si el producto no estaba previamente en el almacén y no llegaba la cantidad pactada, como entraba por el proceso de recepción no se podía añadir al informe de diferencias)	Recepción de productos no existentes en el almacén. (Se puede agregar fácilmente el producto del nomenclador, y adicionar su falta e el informe de diferencias)
Apertura de áreas a través de la recepción. (No se realizaba de ninguna forma)	Apertura de áreas a través de la recepción. (Una vez que se ubican productos en el área a través el proceso de la recepción, y se confirma el documento, se apertura el área)
Apertura de almacenes a través de la primera recepción. (No se realizaba de ninguna forma)	Apertura de almacenes a través de la primera recepción. (Una vez que se ubican productos en un área perteneciente a un almacén sin apertura a través el proceso de la recepción, y se confirma el documento, se apertura el área)
Activar/Desactivar áreas. (No se realizaba de ninguna forma)	Activar/Desactivar áreas. (El área puede ser activada y desactivada en cualquier momento, validando en el segundo caso que no contenga productos ubicados en la misma)

Graficar estructura de ubicación. (No se realizaba de ninguna forma)	Graficar estructura de ubicación. (Se puede conocer la estructura por áreas, secciones, estantes, etc.)

Luego de ver los resultados que se muestran en la tabla anterior, se llega a la conclusión de que efectivamente las nuevas funcionalidades incluidas proporcionan calidad, rapidez y facilidad a la aplicación para gestionar los procesos de inventario.

3.3. Pruebas de software

La prueba es un proceso de ejecución de un programa con la intención de descubrir errores. El objetivo de la prueba es diseñar pruebas que identifiquen errores con la menor cantidad de tiempo y espacio. (20)

A continuación se presenta la propuesta para realizar las pruebas a la aplicación.

3.3.1. Pruebas internas

Las pruebas internas, como se había descrito brevemente en el modelo de desarrollo, son realizadas por el grupo de calidad del centro verificando el resultado de la implementación.

Conformada por actividades que parten de la planificación de la etapa en que se realizan, las pruebas internas están además compuestas por una serie de subprocesos, dentro de los cuales se encuentra fundamentalmente la ejecución de las pruebas. (20)

En este subproceso se realizan actividades que parten de la entrega de los artefactos, la revisión, la distribución del trabajo en caso de estar correctas y la realización de pruebas exploratorias. Consecuentemente con la aprobación de estas últimas, se procede al desarrollo de la iteración de las pruebas y la resolución de las no conformidades, finalizando con la realización de pruebas de regresión. (20)

Cualquier prueba, en este caso interna, puede realizarse de una de dos formas:

- Se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa, las cuales se denominan pruebas de caja negra.
- Se permiten desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada, las cuales se denominan pruebas de caja blanca.

3.3.1.1. Pruebas de caja negra

Las pruebas de caja negra, también llamadas de comportamiento, se concentran en los requisitos funcionales del software, lo que permite derivar conjuntos de condiciones de entrada que ejercitarán por completo todas las funcionalidades de un programa. (20)

Las pruebas se llevan a cabo sobre la interfaz del software, y es completamente indiferente el comportamiento interno y la estructura del programa. Los casos de prueba de caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta.
- La integridad de la información externa se mantiene.

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

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

Como método para realizar este tipo de prueba se escogió la partición de equivalencia, ya que se encarga de definir un caso de prueba que descubra estos errores, reduciendo el número total de casos de prueba que deben desarrollarse. (20)

3.3.1.1. Diseño de caso de pruebas.

Los diseños de casos de pruebas para la partición de equivalencia se basan en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia a su vez representa un conjunto de estados válidos y no válidos para las condiciones de entrada. Las clases de equivalencia se definen de acuerdo a lo siguiente:

- Si una condición de entrada especifica un rango, se definen una clase de equivalencia válida y dos no válidas.

- Si una condición de entrada requiere un valor específico, se definen una clase de equivalencia válida y dos no válidas.
- Si una condición de entrada especifica un miembro de un conjunto, se definen una clase de equivalencia válida y otra no válida.
- Si una condición de entrada es booleana, se definen una clase de equivalencia válida y otra no válida. (20)

Los casos de pruebas se diseñaron para que fuera probada la herramienta obtenida en el proceso de desarrollo, y se le realizaran pruebas por el colectivo de calidad del proyecto ERP-Cuba. Los mismos se pueden encontrar en la documentación del presente trabajo, con los artefactos generados, entregada al proyecto ERP-Cuba. A continuación se relaciona un ejemplo:

Caso de prueba **Generar gráfico de estructura.**

Las especificaciones para llevar a cabo una correcta revisión de la herramienta se detallan a continuación.

Condiciones de ejecución:

- Debe seleccionar el menú Subsistemas/Logística/Inventario/Gestionar estructura ubicación

Tabla 16: Caso de prueba Generar gráfico de estructura de ubicación.

Escenario	**Descripción**	**Graficar**	**Respuesta del sistema**	**Flujo Central**
EC 1.1: Generar gráfico de estructura de ubicación	En este escenario se genera el gráfico de la estructura de ubicación de un almacén.	V (True)	El sistema muestra en un panel adyacente la figura.	Seleccionar el botón que genera el gráfico.
		I (False)	El sistema no realiza acción alguna.	No se selecciona el botón de graficar.

Tabla 17: Descripción de las variables.

No	**Nombre del campo**	**Clasificación**	**Puede ser nulo**	**Descripción**

1	Graficar	Campo de selección	No	El campo recoge la opción para graficar una estructura determinada como valor verdadero o falso.

3.3.1.2. Resultados de las pruebas

En la Tabla 18 se muestran los resultados obtenidos a través de los procesos de revisión por parte de calidad interna del centro:

Tabla 18: Resultados de los procesos de revisión.

Entregas	N/C Significativas	N/C No significativas	Total
1era Entrega 16/05/2013	6	7	13
2da Entrega 31/05/2013	3	5	8
3era Entrega 14/06/2013	0	2	2

Todas las no conformidades detectadas fueron resueltas, quedando liberada por calidad interna para pasar a las siguientes etapas de pruebas.

3.3.1.2. Pruebas de caja blanca

La prueba de caja blanca es un método de diseño de casos de prueba que usa la estructura de control de diseño procedimental para obtener los casos de prueba. (20)

Permite obtener casos de pruebas que:

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

Entre las técnicas de prueba de caja blanca se selecciona aplicar a la solución propuesta la prueba de camino básico, que permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental. (20)

El análisis de la complejidad ciclomática de un algoritmo define por tanto el número de caminos independientes del conjunto básico del programa y proporciona el número mínimo de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez, lo cual fue planteado por Thomas McCabe en 1976, y establece una concordancia entre los valores resultantes de aplicar la métrica y el nivel de riesgo que implicaría probar, entender y modificar el código. (20)

Dicha concordancia se muestra en una tabla a continuación:

Tabla 19: Complejidad ciclomática vs evaluación de riesgo.

Complejidad ciclomática	**Evaluación de riesgo**
1-10	Programa simple, sin mucho riesgo.
11-20	Más complejo, riesgo moderado.
21-50	Complejo, programa de alto riesgo.
50	Programa imposible de probar, muy alto riesgo.

Para la realización de la complejidad ciclomática, se procede a enumerar las sentencias del código y a partir del mismo se construye el grafo de flujo asociado, atendiendo los siguientes componentes:

Nodo: Círculos del grafo de flujo que representan una o más secuencias procedimentales.

Aristas: Saetas que unen los nodos del grafo, representan el flujo de control.

Regiones: Son las áreas delimitadas por las aristas y nodos; el área exterior del grafo también se cuenta como una región más.

```
public function graficarEstructuraUbicacionAction() {
        $arrDatos = array(); 1
        $arrDatosInternos = array();1
        $subsistema = $this->_request->getPost('subsistema');1
        $pos = $this->_request->getPost('pos');1
        if ($pos == 1)2
            $iddeposito = $this->_request->getPost('iddeposito');3
        else
            $iddeposito = $this->_request->getPost('deposito');4
        $pos = $this->_request->getPost('pos');5
        $idarea = $this->_request->getPost('idarea');5
        $idestructuraubic = $this->_request->getPost('idestructuraubic');5
        $nivel = (($this->_request->getPost('nivel') || $this->_request->getPost('nivel') === '0') &&
            $this->_request->getPost('nivel') != 'undefined' ) ? $this->_request->getPost('nivel') + 1 : '';5
        $arrDatos = GestestructubicacionModel::cargararbol($subsistema, $iddeposito, $pos, $idarea,
            $idestructuraubic, $nivel, 2, 1);5

        $idaux = 0;5

        for ($index = 0; $index < count($arrDatos); $index++) {6
            if ($pos + 1 == 1) {7
                $idaux = $arrDatos[$index]['id'];8
            } else {
                $idaux = $arrDatos[$index]['deposito'];9
            }
            $nivel2 = (($arrDatos[$index]['nivel'] || $arrDatos[$index]['nivel'] === '0') &&
                $arrDatos[$index]['nivel'] != 'undefined' ) ? $arrDatos[$index]['nivel'] + 1 : ''; 10
            $arrDatosInternos[$index] = GestestructubicacionModel::cargararbol($arrDatos[$index]['subsistema'],
                $idaux, $pos + 1, $arrDatos[$index]['idarea'], $arrDatos[$index]['idestructuraubic'], $nivel2, 2, 1); 10
        }

        $retorno = Array();11
        $retorno['ubicacion_imagen'] = $this->pintar->graficar($arrDatos, $arrDatosInternos);11
        echo json_encode($retorno);11
    }
```

Figura 28: Código fuente de la funcionalidad Generar gráfico de estructura de ubicación.

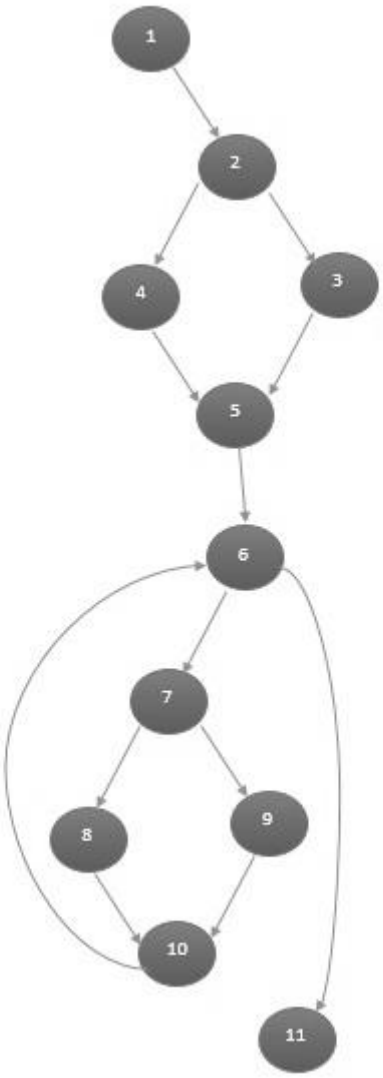

Figura 29: Grafo de flujo asociado a la funcionalidad Graficar estructura de ubicación.

Luego de haber construido el grafo se realiza el cálculo de la complejidad ciclomática mediante las tres fórmulas descritas a continuación, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

V (G)= (A-N) + 2 // Siendo “A” la cantidad total de aristas y “N” la cantidad total de nodos.

V (G)= (13-11) + 2

V (G)= 4

V (G) = P+1 // Siendo “P” la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G)=3+1

V (G)= 4

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

V (G)= 4

El cálculo efectuado mediante las tres fórmulas ha dado el mismo valor, dando como resultado 4, lo que indica que existen 4 posibles caminos donde el flujo básico puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Seguidamente se representan los caminos básicos por los que puede recorrer el flujo:

Camino # 1: 1, 2, 3, 5, 6, 11.

Camino #2: 1, 2, 4, 5, 6, 11.

Camino #3: 1, 2, 3, 5, 6, 7, 9, 10, 6, 11.

Camino #4: 1, 2, 4, 5, 6, 7, 8, 10, 6, 11.

De acuerdo con el resultado obtenido, la función analizada se clasifica como un algoritmo simple sin mucho riesgo para el sistema, ya que solo devolvió 4 caminos para probar al menos una vez cada sentencia.

Para cada camino se realiza un caso de prueba, los cuales se pueden encontrar en la documentación que contiene los artefactos generados para validar el paso por paso del modelo de desarrollo del presente trabajo. A continuación se muestra un ejemplo de uno de ellos:

Tabla 20: Caso de prueba para el camino básico #1

Entradas	**Resultados esperados**	**Resultados reales**
$arrDatos = array(); $arrDatosInternos = array(); $subsistema = $this->_request->getPost('subsistema'); $pos = $this->_request->getPost('pos'); $if($pos==1)	$iddeposito = $this->_request->getPost('iddeposito');	$iddeposito = $this->_request->getPost('iddeposito');

$pos=$this->_request->getPost('pos'); $idarea=$this->_request->getPost('idarea');5 $idestructuraubic=$this->_request->getPost('idestructuraubic'); for($index=0;$index<count($arrDatos);$index++){	$retorno=Array(); $retorno['ubicacion_imagen']=$this->pintar->graficar($arrDatos, $arrDatosInternos);11 echojson_encode($retorno);	

Conclusiones

En este capítulo se presentaron los estándares de codificación establecidos para Cedrux, permitiendo la obtención del código del subsistema de manera organizada, siendo este práctico y entendible para cualquier programador. Con la descripción de las principales clases y de sus correspondientes métodos se logró la comprensión del flujo entre ellas para la implementación de las funcionalidades correspondientes a los módulos Transferencia entre áreas, Recepción, Despacho y Documento de salida, desarrolladas con tecnologías libres. Especialmente el módulo Transferencia entre áreas fue probado mediante pruebas de caja blanca y caja negra, quedando validados mediante pruebas de aceptación y demostrando que cumplen con los requerimientos necesarios para satisfacer las necesidades del cliente. Quedó demostrado que se reduce el tiempo necesario para la realización de las funcionalidades necesarias para llevar a cabo procesos de transferencias, recepciones de productos, estructura de los productos, entre otros.

Conclusiones generales

Con la realización del presente trabajo de diploma se puede afirmar que se logró alcanzar el objetivo general propuesto, luego de realizar todas las tareas trazadas al inicio de la investigación. El desempeño de los objetivos específicos permitió arribar a las siguientes conclusiones:

- La investigación realizada sobre los sistemas ERP a nivel nacional e internacional, sobre todo del sistema integral de gestión empresarial Cedrux, manifestó la necesidad de desarrollar funcionalidades para mejorar los procesos transferencia entre áreas y almacenes.
- El estudio y caracterización de las tecnologías y herramientas definidas para el desarrollo de la aplicación contribuyeron a su correcta utilización garantizando la disminución de costos por concepto de licencias y soporte.
- Las disciplinas de modelado de negocio y requerimientos del modelo de desarrollo propuesto, contribuyeron a una mayor comprensión del negocio y entendimiento de las necesidades del cliente, permitiendo realizar la captura, especificación y validación de los requisitos.
- La elaboración de forma detallada del análisis y diseño, posibilitó mejoras para organizar y documentar el análisis para el desarrollo del software, incidiendo notablemente en la estructura, organización y reutilización de código a través del diagrama de clases del diseño y de los patrones de diseño empleados.
- La disciplina de implementación proporcionó la creación de funcionalidades y la integración satisfactoria de las mismas al ERP Cedrux, lo que implica una gestión eficiente de los procesos de inventario.
- La definición de la propuesta de validación, posibilitó confirmar la estabilidad de los procesos y funcionalidades agregadas a la aplicación.

Con el desarrollo del presente trabajo se consolidaron los conocimientos adquiridos durante la carrera.

Recomendaciones

Para el desarrollo y perfeccionamiento de nuevas funcionalidades relacionadas con los procesos de inventario transferencias entre áreas y almacenes se recomienda:

- Proporcionar dinamismo al componente que se encarga de graficar la estructura de ubicación, de forma tal que se pueda graficar a partir de eventos de clic sobre las estructuras, posibilitando una mayor interacción con el usuario.
- Realizar funcionalidades que generen el informe de diferencia para la transferencia entre almacenes.
- Agregar funcionalidades que permitan realizar transferencias entre dependencias.

Referencias bibliográficas

1. Agipem, una firma de profesionales. Asesoría para la gestión y planificación empresarial. Asesoría para la gestión y planificación empresarial. [En línea] 26 de 10 de 2011. [Citado el: 01 de 01 de 2013.] httpwww.agipem.comindex.phpp=planificacion-economica-financiera.

2. Saily Oliva Martínez, Yaima Álvarez Márquez, Ing. Lester Antonio González Gutiérrez. Sistema para el control de inventarios del ERP . La Habana : s.n., 2010.

3. Ing. Tamara Rodríguez Sánchez, Ing. Mairelys Fernández González, Ing. Eliecer Cabrera Casas. La gestión empresarial de las entidades cubanas. Cedrux a la vuelta de la esquina. La Habana : s.n., 2011.

4. Ing. Virgen Damaris Quevedo Campins, Msc. Meylin Martínez Chong, Ing. Tte Lester Antonio González Gutierrez, Ing. Alien Fernández Fuentes. CEDRUX. Solución para sistemas de control logístico. . La Habana : s.n., 2010.

5. Manual de usuarios del módulo Inventario de Cedrux. La Habana : s.n., 2010.

6. Rumbaugh, James, Jacobson, Ivar y Booch, Grady . El Lenguaje Unificado de modelado. 1999.

7. The Apache Software Foundation. The Apache Software Foundation. The Apache Software Foundation. [En línea] 2012. [Citado el: 05 de 03 de 2013.] http://httpd.apache.org/docs/2.2/es/new_features_2_0.html.

8. PostgreSQL. The world's most advanced open source data base. PostgreSQL. The world's most advanced open source data base. [En línea] 1996. [Citado el: 05 de 03 de 2013.] http://www.postgresql.org.

9. Mozilla Europe. Notas de publicación de Firefox 3.6. Notas de publicación de Firefox 3.6. [En línea] 2009. [Citado el: 30 de 01 de 2013.] httpwww.mozilla-europe.orgesfirefox3.6releasenotes.

10. RapidSVN. RapidSVN. [En línea] http://www.rapidsvn.org/index.php/Main_Page.

11. NetBeans Community. [En línea] 2013. https://netbeans.org/community/releases/72/.

12. Guide to Doctrine for PHP. 2010.

13. Zammetti, Frank W. Practical Ext JS Projects with Gears. New York : Springer-Verlag , 2009.

14. Allen, Rob. Getting started with ZendFramework. [En línea] 2006, 2010. http://alemohamad.com/tutorial-zend-framework/.

15. Mariño, Carlos Vázquez. Programación en PHP 5. Nivel Básico. Ferrol : s.n., 2008.

16. Sánchez, Jorge. JavaScript. Manual de referencia. 2003.

17. Centro de Informatización de la Gestión de Entidades. Subdirección de Producción. Modelo de Desarrollo de Software. La Habana : s.n., 2012.

18. Suárez, José de Jesús Hernández. Entérate en línea. Internet,cómputo y telecomunicaciones. Entérate en línea. Internet,cómputo y telecomunicaciones. [En línea] 02 de 2006. [Citado el: 30 de 01 de 2013.] http://www.enterate.unam.mx/Articulos/2006/febrero/arquitec.htm. 46.

19. Instituo gallego de promoción económica. Procedimiento general. Elaboración de un mapa de procesos. Galicia : s.n., 2009.

20. Pressman, Roger S. Ingeniería de Software, un enfoque práctico. s.l. : McGraw-Hill Companies, 2002.

21. Gamma, Erich, Helm, Richard y Johnson, Ralph. Design Patterns (Elements of Reusable Object-Oriented Software).

22. Ecured, conocimiento con todos y para todos. Métricas de diseño. [En línea] 2013. http://www.ecured.cu/index.php/M%C3%A9trica_de_dise%C3%B1o.

23. CVOSOFT Ingeniería en Sistemas. CVOSOFT Ingeniería en Sistemas. [En línea] Editorial CVOSOFT. httpwww.cvosoft.comsistemas_sap_abaprecursos_tecnicos_abapque-es-sap-mm.php.

24. Fernández Vaca, Juan carlos. Análisis de requisitos, Modelado de Procesos y Configuración del software Openbravo ERP para la gestión de Contabilidad General y Ventas en la empresa P&S. Santa Cruz, Bolivia : s.n., 2010.

25. Espinosa Pompa, Yolegni y Reyes Sosa, Alicia. Propuesta de una metodología para el diagnóstico de los procesos informativos previa instalación/implementación del Software de Gestión Contable Versat Sarasola, en entidades del territorio de Ciudad de La Habana. La Habana : s.n.

26. Desoft, excelencia en software. Desoft, excelencia en software. [En línea] 2013. [Citado el: 12 de 05 de 2013.] http://www.vcl.desoft.cu/smf/index.php?topic=36.0.

Bibliografía

Agipem, una firma de profesionales. Asesoría para la gestión y planificación empresarial. Asesoría para la gestión y planificación empresarial. [En línea] 26 de 10 de 2011. [Citado el: 01 de 01 de 2013.] httpwww.agipem.comindex.phpp=planificacion-economica-financiera.

Saily Oliva Martínez, Yaima Álvarez Márquez, Ing. Lester Antonio González Gutiérrez. Sistema para el control de inventarios del ERP . La Habana : s.n., 2010.

Ing. Tamara Rodríguez Sánchez, Ing. Mairelys Fernández González, Ing. Eliecer Cabrera Casas. La gestión empresarial de las entidades cubanas. Cedrux a la vuelta de la esquina. La Habana : s.n., 2011.

Ing. Virgen Damaris Quevedo Campins, Msc. Meylin Martínez Chong, Ing. Tte Lester Antonio González Gutierrez, Ing. Alien Fernández Fuentes. CEDRUX. Solución para sistemas de control logístico. . La Habana : s.n., 2010.

Manual de usuarios del módulo Inventario de Cedrux. La Habana : s.n., 2010.

Rumbaugh, James, Jacobson, Ivar y Booch, Grady . El Lenguaje Unificado de modelado. 1999.

The Apache Software Foundation. The Apache Software Foundation. The Apache Software Foundation. [En línea] 2012. [Citado el: 05 de 03 de 2013.] http://httpd.apache.org/docs/2.2/es/new_features_2_0.html.

PostgreSQL. The world's most advanced open source data base. PostgreSQL. The world's most advanced open source data base. [En línea] 1996. [Citado el: 05 de 03 de 2013.] http://www.postgresql.org.

Mozilla Europe. Notas de publicación de Firefox 3.6. Notas de publicación de Firefox 3.6. [En línea] 2009. [Citado el: 30 de 01 de 2013.] httpwww.mozilla-europe.orgesfirefox3.6releasenotes.

RapidSVN. RapidSVN. [En línea] http://www.rapidsvn.org/index.php/Main_Page.

NetBeans Community. [En línea] 2013. https://netbeans.org/community/releases/72/.

Guide to Doctrine for PHP. 2010.

Zammetti, Frank W. Practical Ext JS Projects with Gears. New York : Springer-Verlag , 2009.

Allen, Rob. Getting started with ZendFramework. [En línea] 2006, 2010. http://alemohamad.com/tutorial-zend-framework/.

Mariño, Carlos Vázquez. Programación en PHP 5. Nivel Básico. Ferrol : s.n., 2008.

Sánchez, Jorge. JavaScript. Manual de referencia. 2003.

Centro de Informatización de la Gestión de Entidades. Subdirección de Producción. Modelo de Desarrollo de Software. La Habana : s.n., 2012.

Suárez, José de Jesús Hernández. Entérate en línea. Internet,cómputo y telecomunicaciones. Entérate en línea. Internet,cómputo y telecomunicaciones. [En línea] 02 de 2006. [Citado el: 30 de 01 de 2013.] http://www.enterate.unam.mx/Articulos/2006/febrero/arquitec.htm. 46.

Instituo gallego de promoción económica. Procedimiento general. Elaboración de un mapa de procesos. Galicia : s.n., 2009.

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

Gamma, Erich, Helm, Richard y Johnson, Ralph. Design Patterns (Elements of Reusable Object-Oriented Software).

Ecured, conocimiento con todos y para todos. Métricas de diseño. [En línea] 2013. http://www.ecured.cu/index.php/M%C3%A9trica_de_dise%C3%B1o.

CVOSOFT Ingeniería en Sistemas. CVOSOFT Ingeniería en Sistemas. [En línea] Editorial CVOSOFT. httpwww.cvosoft.comsistemas_sap_abaprecursos_tecnicos_abapque-es-sap-mm.php.

Fernández Vaca, Juan carlos. Análisis de requisitos, Modelado de Procesos y Configuración del software Openbravo ERP para la gestión de Contabilidad General y Ventas en la empresa P&S. Santa Cruz, Bolivia : s.n., 2010.

Espinosa Pompa, Yolegni y Reyes Sosa, Alicia. Propuesta de una metodología para el diagnóstico de los procesos informativos previa instalación/implementación del Software de Gestión Contable Versat Sarasola, en entidades del territorio de Ciudad de La Habana. La Habana : s.n.

Desoft, excelencia en software. Desoft, excelencia en software. [En línea] 2013. [Citado el: 12 de 05 de 2013.] http://www.vcl.desoft.cu/smf/index.php?topic=36.0.