Universidad de las Ciencias Informáticas

Centro de Soluciones de Gestión

Título: Implementación de los Componentes Configuración, Nomencladores y Recuperaciones del subsistema Inventario del Sistema Integral de Gestión Cedrux.

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Alién García Hernández

Damián Viltres Ramírez

Tutores: Lic. Arismayda Dorado Risco

Ing. Larisa González Álvarez

Ciudad de la Habana, Junio de 2009

"Año del 50 Aniversario del Triunfo de la Revolución"

DATOS DE CONTACTO

Tutor: *Lic. Arismayda Dorado Risco*

Correo electrónico: dorado@uci.cu

Profesora con 4 años de experiencia. Licenciada en Ciencias de la Computación. Actualmente es Arquitecta de Sistema de los Subsistemas Inventario y Facturación del proyecto ERP-Cuba.

Categoría Científica: Licenciada

Categoría Docente: Instructor

Tutor: *Ing. Larisa González Álvarez*

Correo electrónico: lgalvarez@uci.cu

Profesora recién graduada con Título de Oro en Julio del 2008 de Ingeniería en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Actualmente Arquitecta de Sistema del Subsistema Activos Fijos del proyecto ERP Cuba.

Categoría Científica: Ingeniero

Categoría Docente: Adiestrado

"Si de algo serví antes de ahora, ya no me acuerdo, lo que quiero es servir más"

José Martí

Alién

A mi mamá, por comprenderme y apoyarme siempre en todo, aunque a veces no esté de acuerdo con muchas cosas. Mima: gracias por enseñarme el verdadero significado de una madre.

A mi papá por inculcarme los principios revolucionarios y humanos que rigen mi vida. Por no saber a veces lo que es suyo para darme todo lo que le pida. Por estar orgulloso de mí.

A mi hermano, para que siga mis pasos y se convierta en un intelectual. Para que sepa que es lo que más quiero en el mundo, aunque no se lo diga muy a menudo. Aquí tienes a un hermano, un amigo y un padre.

Damián

A mis padres, no solo por traerme a la vida, sino por ayudarme a transitar por ella, por brindarme sus manos cuando el camino era inseguro, por darme la luz cuando la oscuridad era intensa, a ellos antes que nadie por hacerme el ser humano que hoy soy, a mi mama especialmente porque sin ti este sueño no se hubiera hecho realidad, muchas gracias por todo mami.

A mi novia Yanet Gonzales Pérez, por su amor, dedicación y paciencia. Gracias por tu apoyo y por haber confiado en mí todo este tiempo.

A mi abuelita querida, se que siempre me tuviste presente en tus oraciones pidiendo todo lo mejor para mi.

En fin a toda mi familia, que siempre estuvieron al tanto de todo, apoyándome en todo momento.

Alién

Toda obra humana es la acción concatenada de muchas voluntades, la presente no escapa de esta regla, quisiera agradecer a:

- *La Revolución Cubana, al Comandante Fidel y a la UCI, por convertir en realidad mi sueño.*
- *Mis padres, por sacrificarse toda su vida para convertir a este hijo en Ingeniero.*
- *Mi hermano, por quererme siempre aunque sea en silencio.*
- *Aya, por ser mi madre, mi abuela, y parte fundamental de mi vida.*
- *Mi tía Iliana, por sentirme siempre como un hijo. A Pedro y a mi primo Pedrito, el mejor de todos los karatecas.*
- *Mis tíos Boly, Carmen y a mi primo Javier, por abrirme las puertas de su casa y de su corazón.*
- *Oriel y Orismel, este título es también de ustedes, estoy totalmente orgulloso de mis primos. A mi tía Maida, a la memoria de mi tío Orestes y mi abuela Gudelia . A Aleida, por apoyarme durante estos cinco años en mis viajes constantes. A Misle y a mi abuelo Justo. A toda mi familia y a mis vecinos, que son también familia.*
- *Mis hermanos más que amigos Omar, Chirino, Olivier.*
- *Alisney (Sua), por saber aceptarme y quererme como la hermana que nunca tuve. A Annia, mi amiga desde ese primer día de aula universitaria. A Leidys (la hermana) y Sahilyn por brindarme su amistad incondicional. A Wendy, Yarleny y Marlié, por ser confidentes en todo momento. Pedro Diosmel, para que nunca deje de priorizar la profesionalidad, a Jacqueline, Leandro Miguel. A todas mis amistades de la UCI, es imposible ponerlas en un espacio tan pequeño. Mis amigos de la Vocacional: Dayana Víctores, Yaima, Daily, Ivette, Yuniel, Brisey, Alexey, Frank, El Yerro, El King, Yanet, Yuliet, Yaneisy, Luana, Orlando y todos los demás.*
- *La FEU, por educarme políticamente.*

- *A mi profe de teatro César. A Velázquez y Mairela por enseñarme que el arte dignifica.*
- *A mis profes de la Facultad 1 por siempre apoyarme en todo, especialmente a Nadia, Cao, Matilde, Joel, Yaliana, Dailiany e Isel.*
- *A las tutoras Arismayda y Larisa, así como a Yaima Alvarez por apoyarnos y defendernos siempre.*
- *A todo aquel que confío en mí y me apoyó en algún momento: Gracias.*

Damián

Agradezco el cumplimiento de este sueño a:

- *La revolución, a nuestro comandante Fidel y a la UCI por permitirnos formar parte del proyecto futuro y por formarnos como profesionales revolucionarios.*
- *Nuestras tutoras, Arismayda, Larisa y a Yaima Alvarez Marquez por todo el apoyo brindado en la realización de este trabajo.*
- *Mis amigos Reinier Blanco y Jose Antonio más que amigos, hermanos, por brindarme sus manos en cada momento que las necesité.*
- *Todos mis amigos y amigas que compartieron buenos y malos momentos en estos cinco años.*
- *Todas aquellas personas que de una forma u otra me brindaron su apoyo e hicieron posible estar aquí en estos momentos.*

RESUMEN

Actualmente la situación general de los procesos de gestión de las entidades a escala nacional, está afectada por el control de los datos de forma manual y por la existencia de sistemas informáticos que no explotan las posibilidades que brindan las nuevas tecnologías. Entre estos procesos empresariales se encuentra la Gestión de Almacenes, la cual posibilita el control de los recursos tangibles y el manejo de información fidedigna a la hora de tomar decisiones vitales por los directivos de cada entidad.

El presente trabajo abarca la implementación de los componentes de Configuración, Nomencladores y Recuperaciones del Subsistema de Inventario, lo cual es un paso importante para lograr la eficiencia en la gestión de los procesos inventariales. Cuenta con un estudio del arte enmarcado en los sistemas vinculados al campo de acción. Se realiza una descripción de las herramientas y tecnologías utilizadas, dando paso a la propuesta de solución. En dicha propuesta se muestran varias de las clases más importantes, así como el diagrama de componentes. Se logra medir la calidad de la solución aplicando pruebas de caja blanca y métricas para validar la implementación de las clases.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

La información es uno de los activos más importantes para empresas y organizaciones; la misma va en creciente y constante aumento, y al desarrollarse la tecnología se ha logrado almacenar en diferentes formatos, independientemente del cual, es necesario que se encuentre a disposición de los usuarios; siendo importante crear aplicaciones de software cuyo objetivo fundamental sea realizar recuperaciones de datos, lográndose así un acceso rápido y objetivo a la información almacenada.

Los sistemas informáticos se especializan en negocios determinados. En la actualidad se estila desarrollar un paquete de software empresarial que integra todas las áreas de la organización vinculadas a lograr su objetivo: la generación de productos y servicios. Dicho paquete es conocido como sistema de Planeación de Recursos Empresariales (*ERP, del inglés)* y garantiza la centralización de la información de una empresa.

Implementar un sistema ERP es un proceso largo, costoso, complejo y que requiere de gran cantidad de desarrolladores. Es por ello que es más factible dividir su desarrollo en módulos que representan las distintas áreas de la empresa, de forma que se facilite su proceso de construcción. Uno de estos subsistemas abarca la Gestión de Inventarios, el cual permite gestionar los medios que posee una organización y se encarga de evaluar los procedimientos de entrada y salida de sus bienes, es decir, controla su existencia con el fin de hacer más rentable su posesión y garantizar en cierto grado el camino hacia el éxito.

Hay que destacar que la gestión de almacenes constituye una tarea importante en cualquier empresa. Un adecuado control de los mismos permite a cualquier entidad minimizar el espacio físico para mejorar el aprovechamiento y aumentar las rotaciones de cada material en existencia, utilizando menos recursos financieros en inversiones no productivas (locales y materiales), facilitando de ese modo una mayor rentabilidad económica.

El encargado de gestionar los inventarios a los almacenes de la empresa es uno de los actores más importantes porque permite, entre otras funcionalidades, el control de los recursos tangibles, así como gestionar información fidedigna que favorecerá la toma de decisiones por parte de los directivos, contribuyendo de esa manera al mejor funcionamiento y manejo de los recursos, en aras del desarrollo de la empresa.

La Oficina Nacional de Informatización ha reconocido que en Cuba la situación general de las entidades de producción de aplicaciones informáticas y de las propias aplicaciones en explotación que viabilizan el Control de Inventarios en el entorno empresarial cubano, está caracterizada por:

- Una considerable presencia de sistemas informáticos desarrollados sobre plataformas envejecidas y con poco o ningún criterio de seguridad y auditoría (en el orden técnico y funcional).

- Productos que abordan solamente partes del problema de la gestión de la empresa o la unidad presupuestada y no soportan mecanismos estándares de integración con otras aplicaciones.

- Haber sido desarrollados para un ambiente de escritorio, casi ninguno bajo conceptos de informática multicapa y distribuida en la red. Lo más general son desarrollos sobre arquitectura Cliente-Servidor de base de datos.

- Desarrollar aplicaciones utilizando software libre y estándares abiertos en muy pocas entidades del país, por lo cual hay casi un desconocimiento generalizado sobre estas actuales plataformas de desarrollo.

Si vamos a hablar de los sistemas más potentes que se encuentran en explotación en nuestro país, vale destacar que la mayoría son extranjeros y no abarcan todas las operaciones de gestión por una incompleta implementación en la entidad o porque no lo soportan. Generalmente estos productos están enfocados a un sector específico o se han desarrollado para funcionar según las características de otra economía, como la europea. Además de las limitaciones que constituye el elevado precio de los mismos en el mercado internacional (más aún para un país como el nuestro), está la dependencia que se crea a

un suministrador externo, en quienes se seguiría invirtiendo hasta lograr adaptar el Sistema a las necesidades de la economía cubana.

Las deficiencias planteadas justifican la creación de una nueva solución informática. La tarea es designada por el Ministerio de Finanzas y Precios y asumida por la Universidad de las Ciencias Informáticas. El nuevo sistema de Gestión Integral Cedrux posee varios módulos entre ellos el Control de Inventario, dentro del que se encuentran los procesos de Configuración y Recuperaciones.

Analizando lo expuesto anteriormente surge como **problema científico** el siguiente: *¿Cómo obtener un producto funcional a partir de los requerimientos identificados para la gestión de los procesos de Configuración y Recuperaciones pertenecientes al Subsistema Inventario del Sistema Integral de Gestión Cedrux?*

Teniendo en cuenta el problema planteado se define como **objeto de estudio**: Los *procesos para la gestión de inventarios*

Por lo que se especifica el siguiente **campo de acción**: *Procesos de configuración y Recuperaciones en el país.*

Dadas estas condiciones se plantea lograr como **objetivo general:** *Implementar los componentes de Configuración, Nomencladores y Recuperaciones, pertenecientes al subsistema de Inventario del Sistema Integral de Gestión Cedrux*[1]*.*

Se plantean además como **objetivos específicos**:

- Analizar los procesos de Configuración y Recuperaciones, y los sistemas que existen actualmente para su gestión, así como las herramientas que se utilizarán para el desarrollo de la solución.
- Implementar los componentes Configuración, Nomencladores y Recuperaciones integrado al subsistema Inventario del Sistema Integral de Gestión Cedrux.
- Validar la solución propuesta.

[1] **Cedrux:** Vocablo formado por la unión de las palabras “Cedro” (fortaleza, resistencia) y “Linux” (tecnología libre).

Para lograr dichos objetivos se plantearon las siguientes ***tareas***:

- Análisis de los procesos de Configuración y Recuperaciones.
- Análisis de los sistemas existentes para el control de inventarios.
- Análisis de las tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
- Análisis de los artefactos entregados por los analistas para los procesos de Configuración y Recuperaciones.
- Implementación de las interfaces a partir del prototipo entregado por los analistas.
- Implementación de la capa de negocio que de respuesta a los requisitos propuestos por los analistas.
- Implementación de la capa de acceso a datos.
- Implementación de las validaciones, excepciones.
- Implementación de los servicios incluidos dentro de sus responsabilidades que se necesiten para la implementación de otros módulos.
- Realización de pruebas de unidad a los componentes obtenidos.
- Aplicación de métricas para validar los resultados del producto obtenido.

La investigación se basa en la siguiente **idea a defender**: *Si se realiza la implementación de los componentes de Configuración, Nomencladores y Recuperaciones, pertenecientes al subsistema Inventario del Sistema Integral de Gestión Cedrux, se obtendrá un producto funcional que gestione dichos procesos.*

Los siguientes **métodos teóricos** sustentan la investigación:
Histórico-Lógico: Su empleo permitió el desarrollo evolutivo y coherente en el estudio de las métricas de implementación, patrones de diseño, herramientas CASE[2] y sistemas ERP para el desarrollo de los artefactos que proponen los flujos estudiados.

[2] **CASE:** Computer Aided Software Engineering (Ingeniería de Software Asistida por Ordenador)

Analítico-Sintético: Permitió integrar y descomponer el conocimiento, descubriendo las relaciones para la utilización de artefactos propuestos, determinando los aspectos esenciales y arribando a conclusiones prácticas y teóricas.

Inductivo-deductivo: Permitió pasar de lo particular a lo general y viceversa favoreciendo objetivamente el enlace que se establece en la realidad entre lo singular y lo general ya que ambas se complementan mutuamente en el proceso de desarrollo científico.

Modelación: Pues se crean abstracciones que explican la realidad, por ejemplo, todos los modelos y diagramas presentados.

Técnicas empleadas:

Se emplean técnicas para comprender los procesos de negocio, para comprender sus necesidades a automatizar. Estas son la revisión de documentos y lluvia de ideas.

El **aporte práctico** esperado del trabajo es: *La obtención de los componentes de Configuración y Recuperaciones pertenecientes al subsistema de Inventario.*

El documento se divide en 3 capítulos:

Capítulo 1. Fundamentación Teórica:

Se expone el estado del arte, mostrando los sistemas vinculados al campo de acción. Al mismo tiempo se describe el objeto de estudio. También se detallan tendencias y tecnologías actuales utilizadas para el desarrollo del subsistema y el motivo de su utilización. Se realiza una caracterización del modelo de desarrollo aplicado, así como de las herramientas utilizadas para desarrollar los artefactos.

Capítulo 2. Descripción y análisis de la solución propuesta:

Se realiza un análisis de los componentes existentes a utilizar. Se describen algunas de las clases que fueron definidas y como se integran las mismas, realizando además la complejidad de algoritmos. Se describen los procesos a automatizar. Se detalla la propuesta de solución.

Capítulo 3. Validación de la solución propuesta:

Se valida la solución propuesta a través de la realización de pruebas. Este capítulo se encuentra dividido en dos partes: la explicación detallada de las pruebas de caja blanca que fueron realizadas al código del software y de las métricas utilizadas en la implementación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 INTRODUCCIÓN

Es indudable el ambiente competitivo del mundo empresarial actual, las entidades requieren promover procesos y actividades de negocio que generen ventajas ante sus más fuertes competidores o simplemente que tributen al ahorro y la eficiencia económica. Hoy más que nunca se requieren herramientas que proporcionen control y centralización de la información, con el fin de tomar las mejores decisiones para los procesos y estrategias de negocios.

No solo en Cuba, sino en el mundo, se han desarrollado aplicaciones relacionadas con el Control de Inventarios, cuyo estudio ha servido para tener en cuenta las ventajas y defectos que tienen. A menudo surgen dudas sobre cuál usar, debido a que frecuentemente aparecen nuevas herramientas con mejorías, que se van imponiendo para el desarrollo de software. Es por esto, que surge la necesidad de hacer un estudio de las herramientas a utilizar, buscando obtener la información más actual de las mismas.

1.2 LA GESTIÓN DE INVENTARIOS EN LA PLANEACIÓN DE RECURSOS EMPRESARIALES

Se puede definir por Enterprise Resource Planning (ERP) a los sistemas de planificación de recursos empresariales que integran y manejan muchas de las prácticas de los negocios asociados con las operaciones de producción y los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

Los ERP son parte del conjunto de sistemas de información gerencial que permiten tener un control de la empresa por sus directivos en tiempo real. Maneja los datos y la información de la empresa, controlándolos, centralizándolos y compartiéndolos entre los

diferentes módulos, de manera que evita que los datos se corrompan, dañen o dupliquen *(González Rodríguez, 2003).*

Uno de los módulos gerenciales que existen comúnmente en las empresas y que es integrable a un ERP lo es el referente a los procesos del Control de Inventarios.

Este módulo revierte gran importancia en el manejo de los productos en almacenes controla los medios que posee una organización. Se encarga de inspeccionar las existencias de productos, dándole entrada y salida del almacén a los mismos, regulando el flujo de mercancía en almacenamiento, con el fin de hacer más rentable su posesión y garantizar en cierto grado el éxito de la organización *(Torres Saquipova, y otros, 2008).*

El Control de Inventario involucra entre otros los procesos de Recepción, Apertura, Despacho y Baja. Tributa además a los módulos de Costos y Procesos, Contabilidad Financiera y Capital Humano.

1.3 SISTEMAS EXISTENTES

A continuación se realiza un análisis de dos sistemas nacionales y dos extranjeros vinculados a la gestión empresarial.

Versat Sarasola

El programa Versat-Sarasola fue el primer sistema de contabilidad cubano certificado, en cuya evaluación participaron el Ministerio de Finanzas y Precios, consultorías internacionales y otros organismos. Es una aplicación de escritorio implementada en Delphi. Funciona sobre el sistema operativo Windows y presenta soporte para bases de dato SQL Server 2000.

Está constituido por 12 módulos, entre los que se encuentra el control de inventarios *(del Toro Ríos, y otros, 2008).* En este sentido, registra 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 *(Desoft, 2008).*

Rodas XXI. Versión 3.0

Sistema multiempresa y multiusuario creado por CITMATEL[3] para automatizar la gestión empresarial. Contiene diferentes módulos que pueden usarse integrados o independiente.

El módulo de Inventario de Rodas XXI le permite tener un control detallado de los inventarios de su entidad, realizando en el mismo momento que se registra un movimiento, su contabilización. Se pueden realizar todo tipo de operaciones de entradas y salidas de los almacenes con facilidad en el momento que se desee, generando el documento asociado al movimiento de que se trate de forma automática previa configuración del sistema para ello. Es posible operar con varios almacenes trabajando cada uno de forma independiente.

Este módulo le permitirá además visualizar información correspondiente a períodos anteriores, tan sólo con cambiar de período contable a otros anteriores ya cerrados, aunque en dichos períodos no podrá realizar ninguna operación. Esta característica es compartida por todos los módulos de RODAS XXI *(Rodas XXI, 2009).*

SAP

SAP ERP pone a su disposición gran cantidad de funcionalidades que le permitirán analizar su negocio, optimizar sus finanzas, gestionar sus recursos humanos, operaciones y servicios corporativos. Además también le proporcionan soporte para cuestiones referentes a la gestión de sistemas de administración de usuarios, gestión de configuración, gestión de datos centralizada y gestión de servicios de Web. Es un sistema costoso, que brinda además las posibilidades de mantenimiento.

[3] **CITMATEL:** Empresa de Tecnologías de la Información y Servicios Telemáticos Avanzados.

El subsistema Inventario de SAP permite reducir los costes de almacenamiento, transporte, cumplimiento de pedidos y manipulación de materiales y, a la vez, mejorar el servicio al cliente. Puede mejorar significativamente la rotación de inventario, optimizar el flujo de mercancías y acortar las rutas en su almacén o centro de distribución. Entre los beneficios adicionales de la gestión de inventarios se encuentran la mejora del flujo de caja, la visibilidad y la toma de decisiones.

Para la gestión de almacenes, puede realizar un seguimiento de la cantidad y el valor de todos sus materiales, realizar un inventario físico y optimizar los recursos del almacén. Los empleados pueden planificar, introducir y documentar movimientos de almacén interno gestionando las entradas y las salidas de mercancías, el almacenamiento, la recogida, el embalaje y los traslados físicos *(SAP España, 2007).*

CONDOR Enterprise

Sistema automatizado de alta complejidad y seguridad que abarca todos los aspectos del proceso contable de una entidad, tales como la dualidad de moneda y el pago por resultados.

Este brinda mayor autonomía al cliente para efectuar cambios de estructura sin necesidad de la intervención de especialistas, quedando registrados de forma que puedan ser auditables. Incluye la contabilidad multimoneda *(del Toro Ríos, y otros, 2008).*

Entre sus módulos se encuentra el de Gestión de Invetarios, el cual permite entre otras funcionalidades:

- Control de los esquemas de depósitos y almacenes.
- Realizar inventarios rotativos.
- Serializar los productos.
- Generar transacciones de costeo *(Condor, 2005).*

Valoración crítica de los sistemas estudiados

A partir del estudio de varios sistemas de gestión de entidades, tanto nacionales como extranjeras, que dentro de sus funcionalidades tiene incluida el control de Inventarios, se concluyó que las soluciones internacionales (SAP y Condor) usan tecnologías que pueden ser riesgosas en la situación actual de bloqueo que sufre nuestro país y es que están basados en la plataforma J2EE[4], cuya máquina virtual es propiedad de la empresa norteamericana SUN[5], estando bajo las leyes de su gobierno, que bloquea por todos los medios el acceso a la tecnología informática. El pago de licencias es un aspecto a tener cuenta y no es favorable para el país debido a las condiciones económicas actuales. Además estos sistemas están basados en economías capitalistas, y no poseen las características propias de nuestro sistema.

Las soluciones nacionales (Versat Sarasola y Rodas XXI) están desarrolladas sobre plataformas donde los programas son propietarios cuando el país está en un proceso de migración hacia Linux, además que son aplicaciones de escritorio, por lo que es necesario para su uso, la instalación de la misma en todas las PC de la entidad. En cambio, con una aplicación web solo bastaría tener acceso a la red para acceder al sistema.

Los sistemas SAP, Condor y Versat Sarasola manejan reportes de Inventario, pero lo realizan de una manera estática, cuando lo ideal es darle la posibilidad al usuario de filtrar datos escogidos según las características deseadas en la Recuperación. Por otro lado el sistema Rodas XXI no implementa reportes.

1.4 PROCESO DE DESARROLLO DE SOFTWARE

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas. Aunque un proyecto de desarrollo de software es equiparable en muchos aspectos a cualquier otro proyecto de ingeniería, en el desarrollo de software hay una serie de

[4] **J2EE:** Java Platform, Enterprise Edition o Java EE es una plataforma de programación para desarrollar y ejecutar aplicaciones en lenguaje de programación Java.

[5] **SUN:** Stanford University Network, Red de la Universidad de Stanford.

desafíos adicionales, relativos esencialmente a la naturaleza del producto obtenido *(Autores, 2007).*

Modelo de Desarrollo *(Gestión, 2008)*

Para el desarrollo de la aplicación se utilizó un modelo de desarrollo orientado a componentes, definido por la subdirección de Arquitectura del Centro de Soluciones de Gestión. Este modelo cuenta con los ***Roles*** mostrados en la Figura 1:

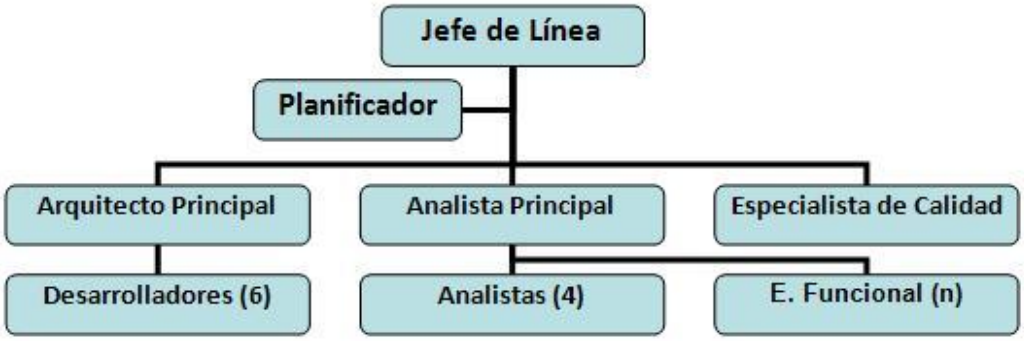


Figura 1. Estructura de las Líneas de Desarrollo

***Flujos de Trabajo/Artefactos*:**

Inicio

- Plan de Iteración (Jefe de la Línea y Planificador)
- Plan de Trabajo Individual (Lo elaboran Todos) (Lo controla el planificador y lo evalúa el Jefe de la Línea)

Análisis

- Arquitectura de Negocio
  - Mapa de Proceso (Analista Principal)
  - Descripción de Procesos de Negocio (Analista)
- Especificación de Requisitos (Analista)
- Casos de Prueba (Especialistas de Calidad, Analistas y Especialistas Funcionales)

- Arquitectura del Sistema (Arquitecto de Sistema)

Diseño e Implementación

- Modelo Conceptual (Arquitecto de Sistema) (Desarrolladores, Analistas)
- Modelo de Datos (Desarrollador)
- Diseño de Clases (Desarrollador)
- Diseño de Interfaz de Usuario (Desarrollador)
- Release (Arquitecto de Sistema)

Prueba

- Casos de Prueba (Equipo de Pruebas Central)
- Registro de no Conformidades (Especialista de Calidad responsable) (Equipo de Pruebas Central registra las No Conformidades)

Estabilización e Integración

- Historia de la Iteración (Jefe de la Línea)
- Informe de Integración (Arquitecto de Sistema)

Mantenimiento

- Peticiones de Actualizaciones (En teoría es el departamento de Implantación).
- Registro de errores (En teoría es el departamento de Implantación).

1.5 HERRAMIENTAS, LENGUAJES Y TECNOLOGÍAS UTILIZADAS

Las herramientas y tecnologías que se describen a continuación en el capítulo, son las utilizadas en la implementación de los componentes. Fue una decisión determinada por el equipo de arquitectura del Centro de Soluciones de Gestión.

1.5.1 Herramientas

Actualmente se consideran a las Herramientas de Desarrollo de Software (HDS) como herramientas basadas en computadoras que asisten el proceso de ciclo de vida de una aplicación, consolidadas en la literatura en la forma de Ingeniería de software asistida por computadora (CASE, por sus siglas en inglés). Las HDS automatizan metodologías de software y desarrollo de sistemas *(Rivas, y otros, 2007).*

Visual Paradigm 6.3

Visual Paradigm para UML es una herramienta ampliamente utilizada en el mundo del software que permite a los profesionales modelar sus diseños. Posibilita la integración de aplicaciones empresariales a las bases de datos. Es capaz de generar código e ingeniería inversa para lenguajes como el PHP. Permite manejar grandes estructuras de manera eficiente, solo requiere una configuración de escritorio común. Soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientado a objetos, construcción, pruebas y despliegues *(Visual Paradigm, 2006).*

Zend Studio para Eclipse 6.0.

Zend Studio para Eclipse posee las capacidades para crear poderosas herramientas web. Permite a los desarrolladores crear estrategias de integración más eficientes. Al proporcionar potentes capacidades de acción con el lenguaje PHP, mejoras de soporte con JavaScript y profunda integración a Zend Framework, el desarrollo de aplicaciones se realiza en un tiempo récord. Incluye un poderoso editor de código con soporte para todos los formatos web. Posee un fuerte manejo de la arquitectura Cliente/Servidor, excelente depuración, elaboración de perfiles y un código de cobertura. Zend estudio tiene todas las herramientas que un desarrollador necesita para asegurarse de que el código es correcto y empezar a diagnosticar problemas con antelación. Tiene características de depuración avanzadas, incluyendo: condiciones límites, visualización de errores, variables y buffer de salida. Asegura la protección máxima de ubicaciones de proyectos o en Internet con depuradores remotos *(Zend, 2007).*

pgAdmin III

pgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo una de las más completa y popular con licencia de código abierto. Está escrita en C++ y permite que se pueda usar en Linux, Windows y otros sistemas operativos. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma.

Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis así como un editor de código de la parte del servidor *(Ubunto, 2008).*

Postgre SQL Manager 2007

Es una aplicación de alto desempeño para la administración y desarrollo de PostgreSQL Database Server. El programa trabaja con cualquier versión de PostgreSQL y soporta todas las últimas características de PostgreSQL, incluyendo espacios de tablas, nombres de argumentos en funciones y más. Su interfaz gráfica es sumamente atractiva e incluye un modo guiado de trabajo. Incluye las herramientas básicas de mantenimiento y administración.

Características:

- Soporte completo para PostgreSQL hasta la versión 8.3
- Administración y navegación rápida de bases de datos.
- Administración fácil de todos los objetos PostgreSQL.
- Administración efectiva de seguridad.

- Capacidades de exportación e importación de datos.
- Modo guiado para labores de mantenimiento *(SQLManager, 2006).*

TortoiseSVN 1.6.2

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones *Subversion*. Maneja ficheros y directorios a lo largo del tiempo. Los ficheros se almacenan en un *repositorio* central. El repositorio es prácticamente lo mismo que un servidor de ficheros ordinario, salvo que recuerda todos los cambios que se hayan hecho a sus ficheros y directorios. Esto permite que pueda recuperar versiones antiguas de sus ficheros y examinar la historia de cuándo y cómo cambiaron sus datos, y quién hizo el cambio. Esta es la razón por la que mucha gente piensa que Subversion, y los sistemas de control de versiones en general, son una especie de "máquinas del tiempo".
Algunos sistemas de control de versiones también son sistemas de manejo de configuración del software (SCM). Estos sistemas están diseñados específicamente para manejar árboles de código fuente, y tienen muchas características que son específicas para el desarrollo de software - tales como el entendimiento nativo de los lenguajes de programación, o proporcionan herramientas para compilar software. Subversion, sin embargo, no es uno de estos sistemas; es un sistema general que puede ser utilizado para manejar *cualquier* colección de ficheros, incluyendo código fuente *(Küng, y otros, 2009).*

1.5.2 Lenguajes

Lenguaje Unificado de Modelado (Unified Modeling Language, UML)

UML es ante todo un lenguaje. Proporciona un vocabulario y reglas para permitir una comunicación. Este lenguaje se centra en la representación gráfica de un sistema y nos indica cómo crear y leer los modelos, pero no dice cómo crearlos. Esto último es el objetivo de las metodologías de desarrollo.

Los objetivos de UML son muchos, pero se pueden sintetizar sus funciones:

- Visualizar: UML permite expresar gráficamente un sistema de forma que otro lo puede entender.
- Especificar: UML permite especificar cuáles son las características de un sistema antes de su construcción.
- Construir: A partir de los modelos especificados se pueden construir los sistemas diseñados.
- Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión.

Aunque UML está pensado para modelar sistemas complejos con gran cantidad de software, el lenguaje es los suficientemente expresivo como para modelar sistemas que no son informáticos, como flujos de trabajo (*workflow*) en una empresa, diseño de la estructura de una organización y por supuesto, en el diseño de hardware *(Hernández Orallo, 2002).*

HTML[6]

El lenguaje HTML fue diseñado para estructurar documentos y presentarlos en forma de hipertexto, estableciendo relaciones unidireccionales entre ellos (hipervínculos). Inicialmente fue concebido para visualizar e interconectar el contenido de documentos electrónicos, considerando por ello un pequeño conjunto de etiquetas. Posteriormente, la ventaja que representaba la simplicidad de HTML se convirtió en un inconveniente, ya que su marcado no siempre cubría todos los aspectos de presentación que los usuarios requerían. La solución adoptada fue el desarrollo de extensiones privadas del lenguaje, lo que complicó su estandarización. Las luchas comerciales entre las principales empresas de desarrollo de navegadores web condujeron a un lenguaje HTML que, aunque universalmente utilizado, carece de estandarización real. Desde finales de 1993 existen comités de estandarización, impulsados por el World Wide Web Consortium20, que tienden a un modelo único de HTML *(Fresno, 2006).*

[6] **HTML:** HyperText Markup Language (Lenguaje de Marcas de Hipertexto)

JavaScript

JavaScript es un lenguaje de programación que ha permitido el desarrollo de la Web, ha sido el avance más significativo en el logro de páginas Web dinámicas y exactas en cuanto a posición y presentación de su contenido, es un lenguaje robusto y a la vez ligero, el cual a pesar de ser considerado por muchos como un lenguaje no orientado a objetos permite implementar varias de las características de este paradigma de programación, basado en prototipos, las nuevas clases se generan clonando las clases bases y extendiendo su funcionalidad, cualquier navegador puede interpretar el código JavaScript dentro de las páginas Web, permite la máxima interactividad entre el usuario y la página, además la verificación de los datos introducidos por el usuario antes de enviar el formulario al servidor *(Castro Mecías, y otros, 2008).*

PHP

PHP es un lenguaje de programación interpretado usado normalmente para la creación de páginas Web dinámicas. PHP es un acrónimo recursivo que significa "PHP Hypertext Pre-processor". Es software libre. Se puede obtener en la Web y su código esta disponible bajo la licencia GPL

- Sencillo de aprender.
- Similar en sintaxis a C y a PERL
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- El análisis léxico para recoger las variables que se pasan en la dirección lo hace PHP de forma automática. Librándose el usuario de tener que separar las variables y sus valores.
- Se puede incrustar código PHP con etiquetas HTML.
- Excelente soporte de acceso a base de datos.
- Se puede hacer de todo lo que se pueda transmitir por vía HTTP.
- Biblioteca nativa de funciones sumamente amplia e incluida
- No requiere definición de tipos de variables.
- Tiene manejo de excepciones.
- PHP no soporta directamente punteros, como el C, de forma que no existen los problemas de depuración provocados por estos.

- Viene acompañado por una excelente biblioteca de funciones que permite realizar cualquier labor (acceso a base de datos, encriptación, envió de correo, gestión de un e-commerce, XML, creación de PDF, etc)
- Al poderse encapsular dentro de código HTML se puede recoger el trabajo del diseñador gráfico e incrustar el código php posteriormente. Ver figura 2.
- Es multiplataforma, funciona en todas las plataformas que soporten apache *(Pérez Armas, y otros, 2008).*

Figura 2. Esquema de representación del funcionamiento del PHP

PostgreSQL 8.2

PostgreSQL es una fuente poderosa, abierta al sistema de la base de datos correlativo. Tiene más de 15 años de desarrollo activo y una arquitectura probada, han ganado una reputación fuerte para la fiabilidad, integridad de los datos, y exactitud. Corre en todos los sistemas operativos incluídos Linux y Windows. También apoya el almacenamiento de grandes objetos binarios, incluso imágenes, sonidos, o video. Tiene interfaces nativas de la programación para C/C++, Java.net, Perl, la Pitón, Ruby, entre otros, y una documentación excepcional.

Entre sus principales características pueden encontrarse:

- Disponible totalmente sin coste alguno.
- Disponible para los Sistemas Operativos UNIX y Windows.
- Soporte total del Modelo Relacional de Bases de datos.

- Extensiones propias a SQL para realizar consultas sobre la base de datos.
- Dependencias entre objetos, integridad referencial.
- Soporta valores no atómicos como dominio de un campo.
- Estructuras de datos de tipo Array para el dominio de un atributo.
- Definición de Tipos de datos y dominios propios del usuario.
- Columnas del sistema para las tablas definidas por el usuario.
- Definición de esquemas dentro de una base de datos.
- Crea índices de diferentes tipos .
- Herencia de Tablas.
- Tipos de datos y operaciones geométricas.
- Da soporte para funciones y operadores definidos por el usuario.
- Soporte para utilizar tipos compuestos (ROW).
- Asignación de privilegios para los diferentes objetos (base de datos, tabla, vista, función, esquema).
- Definición de disparadores y reglas para un objeto de la base de datos.
- Incorpora tipos para la identificación de objetos, OID.
- Existencia de funciones y operadores incorporados al sistema para trabajar con los diferentes tipos de datos soportados.
- Expresiones condicionales dentro de sentencias SELECT.
- Control de concurrencia basado en un modelo de multiversión (Multiversion Concurrency Control, MVCC)
- Protección por medio de usuarios y grupos de usuarios asignando privilegios sobre los objetos de un clúster de bases de datos.
- Existencia de funciones de administración de la base de datos. (Backup, Restore, Monitoreo).
- Facilidades para trabajar con objetos largos *(Gómez Oduardo, y otros, 2008).*

1.5.3 Tecnologías

La tecnología a utilizar en la implementación es vital para minimizar el esfuerzo del desarrollador, los marcos de trabajo y el uso de varias técnicas novedosas de

programación viabilizan el proceso de desarrollo de software. El uso de una arquitectura adecuada favorece a la robustez del sistema y a la futura reutilización del código.

Marcos de Trabajo

Los Frameworks (Marcos de Trabajo), no son más que arquitecturas definidas para un determinado dominio de la aplicación que contiene un conjunto de componentes implementados y sus interfaces bien definidas, estos componentes se pueden utilizar, redefinir y crear nuevos componentes *(Perera Morales, 2007).*

Zend Framework 1.5

Es un framework para desarrollo de aplicaciones Web y servicios Web con PHP. Brinda soluciones para construir sitios web modernos, robustos y seguros. Además es código abierto y trabaja con PHP 5.

Presenta entre otras las siguientes características:

- Simplifica la gestión de archivos de configuración.
- Proporciona los componentes que forman la infraestructura del patrón Modelo-Vista-Controlador.
- Proporciona una capa de acceso a base de datos, construida sobre PDO[7] pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos *(Zend, 2007).*

ExtJs 2.1

La programación con la librería ExtJS, que extiende la librería YUI e integra AJAX, Prototype y Scriptaculous. Con el tiempo se convirtió en un framework independiente y a principio de 2007 se creó una compañía para comercializar y dar soporte al ExtJS.

ExtJS es neutral al lenguaje que se use en el servidor. Siempre que el resultado se envíe a la página en el formato adecuado, ExtJS no se preocupará de lo que pase en el servidor. Hay docenas de widgets a escoger en ExtJS, incluyendo composiciones

[7] **PDO:** PHP Data Objects (capa de abstracción de acceso a datos para PHP)

automáticas de páginas, pestañas, menús, barras de herramientas, diálogos, vistas en árbol. Proporciona un selector de nodos DOM extremadamente poderoso llamado DomQuery (puede usarse como una librería independiente, pero en el contexto de ExtJS se usará para seleccionar elementos para poder interactuar con ellos a través de la interfaz Element, contiene mucho de los métodos y propiedades de DOM que se necesitará proporcionando una interfaz conveniente, unificada y multinavegador) *(ExtJS, 2007).*

Doctrine Framework

El Framework Doctrine es un potente y completo sistema de mapas de relaciones de objetos (Object Relational Mapper, ORM por sus siglas en inglés) para PHP 5.2+ con una base de datos con capas de abstracción incorporada. Además es multiplataforma.

UCID Framework

Es el Framework encargado del trabajo con la vistas. Abarca la integración de ExtJs Framework con el sistema incluyendo el integrador de interfaz, el generador de interfaz dinámica y la impresión de documentos. Integra la iconografía, los diferentes temas de escritorio de la aplicación y el multilenguaje.

Inversión de Control

La Inversión de Control (IoC, *del inglés*) es una técnica de implementación en la que se invierte el flujo de programación realizando llamadas a funciones. Se especifican respuestas deseadas a sucesos o solicitudes de datos concretas, recibiendo la información necesaria de parte de alguna entidad o arquitectura externa que realiza la acción de control. Es el propio código del usuario quien es invocado por la librería. En este caso la librería implementa estructuras de alto nivel y el código del usuario las tareas de bajo nivel.

Arquitectura 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. Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales o estaciones de trabajo y los servidores en procesadores departamentales o de grupo. Ver figura 3.

Entre las principales características de la arquitectura cliente/servidor 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. *(Rosabal Carrión, y otros, 2006)*

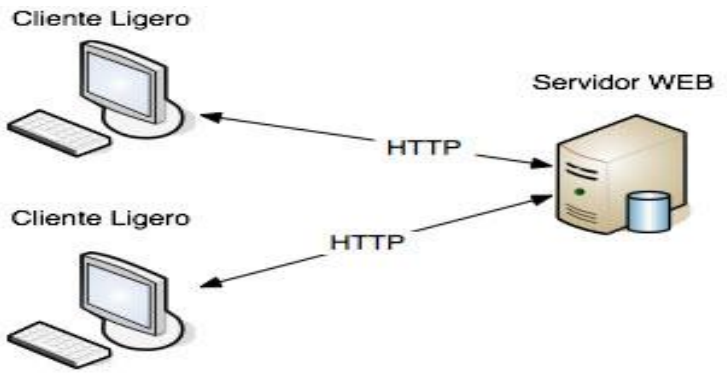


Figura 3. Arquitectura Cliente/Servidor

Servidor Web

Apache HTTP Server 2.0

El servidor Apache se desarrolla dentro del proyecto HTTP Server (httpd) de la Apache Software Foundation. Apache presenta entre otras características mensajes de error altamente configurables, bases de datos de autenticación y negociado de contenido, pero fue criticado por la falta de una interfaz gráfica que ayude en su configuración.
Apache tiene amplia aceptación en la red: en 2005, Apache fue el servidor HTTP más usado, siendo el servidor empleado en el 48% de los sitios web en el mundo. Sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft). La mayoría de las vulnerabilidades de la seguridad descubiertas y resueltas tan sólo pueden ser aprovechadas por usuarios locales y no remotamente. Sin embargo, algunas se pueden accionar remotamente en ciertas situaciones, o explotar por los usuarios locales malévolos en las disposiciones de recibimiento compartidas que utilizan PHP como módulo de Apache.

Ventajas de Apache

- Modular
- Código abierto
- Multiplataforma
- Extensible
- Popular (fácil para conseguir ayuda/soporte)
- Gratuito *(Batista Chillón, y otros, 2008).*

Navegadores

Un navegador, navegador red o navegador web (del inglés, *web browser*) es un programa que permite visualizar la información que contiene una página web (ya esté esta alojada en un servidor dentro de la World Wide Web o en uno local).

El navegador interpreta el código, HTML generalmente, en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares de la red mediante enlaces o hipervínculos.

Mozilla Firefox 2.0.0.17

Mozilla Firefox es un navegador, con interfaz gráfica de usuario desarrollado por la Corporación Mozilla. El programa es multiplataforma y está disponible en versiones para Windows, Linux y otros sistemas operativos. El código fuente de Firefox está disponible libremente bajo la *triple licencia* de Mozilla como un programa libre y de código abierto. Firefox incorpora bloqueo de ventanas emergentes, navegación por pestañas, marcadores dinámicos, compatibilidad con estándares abiertos, y un mecanismo para añadir funciones mediante extensiones *(Montesino Afonso, y otros, 2008).*

1.6 CONCLUSIONES PARCIALES

En este capítulo se conocieron diferentes sistemas nacionales e internacionales que abarcan el control de inventario. Ninguno de ellos realiza recuperaciones dinámicas, elemento que sirvió de análisis al momento de realizar la propuesta de solución. Se sentó el basamento teórico de las herramientas a utilizar, propuestas por el equipo de Arquitectura del Centro de Soluciones de Gestión; teniendo como principales recursos las tecnologías libres, elemento fundamental en las nuevas concepciones de informatización en nuestro país. Están dadas las condiciones para conocer la solución propuesta, luego de introducidos los conceptos fundamentales para su comprensión.

CAPÍTULO II: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1 INTRODUCCIÓN

En este capítulo se delimitan varios puntos importantes en la implementación de los componentes. Se comienza exponiendo los requisitos funcionales detectados por los analistas. Se muestra cómo está concebida la arquitectura y las posibilidades que proporcionan los marcos de trabajo utilizados en la programación de la aplicación, con el objetivo de facilitar la comprensión del funcionamiento de los componentes implementados. También se determina la complejidad a un algoritmo no trivial, se hace una descripción de clases y operaciones utilizadas.

2.2 VALORACIÓN DE LOS ARTEFACTOS PROPUESTOS POR LOS ANALISTAS

En un proceso de desarrollo de software es de gran importancia la descripción de los requisitos funcionales, los que son brindados por el analista de sistema y facilitan un mejor entendimiento de los procesos a desarrollar, permitiendo comprender en profundidad el problema en cuestión y facilitando una mejor identificación de las clases y funcionalidades que serán implementadas.

La especificación de requerimientos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto, debido a que son un reflejo detallado de las necesidades de los clientes o usuarios del sistema. Un requerimiento según el Standard Glossary of Software Engineering Terminology de la IEEE[8] se puede definir como una:

1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.

[8] **IEEE:** Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos)

2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 o 2.

Los requisitos funcionales propuestos por los analistas posibilitaron crear una entrada apropiada y un punto de partida para las actividades de implementación, los mismos fueron los siguientes:

Componente Configuración

RF 1: Gestionar formato.

RF 1.1: Adicionar un formato.

RF 1.2: Modificar un formato.

RF 1.3: Eliminar un formato.

RF 2: Gestionar modificación de cantidad de salida.

RF 2.1: Especificar una modificación de cantidad de salida.

RF 2.2: Eliminar una modificación de cantidad de salida.

RF 3: Gestionar trabajo con comprobante de operaciones.

RF 3.1: Especificar generación de comprobante.

RF 3.2: Modificar generación de comprobante.

RF 4: Confeccionar la ayuda del subcomponente Configuración del almacén.

RF 5: Gestionar áreas.

RF 5.1: Adicionar un área.

RF 5.2: Modificar un área.

RF 5.3: Eliminar un área.

RF 6: Gestionar estructura de las áreas.

RF 6.1: Adicionar una estructura al área.

RF 6.2: Modificar una estructura al área.

RF 6.3: Eliminar una estructura al área.

RF 8: Confeccionar la ayuda del subcomponente Estructura de ubicaciones.

RF 9: Buscar documentos bloqueados.

RF 10: Desbloquear documento.

RF 11: Confeccionar la ayuda del subcomponente Bloqueo de documentos.

Componente Nomencladores

RF 14: Gestionar producto

RF 14.1: Adicionar producto

RF 14.2: Modificar producto

RF 14.3: Eliminar producto

RF 15: Buscar producto

RF 16: Confeccionar la ayuda del nomenclador Producto.

RF 17: Gestionar calidad

RF 17.1: Adicionar calidad

RF 17.2: Modificar calidad

RF 17.3: Eliminar calidad

RF 17.4: Activar calidad

RF 18: Buscar calidad

RF 19: Gestionar categoría

- RF 19.1: Adicionar categoría
- RF 19.2: Modificar categoría
- RF 19.3: Eliminar categoría
- RF 19.4: Activar categoría

RF 20: Buscar categoría

RF 21: Gestionar destino final

- RF 21.1: Adicionar destino final
- RF 21.2: Modificar destino final
- RF 21.3: Eliminar destino final
- RF 21.4: Activar destino final

RF 22: Buscar destino final

RF 23: Gestionar grupo

- RF 23.1: Adicionar grupo
- RF 23.2: Modificar grupo
- RF 23.3: Eliminar grupo
- RF 23.4: Activar grupo

RF 24: Buscar grupo

RF 23: Gestionar concepto

- RF 23.1: Adicionar concepto

RF 23.2: Modificar concepto

RF 23.3: Eliminar concepto

RF 23.4: Activar concepto

RF 24: Buscar concepto

Componente Recuperaciones

RF 27: Imprimir listado de productos en documento

RF 27.1: Buscar documento

RF 27.2: Buscar documento de forma avanzada

RF 27.3: Buscar cliente

RF 27.4: Seleccionar cliente

RF 27.5: Eliminar cliente

RF 27.6: Buscar producto

RF 27.7: Seleccionar producto

RF 27.8: Eliminar producto

RF 27.9: Imprimir listado

RF 28: Limpiar campos de Productos en documento

RF 29: Confeccionar la ayuda del subcomponente Productos en Documento.

RF 30: Mostrar valor del movimiento

RF 30.1: Mostrar rango de fecha

RF 30.2: Buscar movimientos

RF 30.3: Seleccionar depósito

RF 30.4: Mostrar

RF 31: Limpiar campos de Valor del movimiento

RF 32: Confeccionar la ayuda del subcomponente Valor del movimiento.

RF 33: Mostrar productos con déficit/ingreso.

RF 33.1: Seleccionar depósito

RF 33.2: Seleccionar rango de fecha

RF 33.3: Seleccionar déficit/ingreso

RF 33.4: Seleccionar depósito

RF 33.5: Mostrar

RF 34: Gestionar producto

RF 34.1: Adicionar producto

RF 34.2: Modificar producto

RF 34.3: Eliminar producto

RF 35: Limpiar campos de Productos con déficit/ingreso

RF 36: Confeccionar la ayuda del subcomponente Productos con déficit/ingreso.

RF 37: Imprimir modelo de documento pendiente.

RF 37.1: Seleccionar depósito

RF 37.2: Buscar documento

RF 37.3: Imprimir modelo

RR 38: Imprimir listado de documentos pendientes.

RF 39: Mostrar cliente y productos.

RF 39.1: Seleccionar depósito

RF 39.2: Buscar cliente

RF 39.3: Seleccionar cliente

RF 39.4: Eliminar cliente

RF 39.5: Buscar suministrador

RF 39.6: Seleccionar suministrador

RF 39.7: Eliminar suministrador

RF 39.8: Seleccionar rango de fecha

RF 39.9: Escoger estado del producto

RF 39.10: Buscar producto

RF 39.11: Seleccionar producto

RF 39.12: Eliminar producto

RF 39.13: Mostrar

RF 40: Resumen de clientes y productos.

RF 41: Limpiar campos de Clientes y productos

RF 42: Confeccionar la ayuda del subcomponente Clientes y productos.

R43- Mostrar documentos emitidos y recibidos.

RF 43.1: Seleccionar rango de fecha

RF 43.2: Seleccionar tipo de documento

RF 43.3: Buscar cliente

RF 43.4: Seleccionar cliente

RF 43.5: Eliminar cliente

RF 43.6: Buscar suministrador

RF 43.7: Seleccionar suministrador

RF 43.8: Eliminar suministrador

RF 43.9: Buscar transportador

RF 43.10: Seleccionar transportador

RF 43.11: Eliminar transportador

RF 43.12: Escoger depósito

RF 43.13: Mostrar

RF 44: Limpiar campos de Documentos emitidos y recibidos

RF 45: Confeccionar la ayuda del subcomponente Documentos emitidos y recibidos.

2.3 INFORMACIÓN QUE SE MANEJA

Cliente-productos: El sistema le brinda al actor la posibilidad de saber que productos se han recibido o entregado por el o los clientes seleccionados en el componente cliente *(Ver Anexo 1)*.

Documentos emitidos-recibidos: El sistema le brinda al actor la posibilidad de ver un listado de documentos emitidos o recibidos en un rango de fecha seleccionado *(Ver Anexo 2)*.

Documentos pendientes: El sistema le brinda al actor la posibilidad de mostrar los documentos emitidos y que aún no se han cumplimentado *(ver Anexo 3)*.

Productos con déficit-ganancia de presupuesto: El sistema le brinda al actor la posibilidad de ver los productos que tienen déficit o ingreso de presupuesto. Es decir que los saldos

en dinero se encuentran en negativo o que fueron vendidos por encima de su precio promedio que significa que se obtuvieron ganancias.

Valor de los movimientos: El sistema le brinda al actor la posibilidad de ver el valor de las entradas y salidas, ajustes y cantidades involucradas, es decir muestra el valor al que ascienden las entradas y salidas en el punto de contabilidad material.

Movimiento de los productos: El sistema le brinda al actor la posibilidad de ver los movimientos que ha tenido un producto seleccionado, los cuales pueden ser, de salida o entrada en el almacén.

2.4 PROCESOS OBJETO DE AUTOMATIZACIÓN

A continuación se describen los procesos de negocio que son objeto de automatización, con el objetivo de brindar posteriormente la solución propuesta.

Configuración de Almacenes

La configuración de un almacén es unos de los primeros pasos a seguir cuando se desea tener un control eficiente. La ubicación exacta de cada producto en una posición ideal es fundamental a la hora de tomar decisiones urgentes, o simplemente, al momento de inventariar. Este proceso roba un tiempo considerable a las personas designadas, automatizarlo es prioridad en todo sistema que desee mejorar la eficiencia empresarial.

Reportes de Inventario

Al momento de tomar decisiones vitales en una entidad, es importante el control de los recursos; saber a ciencia cierta lo que se posee o no se posee. Para ello, son fundamentales varios reportes de los procesos inventario, que tributen a la dirección un mejor control de los recursos. El conteo físico, el filtrado de datos y la comparación numérica de forma manual se hace extremadamente costosa en el tiempo. Permitir reportes parametrizables constituye un adelanto significativo en la eficiencia económica.

2.5 PROPUESTA DE SOLUCIÓN

Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación *(Pressman, 1998).*

En la Figura 4 se muestra el Digrama de Integración entre Componentes y en la Figura 5 el Diagrama de Componentes.

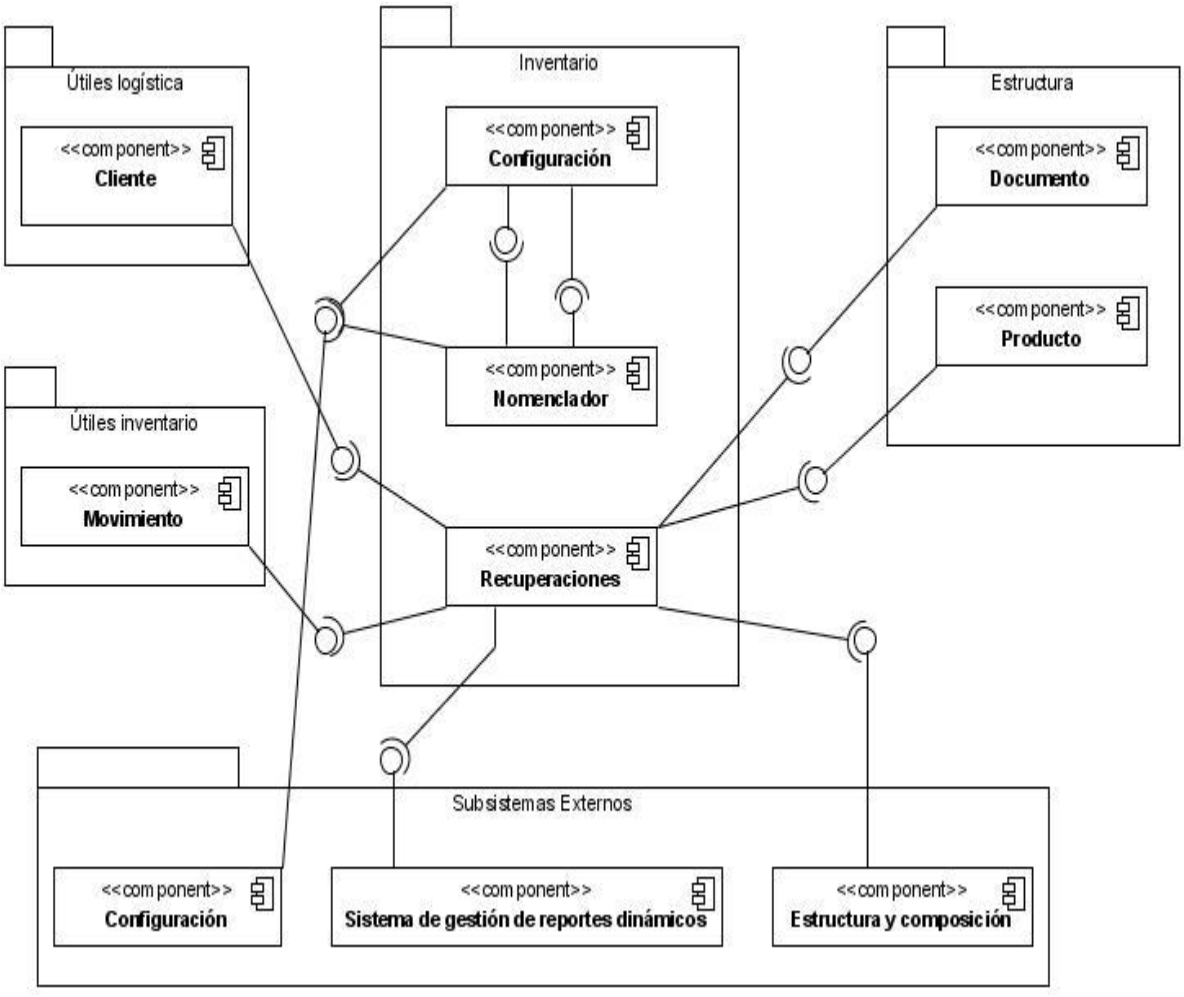
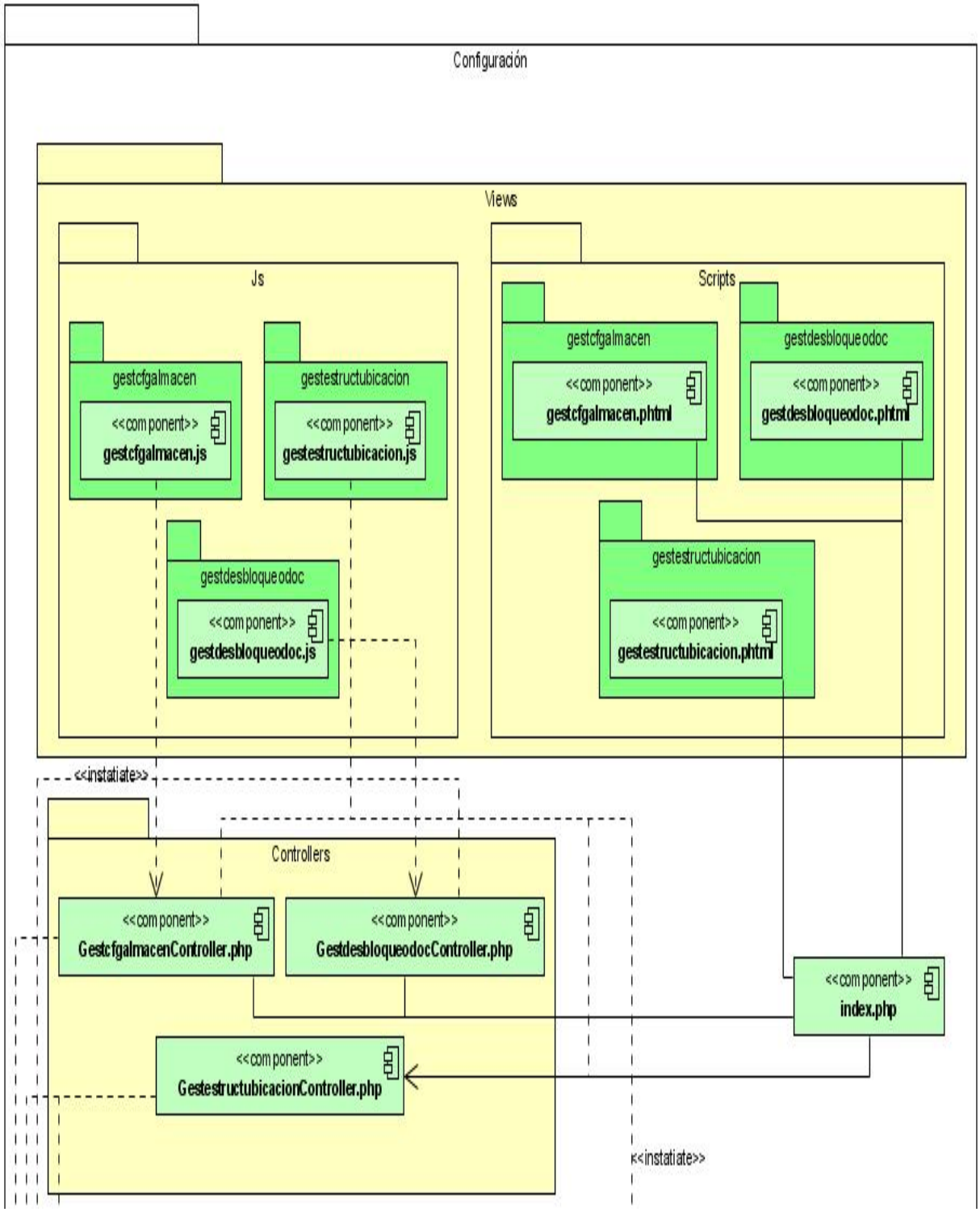


Figura 4. Diagrama de Integración entre Componentes


Configuración
Views
Js
gestcfgalmacen
<<component>>
gestcfgalmacen.js
gestestructubicacion
<<component>>
gestestructubicacion.js
gestdesbloqueodoc
<<component>>
gestdesbloqueodoc.js
Scripts
gestcfgalmacen
<<component>>
gestcfgalmacen.phtml
gestdesbloqueodoc
<<component>>
gestdesbloqueodoc.phtml
gestestructubicacion
<<component>>
gestestructubicacion.phtml
<<instatiate>>
Controllers
<<component>>
GestcfgalmacenController.php
<<component>>
GestdesbloqueodocController.php
<<component>>
GestestructubicacionController.php
<<component>>
index.php
<<instatiate>>

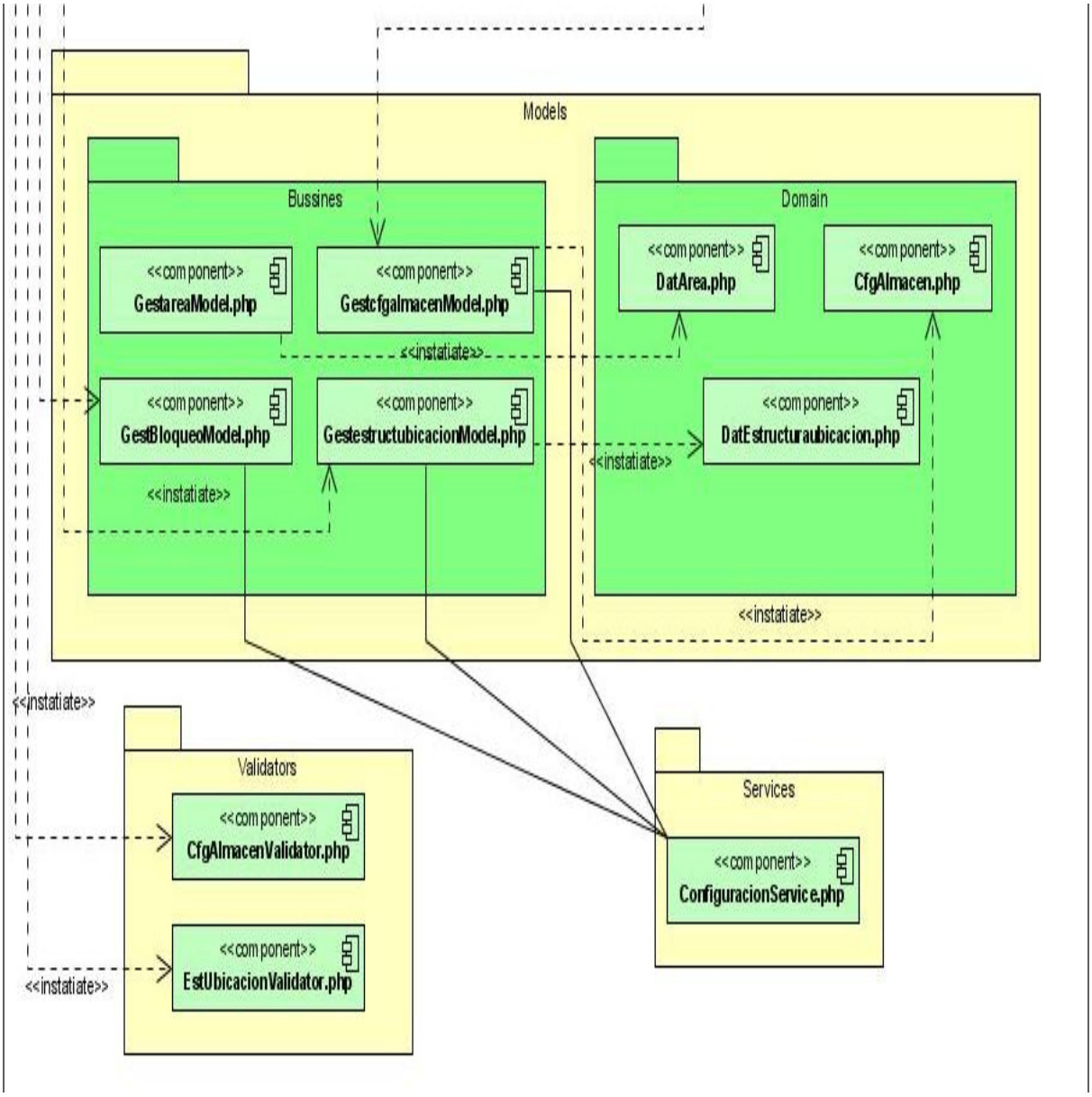


Figura 5. Diagrama de Componentes Internos

Diagrama de despliegue

El modelo de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los links de comunicación entre ellos, y las instancias de los componentes y objetos que residen en ellos. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento, y las conexiones entre estos elementos en el sistema. El modelo consiste en uno o más nodos (elementos de

procesamiento con al menos un procesador, memoria, y posiblemente otros dispositivos), dispositivos (nodos estereotipados con una capacidad de procesamiento en el nivel modelado de abstracción), y conectores, entre nodos, y entre nodos y dispositivos. El modelo de despliegue también mapea procesos dentro de estos elementos de procesamiento, permitiendo la distribución del comportamiento a través de los nodos que son representados *(Pressman, 1998).*

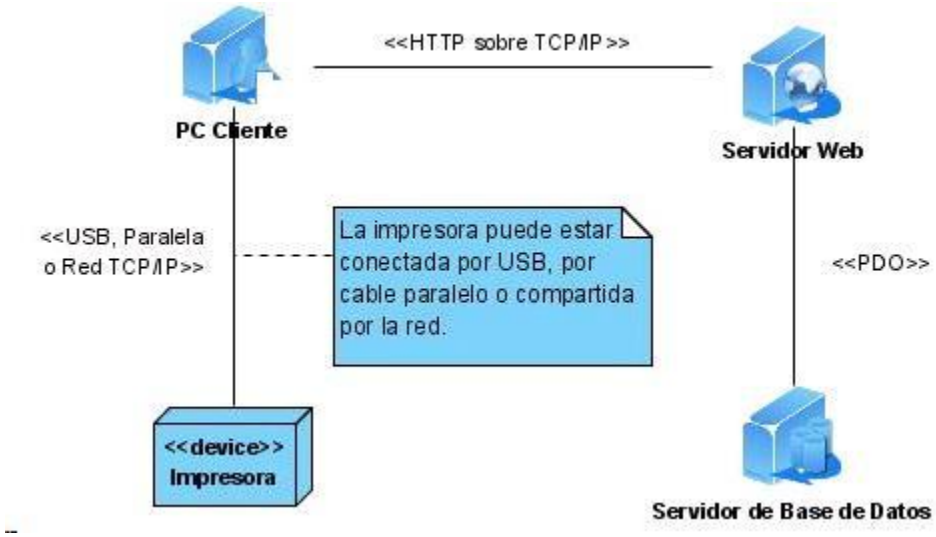


Figura 6. Diagrama de despliegue para el escenario de Clientes Ligeros

Propuesta del Sistema

Los componentes a implementar están enmarcados dentro del subsistema Inventario de la Línea Logística del Sistema Integral de Gestión de Entidades CEDRUX *(Ver Anexo 4).*

Uno de los primeros pasos que se realiza en una entidad es llevar a cabo el proceso de configuración del almacén. Para ello el sistema le dará al usuario según sus privilegios la opción de acceder al menú que contendrá las funcionalidades para la gestión de dicho proceso. Gestionar formato de estructura de ubicación es una de estas funcionalidades, el usuario al seleccionar esta opción podrá ver la configuración del almacén al cual accedió, si el almacén nunca ha sido configurado o el formato existente no ha sido usado en la creación de documentos o productos, el sistema dará la opción de escoger un nuevo formato, cambiar el existente y eliminar dicha configuración. Otras de las opciones que

brinda esta funcionalidad es chequear el real ordenado, siendo esta una configuración para los documentos de salida y facturas, que permitirá verificar que la cantidad de productos que el usuario va a llevar no sea menor que la cantidad especificada es estos documentos. Generar comprobante de operaciones es otra de las opciones a la hora de gestionar el formato del almacén, esta permite la integración con contabilidad financiera, quien tiene la responsabilidad de informar si se genera o no un comprobante de operaciones. Por último, la opción de categoría producto, permite saber si la entidad utiliza entre los atributos del producto, la categoría, posibilitando a las entidades militares el uso de la misma. *(Ver Anexo 5).*

Gestionar la estructura de ubicación de un producto es otra de las funcionalidades que entra en la configuración del almacén. Este proceso se inicia cuando el sistema muestra de forma jerárquica la estructura de la entidad a la cual el usuario ha accedido, mostrándose como nodo padre dicha entidad, en el primer nivel, los depósitos pertenecientes a la misma, en el segundo las áreas asociadas a cada uno de los depósitos y finalmente estarán las estructuras de ubicaciones de los productos *(Ver Anexo 6).* En este proceso el usuario tendrá la opción de adicionar, modificar y eliminar una estructura de ubicación *(Ver Anexo 7)*. Crear un área es uno de los primeros pasos para formar una estructura. Para ello hay que tener en cuenta la selección del formato del área, el código y la descripción. Ya creada un área, se podrán adicionar las estructuras internas de la misma, siempre teniendo en cuenta los parámetros del formato seleccionado (longitud, nivel), ya que el código para cada estructura será de la misma longitud del que fue definido en el nivel del formato dependiendo en el nivel que se encuentre la estructura.

Dentro del componente Nomencladores, una de las funcionalidades es la gestión de cada nomenclador; para ello el sistema le brinda al usuario la posibilidad para cada nomenclador las acciones de adicionar, modificar, eliminar. Hay que tener en cuenta que para llevar a cabo estas acciones se hacen diferentes validaciones, ejemplo de ellas es a la hora de adicionar; el usuario introduce los datos, al terminar esta operación el sistema comprueba que no existan estos datos duplicados en el nomenclador y muestra un mensaje de alerta en caso positivo *(Ver Anexo 8)*. Algo similar sucede a la hora de modificar, si el dato ya se encuentra en el nomenclador, el sistema alerta al usuario que

los datos están repetidos y no podrá ser modificado. En caso de que el usuario desee eliminar, el sistema verifica que el dato no se encuentre en ninguna operación, si el mismo está siendo utilizado se muestra un mensaje de alerta, dándole a conocer que ese recurso no puede ser eliminado porque está siendo utilizado y luego de esto lo desactiva *(Ver Anexo 9)*. Otra de las funcionalidades que brindan los nomenclador es la de activar la información, para esta el sistema habilita un botón en caso de que el usuario seleccione un recurso desactivado, preguntándole al mismo si desea activar la información. También el sistema permite hacer una búsqueda para cada nomenclador por diferentes parámetros.

Para llevar a cabo el proceso de las recuperaciones del subsistema de inventario, el sistema inicialmente le mostrará al usuario un menú principal (Recuperaciones) y dentro de este los diferentes casos por el cual el usuario desea recuperar datos. Seguidamente el usuario tendrá que introducir o simplemente escoger algunos datos de filtrado, que le permitirá al sistema llevar a cabo la recuperación por estos parámetros. Un ejemplo de estas recuperaciones es (Productos con déficit-ingreso al presupuesto), en este caso el usuario tendrá la opción de acceder al componente de producto, que mostrará todos los productos existentes en la entidad, para que el mismo pueda escoger los productos que desea verificar *(Ver Anexo 10)*, también podrá seleccionar un rango de fecha para enmarcar el período y si la recuperación va a ser por déficit o ingreso. Ya seleccionados estos parámetros, el sistema buscara en la base de datos por medio de una función, los datos que concuerden con los datos de filtrado introducidos por el usuario.

2.6 COMPONENTES REUTILIZADOS

El subsistema Inventario posee implementados varios componentes que brindan servicios necesarios para la solución propuesta. A continuación se realiza una breve descripción de los mismos.

Componente Documento

Tiene como principal función gestionar todos los documentos de la entidad.

Componente Producto

Es el componente encargado de la gestión de los productos dentro del almacén, ya sea los que están nomenclados, como los nuevos que se incluyen. Este servicio brinda todas las funcionalidades necesarias para operar sobre los productos, que a través de los movimientos se relacionan con los documentos.

Componente Cliente

El componente cliente tiene una función muy importante, ya que es el encargado del manejo de todos los clientes o proveedores de la empresa. Muestra el listado de los mismos, dando la posibilidad de realizar la selección múltiple, permite además eliminar clientes previamente seleccionados.

Componente Movimiento

El componente movimiento es el encargado de la unión entre el documento y los productos. Queda constancia documental de la operación realizada sin afectar los valores reales del producto.

2.7 ARQUITECTURA

Existen varios lugares donde la arquitectura está presente, principalmente en el desarrollo de aplicaciones informáticas, sin lugar a duda lo que la convierte en una herramienta de puntería para la organización del trabajo, partiendo de esto se puede decir que hay diferentes caminos por lo cual guiar un trabajo o un proyecto. Para el desarrollo de aplicaciones hay creados diferentes estilos arquitectónicos los cuales solucionan un problema, en dependencia de las características del mismo, ejemplo de esto tenemos el modelo-vista-controlador, arquitectura en capas entre otros. Por tanto es de vital importancia una selección óptima del mismo *(IEEE, 2004).*

En la implementación de los componentes Configuración, Nomencladores y Recuperaciones, se utilizó el Estilo basado en Capas y el Modelo-Vista-Controlador, el

primero para las capas que nos brindan los servicios con módulos externos y los propios servicios internos del subsistema Inventario. El segundo estilo se manifiesta con la utilización del Zend Framework para implementar la lógica del negocio.

Estilo basado en capas

Cada capa puede ser modificada tanto como sea necesario sin provocar cambios en las demás. Una capa no tiene dependencias con la capa superior, cada capa depende solamente de la fachada que le permite comunicarse con la capa inmediata inferior. Esta dependencia entre capas es normalmente a través de fachadas, asegurando que el acoplamiento sea el más bajo posible y la abstracción del funcionamiento de la capa inferior, sea casi total.

Esto permite el desarrollo de aplicaciones más robustas debido al encapsulamiento. Es factible un mantenimiento y soporte más sencillo así como una mayor flexibilidad *(Expósito Álvarez, y otros, 2008).*

Modelo-Vista-Controlador

La base arquitectónica de la solución se ha modelado haciendo uso del Patrón (MVC) con el objetivo de utilizar la separación de las responsabilidades de cada una de las capas que lo conforman y así lograr facilidades de desarrollo.

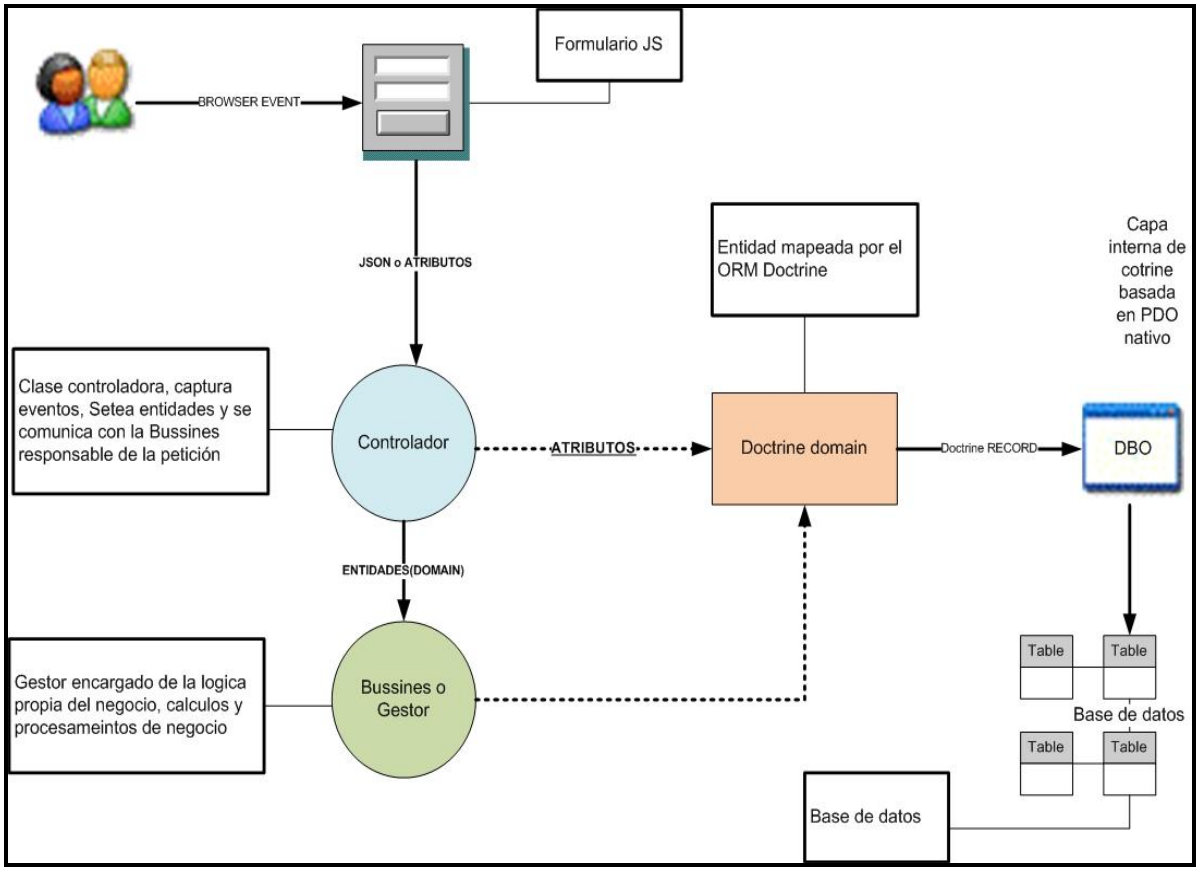


Figura 7. Estructura del patrón Modelo-Vista-Controlador (MVC)

Modelo: Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.

Vista: Intercambia la información con el usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.

Controlador: Recibe las entradas, traducidas a solicitudes de servicio para el modelo *(Hernández Reyes, y otros, 2008).*

2.8 DESCRIPCIÓN DE CLASES

Clases Controladoras

La clase controladora es responsable de ejecutar una determinada lógica en función de las acciones que se producen en una aplicación. Se encargan de la captura de eventos, del seteo de entidades y permiten la comunicación con las clases del negocio.

Nombre: GestcfgalmacenController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
cargarcomboformatoAction	Carga todos los formatos existentes en Configuración
salvarcfgalmacenAction	Adiciona una nueva configuración de almacén si no hay ninguna existente y permite modificar los datos de las mismas.
eliminarcfgalmacenAction	Elimina una configuración de almacén.
verificarcfgAction	Devuelve los datos de la configuración así como la existencia de productos en la estructura.

Tabla 1.Descripción de la clase GestcfgalmacenController

Nombre: GestestructubicacionController	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	**Descripción:**
longitudnivelAction	Devuelve de un formato dado, la longitud del nivel.
cargarpadreAction	Carga el nodo padre del árbol de las estructuras de ubicaciones.

cargararbolAction	Carga el árbol de las estructuras de ubicaciones existentes en la entidad.
cargarformestubicAction	Devuelve los datos de la estructura de ubicación.
cargarformareaAction	Devuelve los datos de un área.
cargarcomboformatoAction	Carga todos los formatos existentes en Configuración
adicionarestubicAction	Adiciona una nueva estructura de ubicación.
modificarestubicAction	Modifica los datos de una estructura de ubicación.
eliminarestubicAction	Elimina una estructura de ubicación.
adicionarareaAction	Adiciona una nueva área a la estructura de ubicación.
modificarareaAction	Modifica un área de la estructura de ubicación.
eliminarareaAction	Elimina un área de la estructura de ubicación.

Tabla 2.Descripción de la clase GestestructubicacionController

Nombre: GestnomproductoController	
Tipo de clase: Controladora	
Atributo	**Tipo**
$idformato	numérico
$idespecialidad	numérico
$idestructura	numérico

Para cada responsabilidad:	
Nombre:	**Descripción:**
traerpartesformatoAction	Devuelve las partes de un formato dado.
longitudformatoAction	Devuelve la longitud de un formato.
obtenerafgalmacenAction	Devuelve el formato para los productos de un almacén dado.
longitudnivelAction	Devuelve la longitud del nivel de un formato.
obtenerespecialidadAction	Devuelve la especialidad de una estructura dada.
cargarnomencladorAction	Carga todos los productos del nomenclador .
cargarformproductoAction	Carga los datos de un producto dado su id.
cargaprodarbolAction	Carga el árbol del nomenclador de producto.
cargarunidadmedidaAction	Carga la unidad de medida del producto.
cargarnacionalidadAction	Carga la nacionalidad del producto.
adicionarnomprodtreeAction	Adiciona un nivel al árbol de producto.
adicionarnomprodgridAction	Adiciona un nuevo producto.
modificarprodtreeAction	Modifica un nivel al árbol de producto.
modificarprodgridAction	Modifica un producto.
eliminarprodtreeAction	Modifica un producto.

Tabla 3. Descripción de la clase GestnomproductoController

Clase Auxiliares

Las clases auxiliares son las que guardan poca o ninguna información del estado por si misma, pero asisten en la ejecución de tareas complejas. Se encargan de la lógica propia del negocio, cálculos y son las responsables de las peticiones echas por la clase controladora.

Nombre: GestestructubicacionModel	
Tipo de clase: Auxiliar	
Para cada responsabilidad:	
Nombre:	**Descripción:**
adicionarEstUbic($idnodo, $objubic)	Adiciona una nueva estructura de ubicación pasándole todos los datos de la misma.
modificarEstUbic($objEstUbic, $codsel, $codanterior, $codmod)	Modifica una estructura de ubicación pasándole los datos a modificar.
eliminarEstUbic($idestubic)	Elimina una estructura de ubicación pasándole el id de la misma.
getIdFormato($idarea)	Devuelve el idformato de un área determinada.
getAreasDeposito($iddeposito)	Devuelve todas las aéreas asociadas a un depósito.
obtenerEstUbicacion($idestubic, $longn)	Devuelve todos los datos de la estructura de ubicación.
verificaExistenciaProducto($idarea)	Verifica la existencia de productos en la estructura de ubicación del área especificada.
obtenerUbicacion($idestubic)	Devuelve la ubicación a partir de un idestructura.

Tabla 4. Descripción de la clase GestestructubicacionModel

Nombre: GestnomproductoModel	
Tipo de clase: Auxiliar	
Para cada responsabilidad:	
Nombre:	**Descripción:**
insertarElementNomProd($idpadre, $objProd)	Inserta un nuevo elemento en el nomenclador de producto.
modificarElementNomProd($objProducto, $codprod, $codanterior, $codmod)	Modifica un elemento en el nomenclador de producto.
modificarElementNomProdGrid($objProducto)	Modifica los datos de un producto.
eliminarElementNomProd($objNomProducto)	Elimina un producto.
obtenerProductosNomenclador($start, $limit, $idprod, $idestructura, $idespecialidad, $codarbol, $longform, $codigo, $nropieza, $descripcion)	Devuelve los productos del nomenclador.
devolverArbolNomProductos()	Devuelve el árbol del nomenclador de producto.
obtenerNomProductoDadoId($idprod, $longn, $tipo)	Devuelve el nomenclador de producto dado un id.

Tabla 5. Descripción de la clase GestnomproductoModel

Clases Entidad

Las clases entidad modelan información que poseen una larga vida y que a menudo son conceptos. También se denominan clase dominio, ya que suelen tratar con abstracciones de entidades del mundo real. Se encargan de modelar la información del sistema y el comportamiento asociado a una información.

Nombre: Datestructuraubicacion	
Tipo de clase: Entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**Descripción:**
existeEstUbic($idnodo, $cod, $pos)	Verifica la existencia de una estructura de ubicación.
obtenerUbicacionDadoEstructura($idnodo, $pos)	Devuelve la ubicación dado una estructura.
obtenerAreas($idubicalmacen)	Devuelve las áreas asociadas a una ubicación dada.
obtenerAreasDep($deposito)	Devuelve todas las áreas asociadas a un depósito.
obtenerParteFormato($idarea)	Devuelve el idformato de un área determinada.
obtenerDatEstUbic($codsel)	Devuelve la estructura de ubicación.
obtenerEstUbicDadoIdArea($idarea)	Devuelve la estructura de ubicación de un área específica.
obtenerEstUbicDadoIdEstUbic($idestubic)	Devuelve una estructura de ubicación dado el id de la misma.

Tabla 6. Descripción de la clase Datestructuraubicacion

Nombre: Nomproducto
Tipo de clase: Entidad

Para cada responsabilidad:	
Nombre:	**Descripción:**
obtenerProdDadoIdEstructura($idestructura)	Devuelve los productos pertenecientes a una estructura dada.
devolverNomProd($idespecialidad,$logform, $idestructura= '', $codigo = '', $nropieza = '', $descripcion = '')	Devuelve el nomenclador de producto.
obtenerProductosNomenclador($idprod, $idestructura='', $idespecialidad='', $codarbol, $longform, $codigo = '', $nropieza = '', $descripcion = '')	Devuelve los productos que concuerden con los parámetros pasados.
devolverNomProdDadoIdProd($idprod, $codigo='', $nropieza='', $descripcion='')	Devuelve el nomenclador de producto dado un idproducto.
getDatosProducto($codigoesp,$codigocup, $codigo = '', $nropieza = '', $descripcion = '')	Devuelve los datos de un producto dado.
devolverArbolNomProductos($idnodo, $idestructura='', $idespecialidad, $nivel, $longitud)	Devuelve el árbol del nomenclador de producto.
obtenerEstUbicDadoIdArea($idarea)	Devuelve la estructura de ubicación de un área específica.
obtenerEstUbicDadoIdEstUbic($idestubic)	Devuelve una estructura de ubicación dado el id de la misma.

Tabla 7. Descripción de la clase Nomproducto

2.9 ANÁLISIS DE LA COMPLEJIDAD DE ALGORITMOS

Para conocer la complejidad del algoritmo es necesario calcular la complejidad ciclomática del mismo, para hacer dicho cálculo, es imprescindible primeramente tener el

código o el diseño del procedimiento, luego marcar cada instrucción del código con un número, que representa cada lugar del camino que puede seguir la secuencia del algoritmo, a continuación se representa el código con sus instrucciones enmarcadas. Ver figura 8.

```
public function cargararbolAction() {
 $arrDatos = array ( ); 1
 $arrDatosl = array ( ); 1
 $arrRes = array ( ); 1
 $iddeposito = $this->_request->getPost ( 'node' ); 1
 $pos = $this->_request->getPost ( 'pos' ); 1
 $idarea = $this->_request->getPost ( 'idarea' ); 1
 $idestructuraubic = $this->_request->getPost ( 'idestructuraubic' ); 1
    if ($iddeposito == 0) 2 {
      $depositos = $this->integrator->metadatos->DameEstructurasInternas ( $this->global->Estructura->idestructura, false ); 3
      $k = 0; 3
        foreach ( $depositos as $key => $Obj ) 4 {

          if ($Obj->idnomeav == 8) 5 {
             $arrDatos [$k] ['id'] = $Obj->idestructuraop; 6
             $arrDatos [$k] ['text'] = $Obj->denominacion; 6
             $arrDatos [$k] ['leaf'] = false; 6
             $k ++; 6
             }
          } 7
```

```
            echo json_encode ( $arrDatos ); 8
        } else

        if ($iddeposito != 0 && $pos == 1) 9 {
            $areas = GestestructubicacionModel::getareasdeposito ( $iddeposito ); 10

            foreach ( $areas as $index => $Obj ) 11 {
                $arrDatosl [$index] ['id'] = $Obj['idarea'].$Obj['idarea']; 12
                $arrDatosl [$index] ['idarea'] = $Obj['idarea']; 12
                $arrDatosl [$index] ['text'] = $Obj['descripcion']; 12
                $arrDatosl [$index] ['idformato'] = $Obj['idformato']; 12
                $arrDatosl [$index] ['codigoarea'] = $Obj['codigoarea']; 12

            } 13
            echo json_encode ( $arrDatosl ); 14
        } else

        if ($iddeposito != 0 && $pos >= 1) 15 {
            $res = DatEstructuraubicacion::obtenerUbicacionDadoEstructura ( $idestructuraubic, $pos ); 16

            foreach ( $res as $index => $objres ) 17 {
                $arrRes [$index] ['id'] = $objres->idestructuraubicacion.$objres->idarea; 18
                $arrRes [$index] ['text'] = $objres->descripcion; 18
                $arrRes [$index] ['nivel'] = $objres->nivel; 18
                $arrRes [$index] ['idestructuraubic'] = $objres->idestructuraubicacion; 18
                $arrRes [$index] ['idarea'] = $objres->idarea; 18
                $arrRes [$index] ['codigo'] = $objres->codigo; 18
                $area = Doctrine::getTable ( 'DatArea' )->find ( $idarea ); 18
                if ($objres->nivel == count ( $this->integrator->parametros->BuscarParteFormato ( $area->idformato ) )) 19
                    $arrRes [$index] ['leaf'] = true; 20 else
                    $arrRes [$index] ['leaf'] = false; 21
            } 22
            echo json_encode ( $arrRes ); 23
    }
} 24
```

Figura 8. Algoritmo cargararbolAction()

Un grafo de flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, su comprensión y brindan información para confirmar que el trabajo se está haciendo adecuadamente.

Componentes del grafo de flujo

- Nodo: Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede

ser también que hallan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

- Aristas: Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.
- Regiones: Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

En la figura 9 se observa la representación un grafo de flujo, en el cual aparecen sus componentes.

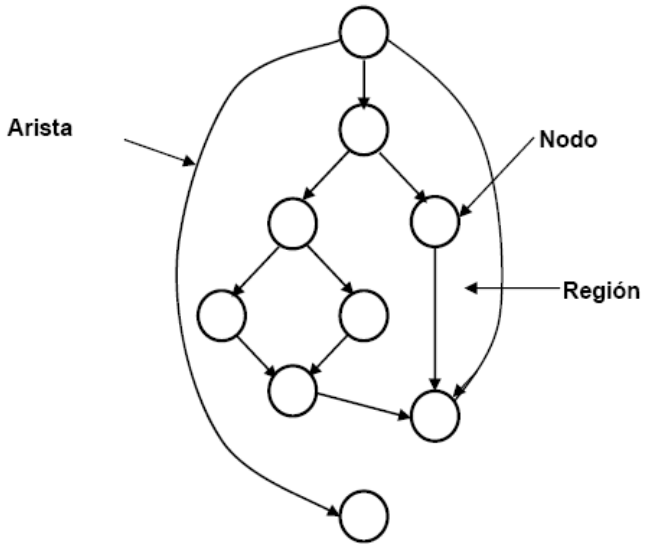


Figura 9. Componentes de los grafos de flujo

La complejidad ciclomática es una métrica de software extremadamente útil, pues proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado como complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite superior para el número de pruebas que se

deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Ver figura 10

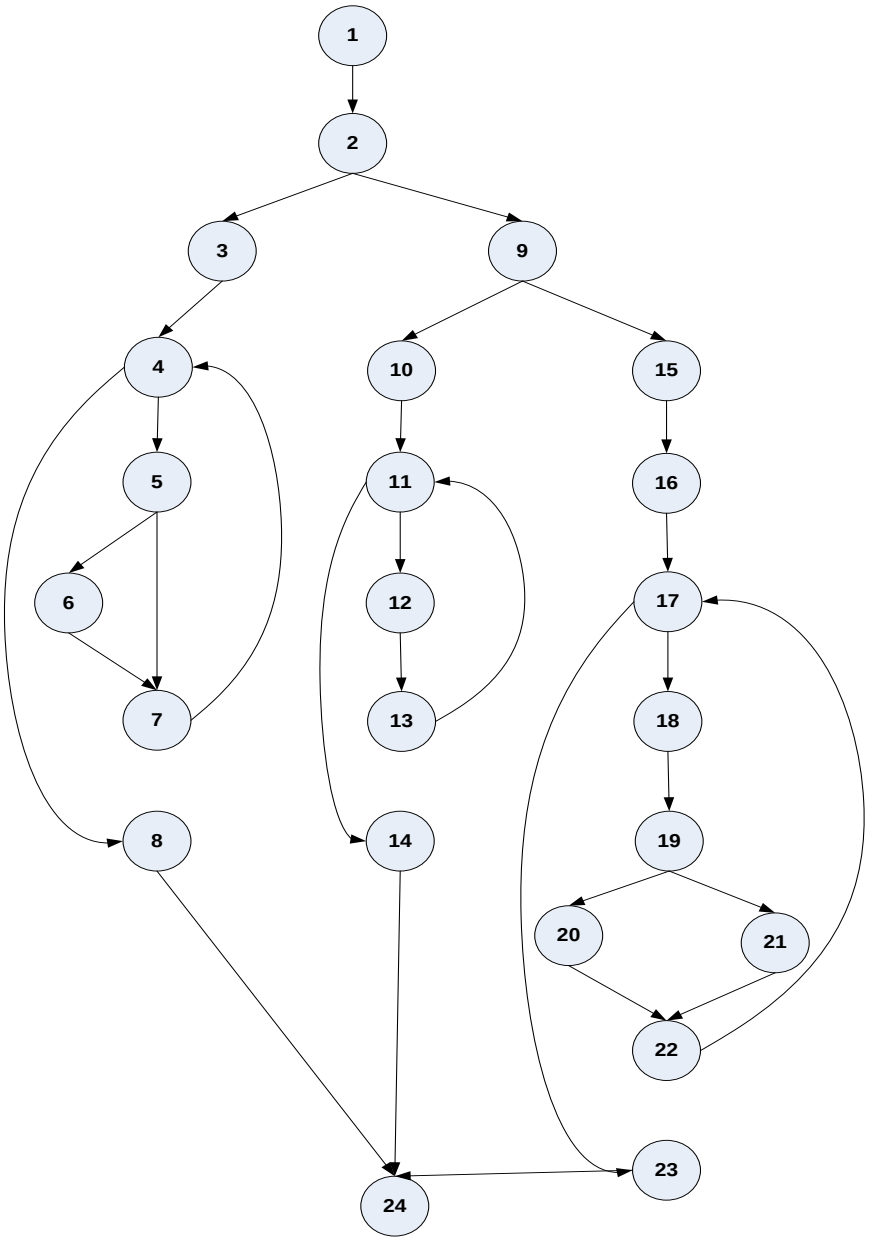


Figura 10. Grafo de flujo del algoritmo cargararbolAction()

Formas fundamentales de calcular la complejidad:

1. La complejidad ciclomática, V(G), se define como:

   V(G) = A - N + 2

   donde: A es el número de aristas del grafo y N es el número de nodos.

2. La complejidad ciclomática, V(G), también se define como:
   V(G) = P + 1
   donde: P es el número de nodos predicado contenidos en el grafo G.
3. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.

Seguidamente a la construcción del grafo de flujo se procede a efectuar el cálculo de la complejidad ciclomática del código, el cálculo es necesario efectuarlo mediante tres vías o fórmulas que para concluir que el cálculo fue correcto es necesario que por las tres vías el resultado sea el mismo, las fórmulas para calcular son las siguientes:

1. V (G) = (A – N) + 2

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

2 V (G) = (30 – 24) + 2

V (G) = 8

2. 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) = 7 +1

V (G) =8

3. V (G) = R

Siendo “R” la cantidad total de regiones, para cada formula “V (G)” representa el valor del calculo.

V (G) = 8

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 8 que da la visión de que existen a lo sumo ocho caminos lógicos por donde recorrer el algoritmo.

2.10 CONCLUSIONES PARCIALES

Con el desarrollo de este capítulo se conocieron los procesos objeto de automatización, lo que proporcionó una temprana idea de la complejidad de la solución. Se explicaron los componentes que fueron reutilizados para el desarrollo de la aplicación, teniendo en cuenta la arquitectura del sistema. Todo lo anterior posibilitó la implementación de las interfaces. La descripción de clases importantes permitió tener una visión para la puesta en práctica de patrones como el Bajo Acoplamiento y la Alta Cohesión. De forma general se logró implementar los componentes de Configuración, Nomencladores y Recuperaciones, pertenecientes al subsistema de Inventario del Sistema de Gestión Integral Cedrux.

CAPÍTULO III VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 INTRODUCCIÓN

En la implementación de diferentes sistemas informáticos juega un papel fundamental el uso de las técnicas de evaluación dinámica o pruebas. Para lograr esto se debe llevar a cabo estrategias a la hora de evaluar dinámicamente el software, porque para la evaluación del mismo se debe comenzar por los componentes más simples y más pequeños e ir avanzando progresivamente hasta probar todo el software en su conjunto. Es por eso que la técnica utilizada es las Pruebas Unitarias, pues es una forma de probar el correcto funcionamiento de un módulo de código. En este capítulo se aplican las pruebas unitarias de Camino Básico

Hoy en día se calcula que la fase o proceso de pruebas representa más de la mitad del coste de un programa ya que requiere, un tiempo similar al de la programación lo que obviamente acarrea un alto costo económico, siendo esta etapa más cara que el propio desarrollo y diseño de los distintos programas que conforman el sistema.
Así mismo el uso de métricas tributa directamente a una mejor calidad, es la forma de medir características propias de cada sistema. En este capítulo se aplican métricas para medir la calidad de la implementación.

3.2 PRUEBAS DE SOFTWARE

Las pruebas de software son un conjunto de herramientas, técnicas y métodos que evalúan el desempeño de un programa. Involucran las operaciones del sistema bajo condiciones controladas y evaluando los resultados, es por eso que la realización de las mismas a los software es un factor de vital importancia. Antes de liberar el producto es necesario tener la certeza de que este se encuentra libre de errores, aunque se considera que no se puede estar 100% seguro de esto. Pero una cosa si es cierta, mientras más pruebas se le realicen al software más cerca se podrá estar de lograr un producto con la

calidad esperada por los usuarios. De las pruebas se plantean varias normas que pueden servir acertadamente como objetivos de las mismas.
Probar es un proceso de ejecución de un programa con la intención de descubrir un error. Una prueba tiene éxito si se descubre un error no detectado hasta entonces *(Myers, 1979).*

3.3.1 Pruebas de Unidad

Es la escala más pequeña de la prueba, está basada en la funcionalidad de los módulos del programa, como funciones, procedimientos y módulos de clase. En ciertos sistemas también se verifican o se prueban los drivers y el diseño de la arquitectura *(Pressman, 1998).*

Los casos de pruebas que se diseñen para este nivel de pruebas, deben descubrir errores como:

- Comparaciones entre tipos de datos distintos.
- Operadores lógicos o procedencia incorrecta.
- Igualdad esperada cuando los errores de precisión la hacen poco probable.
- Las variables o comparaciones incorrectas.
- Terminaciones de bucles inapropiadas o inexistentes.
- Fallo de salida cuando se encuentra una iteración divergente.
- Bucles que manejan variables modificadas de forma inapropiada.

Para probar los componentes implementados como unidades individuales, se realizan las pruebas de especificación o de caja negra, que verifican el comportamiento de la unidad observable externamente y pruebas de estructura, o de caja blanca, que verifican la implementación interna de la unidad *(Santos Lebeque, y otros,, 2008).*

Prueba del camino básico

La técnica del camino básico permite obtener una medida de la complejidad lógica del código de cada método, programa o módulo dado. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes que existen en la codificación por los cuales puede circular el flujo de control. Es además una de las más eficientes en

cuanto a cobertura de código, pues logra que se ejecuten todos los bucles en sus límites operacionales *(Márquez Alpízar, y otros, 2008).*

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Un camino es aquel para el cual puede efectuarse una traza a través del código desde el comienzo del programa hasta el final del mismo. Hay que tener en cuenta que dos caminos son diferentes si se ejecutan distintas sentencias o las mismas pero en diferente orden.

Alcance: Se le aplicará a la función BuscarDocRec.

Para ejecutar el test es necesario realizar primeramente los procesos descritos en el capítulo 2, epígrafe 2.9 “Análisis de la complejidad de algoritmos” para calcular los valores de la complejidad ciclomática del método al cual se le va a aplicar la prueba. A continuación se enumera las sentencias de código del procedimiento. Ver figura 11

```
public function BuscarDocRec($idusuario,$idestructura, $datos)
{

  $doc = $this->pIntegrator->documentos->BuscarDocRec($idusuario,$idestructura, $datos); 1

  if($doc->cant != 0) 2
  {
    foreach($doc[0]->datos as $index => $client) 3
    {
      $idcliente = $client['idcliente']; 4
      if($idcliente) 5
      {
         $nomcliente = $this->pIntegrator->utilesLogistica->getClientesNombre($idcliente); 6
         $doc[0]->datos[$index]['cliente'] =$nomcliente ; 6
      }
    } 7
  }
  return $doc[0]; 8
}
```

Figura 11. Algoritmo BuscarDocRec()

Seguidamente en la Figura 12 se muestra el grafo de flujo asociado al código anterior.

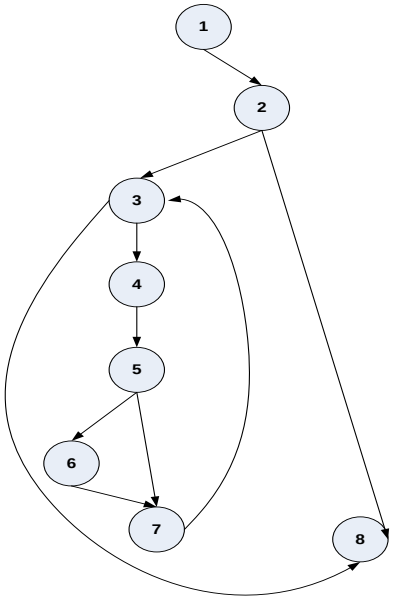


Figura 12. Grafo de flujo del algoritmo BuscarDocRec()

1. V (G) = (A – N) + 2

V (G) = (10 – 8) + 2

V (G) = 4

2. V (G) = P + 1

V (G) = 3 +1

V (G) =4

3. V (G) = R

V (G) = 4

Realizado el cálculo por las 3 vías necesarias se llega a la conclusión que el algoritmo presentado anteriormente tiene un complejidad ciclomática de 4 que da la visión de que existen a lo sumo cuatro caminos lógicos por donde recorrer el algoritmo.

Camino básico #1:
1 - 2 - 8

Camino básico #2:
1 - 2 – 3 - 4 - 5 - 6 - 7- 3 - 4 - 5 – 7 - 8

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico.
Para realizar los casos de pruebas es necesario cumplir con las siguientes exigencias:
Descripción: Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
Condición de ejecución: Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
Entrada: Se muestran los parámetros que entran al procedimiento
Resultados Esperados: Se expone resultado que se espera que devuelva el procedimiento.

Caso de prueba para el camino básico # 1.

Descripción: Los datos de entrada cumplirán con los siguientes requisitos: El *$idusuario*, *$idestructura y $datos['nrodoc']* serán números enteros.

Condición de ejecución: el *$datos['nrodoc']* será igual a 1000000325

Entrada: *$datos['nrodoc']* = 1000000325.

Resultados esperados:

Se muestra un mensaje informando que no existe un documento con esas características.

Caso de prueba para el camino básico # 2.

Descripción: La misma descrpición que el caso de prueba para el camino básico 1.

El *$idusuario*, *$idestructura, $datos['nrodoc'] y $datos['tipo']* serán números enteros.

Condición de ejecución: el *$datos['nrodoc']* será igual a 14 y *$datos['tipo']* = 110

Entrada: *$datos['nrodoc']* = 14; *$datos['tipo']* = 10

Resultados esperados:

Se muestra el documento 14 de tipo Factura

.

3.4 VALIDACIÓN DE LA IMPLEMENTACIÓN

El desarrollo de software es algo muy complejo y la medida de su calidad real no es automatizable. Por esta razón la aplicación de métricas de calidad para evaluar el diseño orientado a objeto posee gran importancia. A continuación presentaremos un estudio que brinda un esquema sencillo de implementar y que a la vez cubre los principales atributos de calidad de software, siendo esto la principal razón de la concepción de las métricas.

Atributos de calidad que se abarcan:

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

2. Complejidad del diseño: Consiste en la complejidad que posee una estructura de diseño de clases.

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

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

5. Acoplamiento: Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, esta muy ligada a la característica de Reutilización. Complejidad del mantenimiento. Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

6. Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (Componente, modulo, clase, conjunto de clases, etc.) diseñado.

7. Nivel de Cohesión: Consiste en el grado de especialización de las clases concebidas para modelar un dominio o concepto específico. *(Yzquierdo Herrera, y otros, 2007)*

3.4.1 Tamaño operacional de clase (TOC)

Esta métrica está relacionada con el número de métodos asignados a una clase. Teniendo en cuenta los siguientes atributos:

Atributo que afecta	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la

	responsabilidad asignada a la clase.
Complejidad de Implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 8. Tamaño operacional de clase (TOC)

Ver instrumentos y tabla de resultados en (Anexo 11 Instrumento de medición de la métrica Tamaño operacional de clase (TOC)).

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

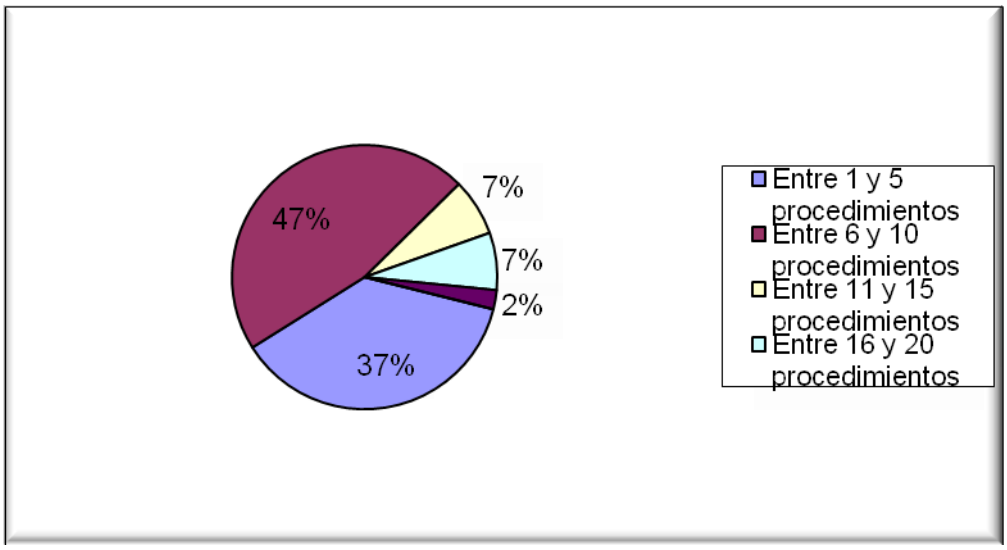


Figura 13. Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

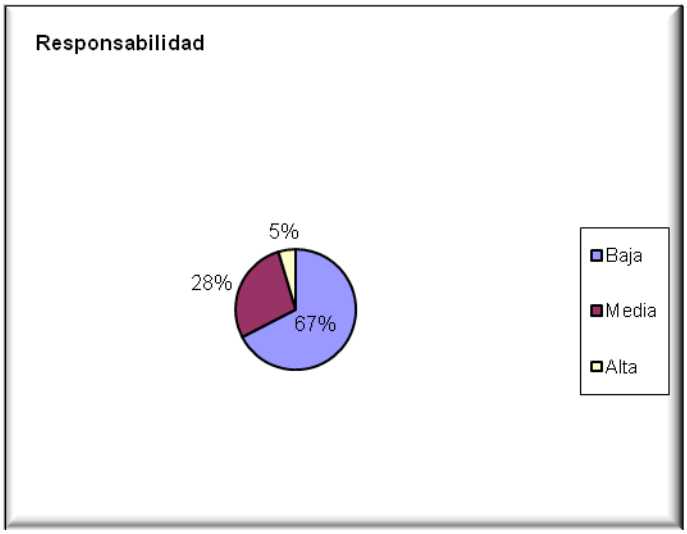


Figura 14. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad

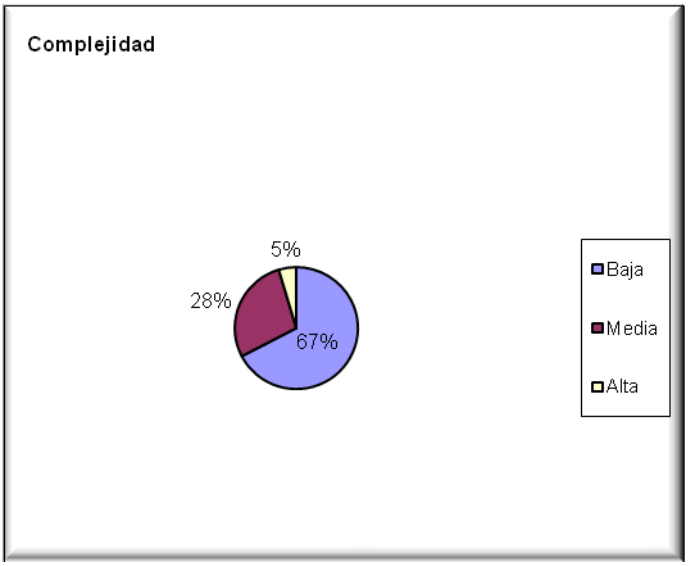


Figura 15. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación

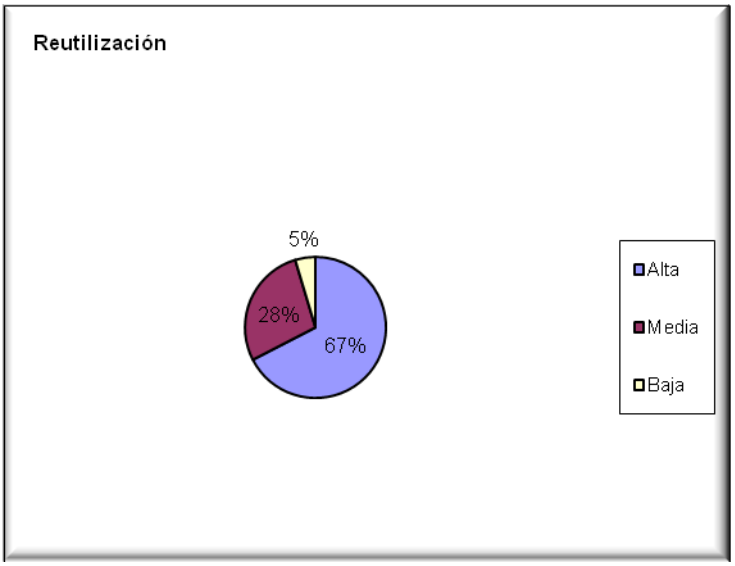


Figura 16. Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que el diseño del componente Nomenclador y Configuración tienen una buena calidad pudiéndose observar que el 84% de las clases posee menos cantidad de operaciones que la media registrada en las mediciones. Además el 67% de las clases posee evaluaciones positivas en los atributos (Responsabilidad, Complejidad de Implementación y Reutilización).

3.4.2 Relaciones entre clases (RC)

Está dado por el número de relaciones de uso de una clase con otras.

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

Tabla 9. Relaciones entre clases (RC)

Ver instrumentos y tabla de resultados en (Anexo 12 Instrumento de medición de la métrica Relaciones entre clases (RC)).

Luego de realizado un análisis profundo en cada una de las clases utilizadas, la evaluación de la métrica reflejó lo siguiente:

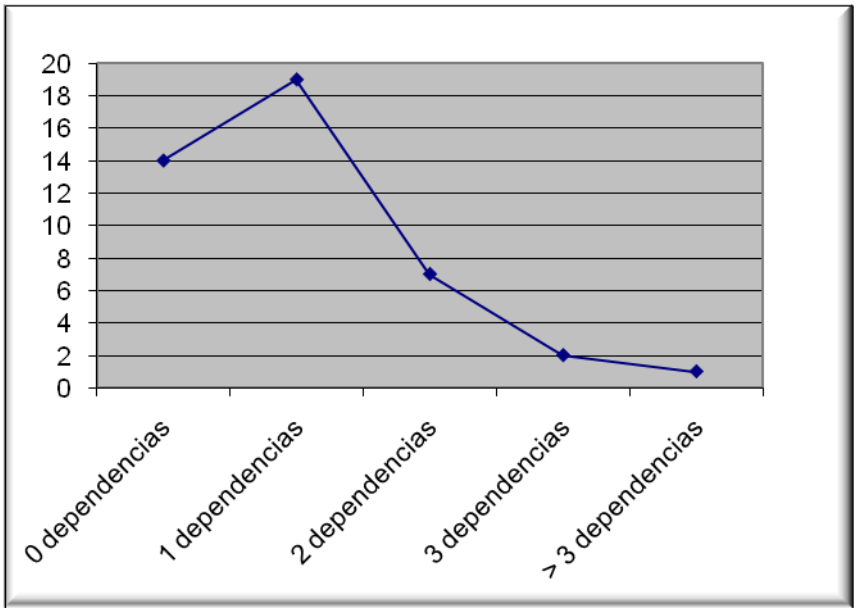


Figura 17. Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los Valores

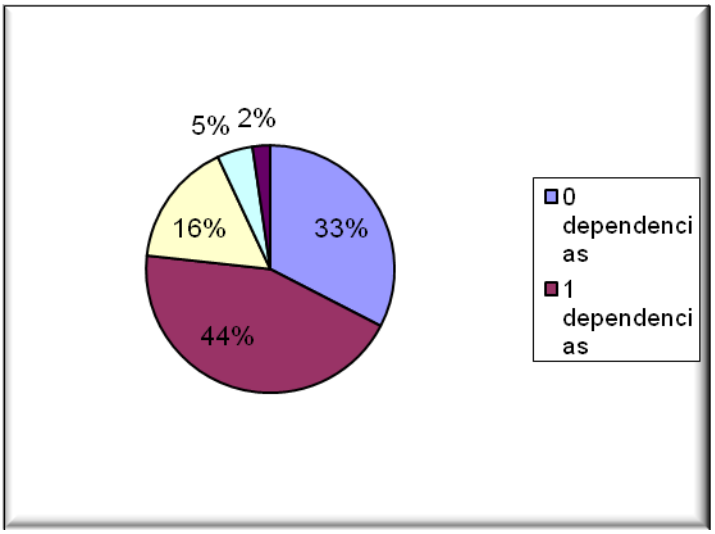


Figura 18. Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos

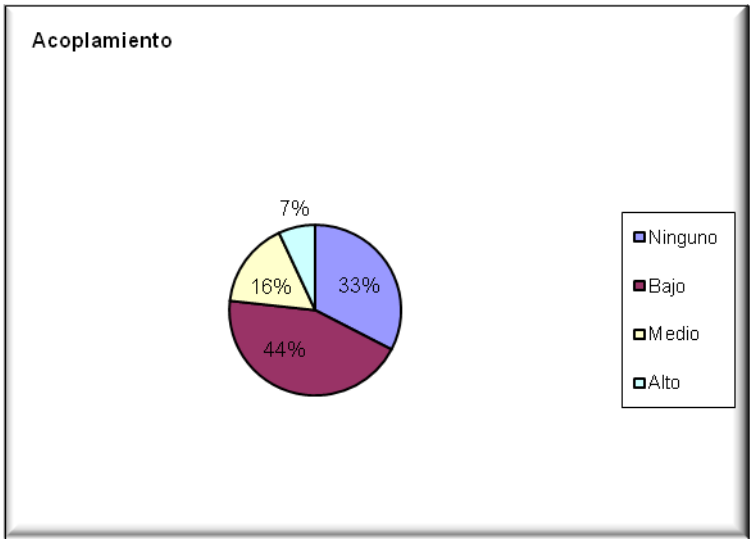


Figura 19. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento

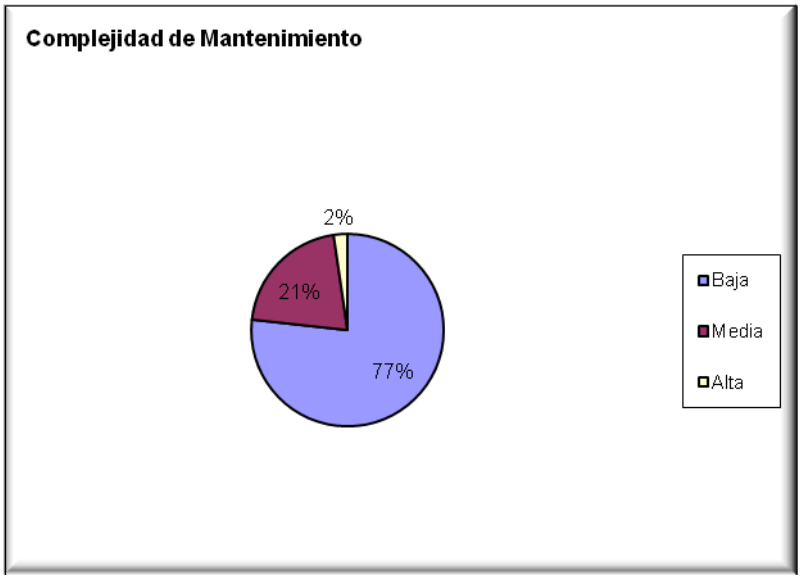


Figura 20. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento

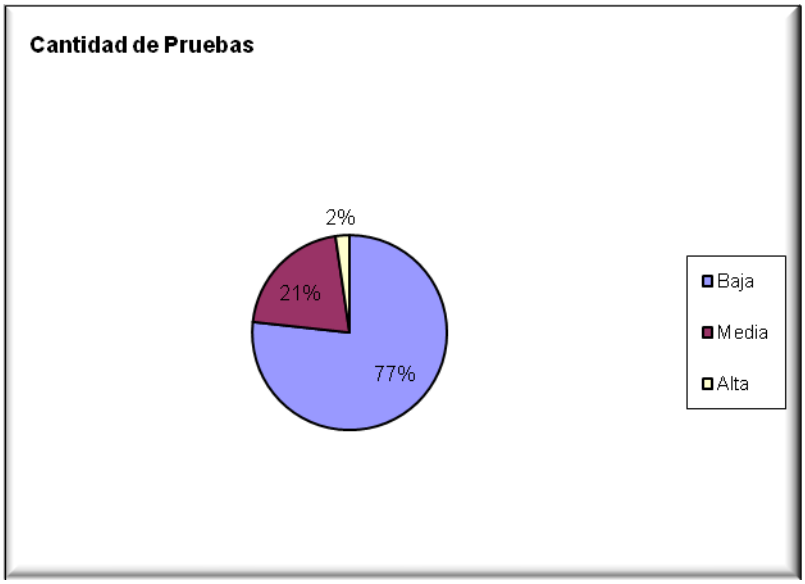


Figura 21. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas

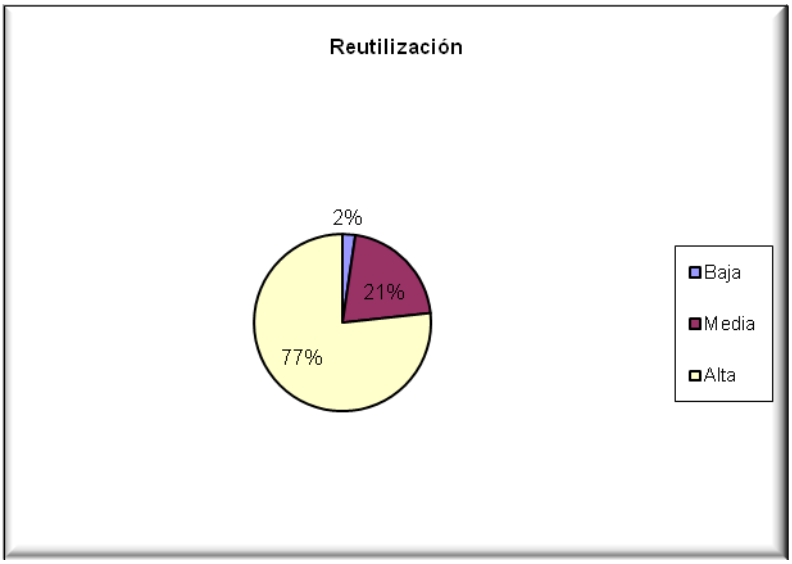


Figura 22. Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización

Haciendo un análisis de los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño de los componente Configuración, Nomencladores y Recuperaciones tiene una buena calidad pudiéndose observar que el 95% de las clases posee menos de 3 dependencias de otras clases. Además el 33% de las clases no poseen acoplamiento con otras y el 44% posee índices aceptables en cuanto a Acoplamiento. Así mismo los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización se comportan satisfactoriamente en un 77% de las clases. A manera de resumen se han tabulado los resultados obtenidos en la siguiente tabla:

Criterio	Evaluación
<=0	M
0 < a >= 0.5	R
0.5 < a >= 1	B

Tabla 10. Umbrales

Atributos	TOC	RC	Total
Complejidad de Implementación	1	-	1
Reutilización	1	1	1
Acoplamiento		1	1
Complejidad Mantenimiento		1	1

Cantidad de Pruebas		0.5	0.5
Responsabilidad	1	-	1
Total			**1**

Tabla 11. Resumen de los resultados

3.5 CONCLUSIONES PARCIALES

En el desarrollo de este capítulo se analizaron diferentes aspectos como el significado de las pruebas de software, sus objetivos y alcance. Esta fase añade valor al producto, todos los programas poseen errores y la fase de pruebas los descubre. Se realizó una descripción de los test de unidad, dentro de los cuales se analizó la prueba de Caja Blanca. Se determinó que el diseño de la implementación estuvo correcto, posibilitando una futura reutilización del código debido al bajo acoplamiento y la alta cohesión, tal como fue mostrado por las métricas utilizadas.

CONCLUSIONES GENERALES

A manera de conclusión se puede expresar que el estudio realizado alrededor del proceso de la Gestión de Almacenes en las entidades arroja que este es de gran importancia a la hora de tomar decisiones vitales por los directivos empresariales. Esta complejidad ha propiciado el surgimiento de sistemas informáticos que simplifiquen el problema, sin embargo, no abarcan las completas necesidades de las empresas de nuestro país, ni son desarrollados sobre tecnología libre.

Se detallaron las herramientas designadas para la implementación de los componentes. Para identificar e implementar las necesidades de integración con otros componentes dentro del subsistema Logística, se realizó un estudio de aquellos que podían ser reutilizados, lo que posibilitó acceder a funcionalidades que hubieran sido necesarias implementar, lo que hubiese provocado un esfuerzo mucho mayor para lograr la solución.

Se logró dar cumplimiento al objetivo general al obtener una aplicación funcional que viabiliza los procesos de Configuración y Recuperaciones del Subsistema Inventario, dentro del entrono empresarial cubano, definiendo para ello estándares de implementación que posibilitaron la excelente comunicación entre implementadores.

Para darle cumplimiento al último objetivo específico se aplicaron las pruebas del camino básico y se realizó una validación del diseño mediante la utilización de instrumentos de medición que se inspiraron en métricas para la calidad del diseño. Los resultados arrojados permitieron concluir que el diseño presentaba valores positivos en indicadores de calidad tales como Reutilización, Facilidad de Mantenimiento, Complejidad del Diseño, Complejidad de Implementación, Cohesión, Acoplamiento, Cantidad de pruebas entre otros. Esto apoya la afirmación de que el diseño desarrollado se puede considerar como aceptable, y el código puede ser reutilizado en futuras implementaciones.

RECOMENDACIONES

Con la culminación de este trabajo recomendamos:

- Solucionar las No Conformidades de las pruebas pilotos con el objetivo de refinar la solución.
- Implementar futuras funcionalidades que puedan ser propuestas por los analistas.

BIBLIOGRAFÍA

Adpime. 2007. [En línea] 21 de mayo de 2007. [Citado el: 2 de febrero de 2009.] http://www.adpime.com/ERP/Es_ERP_intro.html.

Autores, Colectivo de. 2007. *Introducción al proceso de desarrollo de software..* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2007.

Batista Chillón, Yojandy y Thompson Martínez, Rogfel. 2008. *Diagnóstico de vulnerabilidades de PC Vía Web.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

Booch, G. 1991. *Object-Oriented Design with Applications. The Benjamin/Cummings Publishing Company.* 1991.

Brown, W. 1998. *AntiPatterns: refactoring software, architectures, and projects in crisis.* 1998.

Buschmann, F. 1996. *Pattern-Oriented Software Architecture.* 1996.

Canos, José H y Penades, Carmen. 2003. Metodologías Agiles en el Desarrollo de Software. [En línea] 2003. [Citado el: 21 de febrero de 2009.] http://issi.dsic.upv.es/tallerma/actas.pdf 2003.

Cockburn, A. 1997. *Using Goal-Based Use Cases.* 1997. Vol. 7.

Company, Visual Paradigm international. 2007. Build Quality Applications Faster, Better and Cheaper. [En línea] 2007. [Citado el: 30 de enero de 2009.] http://www.visual-paradigm.com/.

Condor. 2005. El Sistema de Gestión con el que su empresa ganará en competitividad. [En línea] 9 de junio de 2005. [Citado el: 22 de marzo de 2009.] http://www.open-sol.com.ar/spa/productos/CondorEnterprise.pdf.

Davis, A. 1995. *Principles of Software Development.* [Digital] 1995.

del Toro Ríos, José Carlos y González Brito, Henry Raúl. 2008. *Documento Visión. Proyecto ERP-Cuba.* [Digital] La Habana : Universidad de las Ciencias Informáticas, 2008.

Desoft. 2008. El VERSAT-Sarasola: Sistema cubano de Gestión Contable-Financiero. [En línea] 7 de agosto de 2008. [Citado el: 21 de mayo de 2009.] http://www.vcl.desoft.cu/smf/index.php?topic=36.0.

Echarte, Patxi. 2007. Frameworks de Zend para el desarrollo de aplicaciones PHP. [En línea] 3 de julio de 2007. [Citado el: 11 de enero de 2009.] http://www.eslomas.com/index.php/archives/2007/07/03/framework-de-zend-para-el-desarrollo-de-aplicaciones-php/.

Expósito Álvarez, Alejandro y Escobar Requejo, Yunieski. 2008. *Diseño e implementación de las capas de Negocio y Acceso a datos de los módulos Salidas Transitorias y Traslados Interpenal del SIGEP.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

ExtJS. 2007. Ext JS JavaScript Library. [En línea] 2007. [Citado el: 3 de mayo de 2009.] http://extjs.com/deploy/dev/docs.

Fernández Carballo, Leamny y Castellanos Rolo, Betxy. 2007. *Estrategia motodológica para el desarrollo de Software de gestión a Distancia basado en Programación Extrema.* Ciudad de la Habana : s.n., 2007.

Fresno, V.D. 2006. Representación Autocontenida de Documentos HTML: una propuesta basada en Combinaciones Heurísticas de Criterio. [En línea] 2006. [Citado el: 6 de mayo de 2009.] http://www.escet.urjc.es/~vfresno/phd_sp.html..

Garlan, M. 1996. *Software Architecture: Prespectives on an emerging discipline.* 1996.

Gedise. 2009. [En línea] 2009. [Citado el: 21 de enero de 2009.] http://www.gedise.es/amedia.asp.

Gestión, Centro de Soluciones de. 2008. *Modelo de desarrollo orientado a componentes.* [Digital] La Habana : Universidad de las Ciencias Informáticas, 2008.

—. 2008. *Normas y estándares de codificación.* [Digital] Ciudad de la Habana : Universidad de la Habana, 2008.

Gómez Oduardo, Yen, Leyva Báez, Iyugnis y Hernández Grave de Peralta, Beatriz. 2008. *Análisis y diseño del Componente del Sistema de Ingresos dentro del ERP Postal.* [Digital] La Habana : Universidad de las Ciencias Informáticas, 2008.

González Rodríguez, Victoria. 2003. *El impacto de un ERP en la empresa.* [Digital] 2003.

González, H. 2006. *ERP cubano, un paso estratégico para la consolidación del Software Libre en Cuba.* 2006. Vol. 1.

Harbinter. 2005. [En línea] 2005. [Citado el: 26 de diciembre de 2008.] http://harbinter-soluciones.com/pagina.php?seccion=inicio.

Hernández Orallo, Enrique. 2002. El Lenguaje unificado de Modelado (UML). [En línea] 17 de diciembre de 2002. http://www.acta.es/articulos_mf/26067.pdf.

Hernández Reyes, Carlos Enrique y Leyva Durán, Julio Alberto. 2008. *Desarrollo de la Arquitectura del Sistema Automatizado de Control de Gestión de Indicadores de Refinación, SACGIR.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

IEEE. 2004. *Guide to the software engineering body of knowledge SWEBOK. A project of the IEEE Computer.* USA : s.n., 2004. s.n..

—. 1993. *Standards Collection: Software Engineering.* 1993. IEEE Standard 610.12-1990.

Küng, Stefan, Onken, Lübbe y Large, Simon. 2009. TortoiseSVN. Un cliente de Subversion para Windows. [En línea] 9 de mayo de 2009. [Citado el: 7 de junio de 2009.] http://hivelocity.dl.sourceforge.net/sourceforge/tortoisesvn/TortoiseSVN-1.6.2-es.pdf.

Larman, Graig. 2004. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* s.l. : Prentice Hall, 2004.

Márquez Alpízar, Yaimí y Valdés Echavarría, Yenni. 2008. *Procedimiento general de pruebas de Caja Blanca aplicando la técnica del Camino Básico.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

Montesino Afonso, Yasmany y Parra Lubín, Orestes. 2008. *Implementación del submódulo de Recuperaciones de Información del Módulo de Recuperaciones Dinámicas.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

Myers, Glen. 1979. *The Art of Software Testing.* 1979. ISBN-10: 0471078786.

Perera Morales, José Raúl. 2007. *Arquitectura de Software para Sistema Gestion de Inventario.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2007.

Pérez Armas, Armando Javier y Martínez Borges, Andy Vidal. 2008. *Subsistema de Configuración del Generador de Recuperaciones Dinámicas para aplicaciones Web.* [Digital] La Habana : Universidad de las Ciencias Informáticas, 2008.

Pressman, R. S. 1998. *Ingeniería de software. Un enfoque práctico.* 1998.

Rivas, Lornel A., y otros. 2007. *Herramientas de Desarrollo de Software: Hacia la construcción de una Ontología.* [Digital] 31 de marzo de 2007.

Rodas XXI. 2009. Rodas XXI: un producto cubano para la empresa cubana. [En línea] 9 de enero de 2009. [Citado el: 22 de mayo de 2009.] http://www.rodasxxi.cu/inventario.php.

Rosabal Carrión, Yarisbel y Palacios Martínez, Orlando. 2006. *Sistema de Contabilidad Material para la Actividad Presupuestada en las Fuerzas Armadas Revolucionarias. Módulo de entrega de medios materiales.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2006.

Santos Lebeque, Adriana y Hernández Valladares, Danaisy. 2008. *Estrategia de Pruebas para Juegos de Realidad Virtual.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

SAP España. 2007. SAP: número uno en software ERP. [En línea] 2007. [Citado el: 11 de febrero de 2009.] http://www.sap.com/spain/solutions/business-suite/erp/index.epx.

Shaw, D. 1994. *An introduction to software architecture. CMU Software Engineering Institute Technical Report.* 1994.

SQLManager. 2006. [En línea] 2006. [Citado el: 15 de abril de 2009.] http://sqlmanager.net/.

Stapleton, J. 2003. *DSDM Bussines Focused Development. DSDM Consortium.* 2003.

Stevens, W y Constantine, L. 1974. *Structured Design IBM System Journal.* 1974.

Torres Saquipova, Dina Yaksilik y de la Rosa Hernández, Carlos. 2008. *Análisis y Diseño de los módulos Inventario y Administración del proyecto ERP Cubano.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2008.

Ubunto. 2008. PgAdmin III. [En línea] marzo de 2008. [Citado el: 18 de abril de 2009.] http://www.guia-ubuntu.org/index.php?title=PgAdmin_III.

USC. 2002. Edición, compilación y ejecución de programas. [En línea] 22 de enero de 2002. [Citado el: 6 de marzo de 2009.] https://www.gsi.dec.usc.es/~alberto/fdp/practicas/SunForte/ForteSun.pdf.

Visual Paradigm. 2006. [En línea] 2006. [Citado el: 25 de febrero de 2009.] http://www.visual-paradigm.com/product/vpuml/.

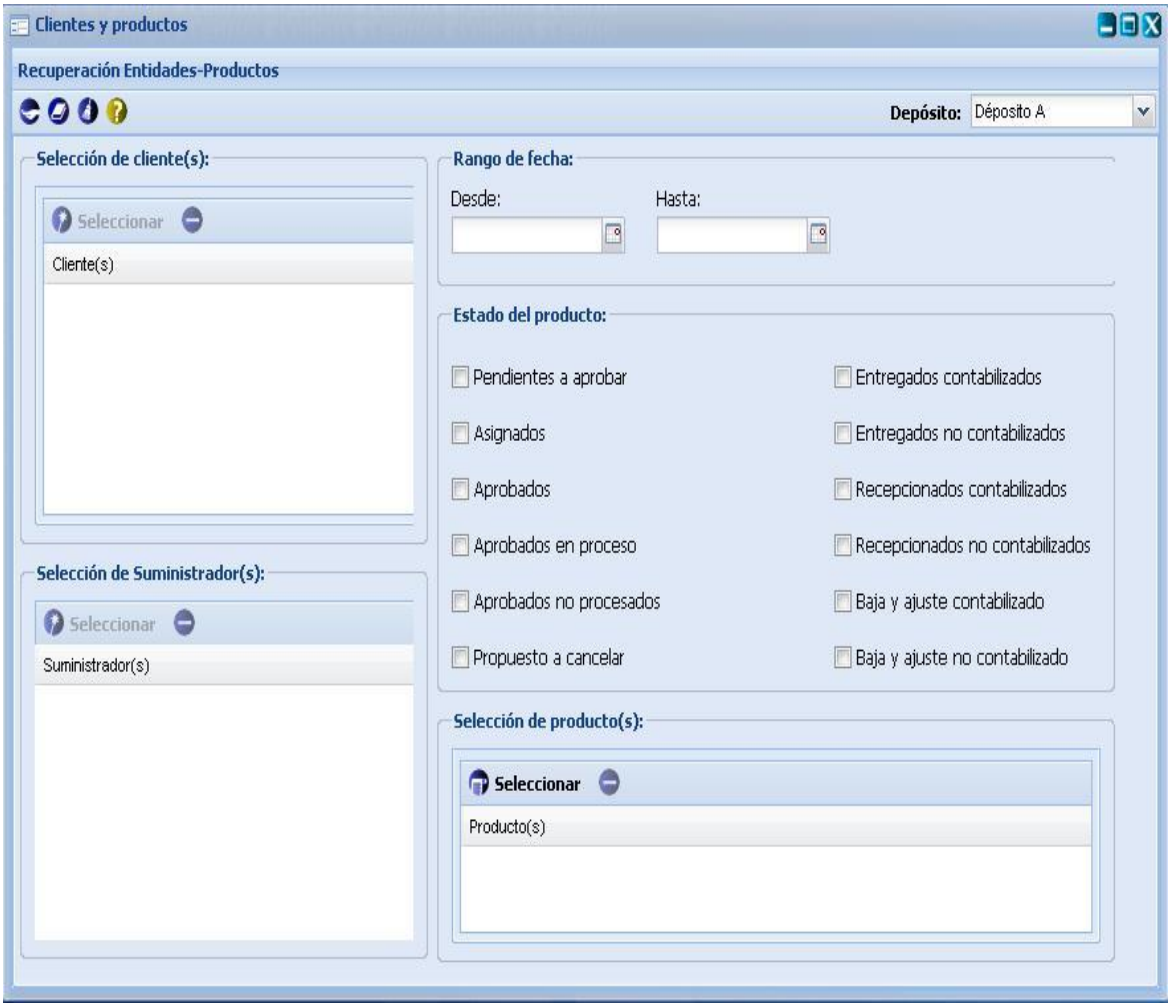
Wailgum, Thomas. 2008. ABC: An introduction to ERP. [En línea] 2008. [Citado el: 1 de marzo de 2009.] http://www.cio.com/research/erp/edit/erpbasics.html.

Yzquierdo Herrera, Raykenler y Lazo Ochoa, René. 2007. *El modelo de diseño del sistema HyperWeb. Módulos de Tratamiento Farmacológico y Configuración.* [Digital] Ciudad de la Habana : Universidad de las Ciencias Informáticas, 2007.

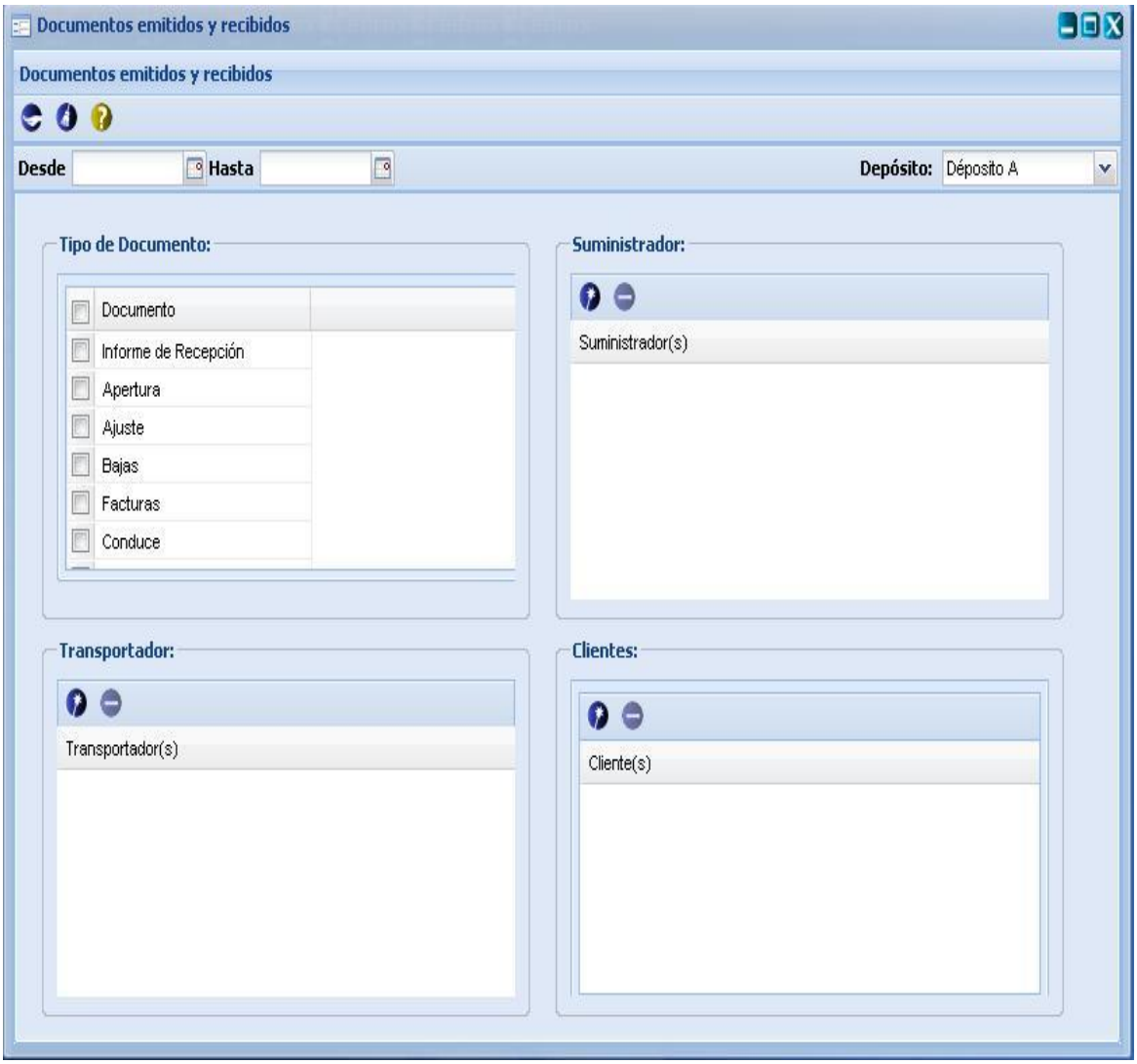
Zend. 2007. Zend Studio for Eclipse. [En línea] 2007. [Citado el: 3 de mayo de 2009.] http://www.zend.com/products/studio/.

ANEXOS

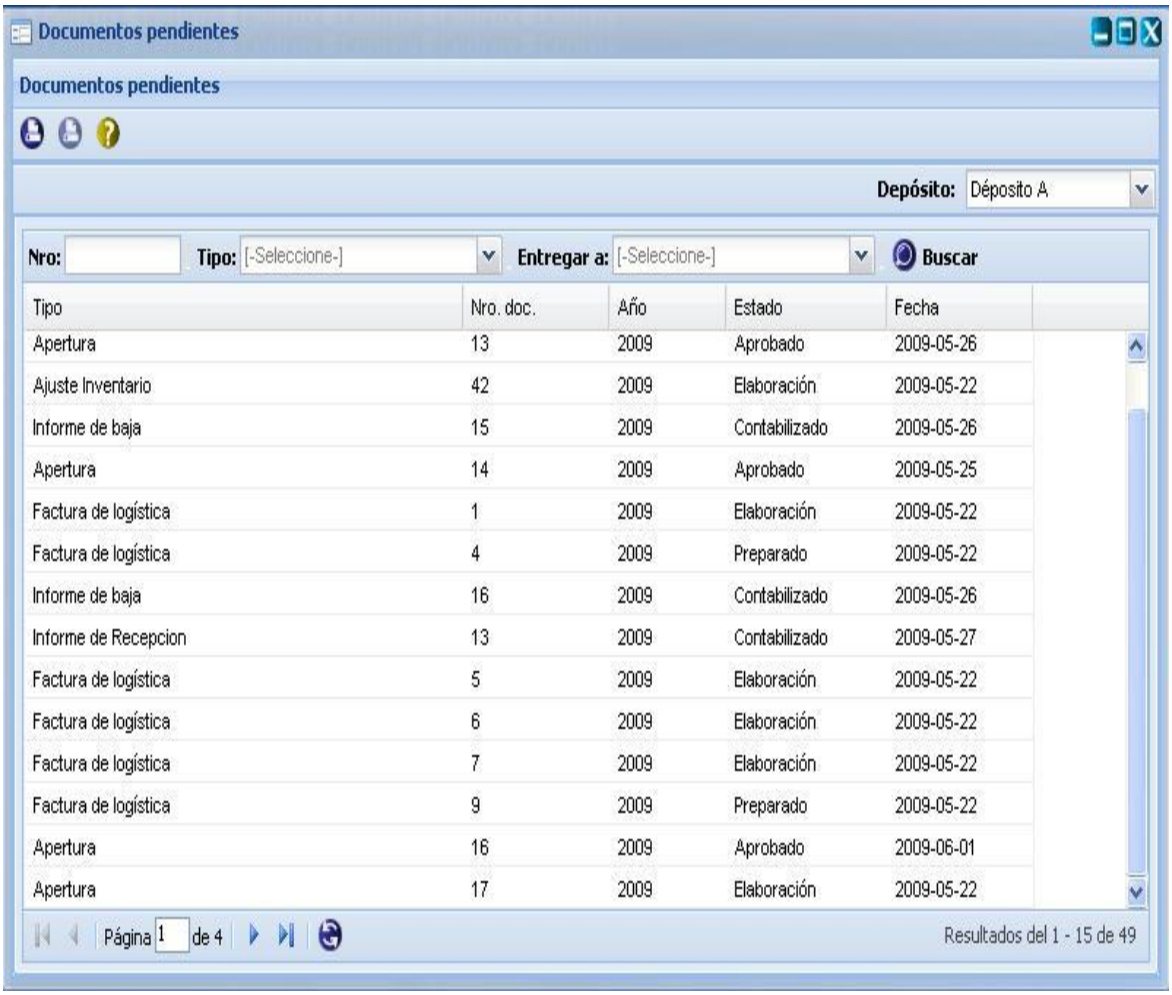
Anexo 1. Interfaz de la Recuperación Clientes -Productos



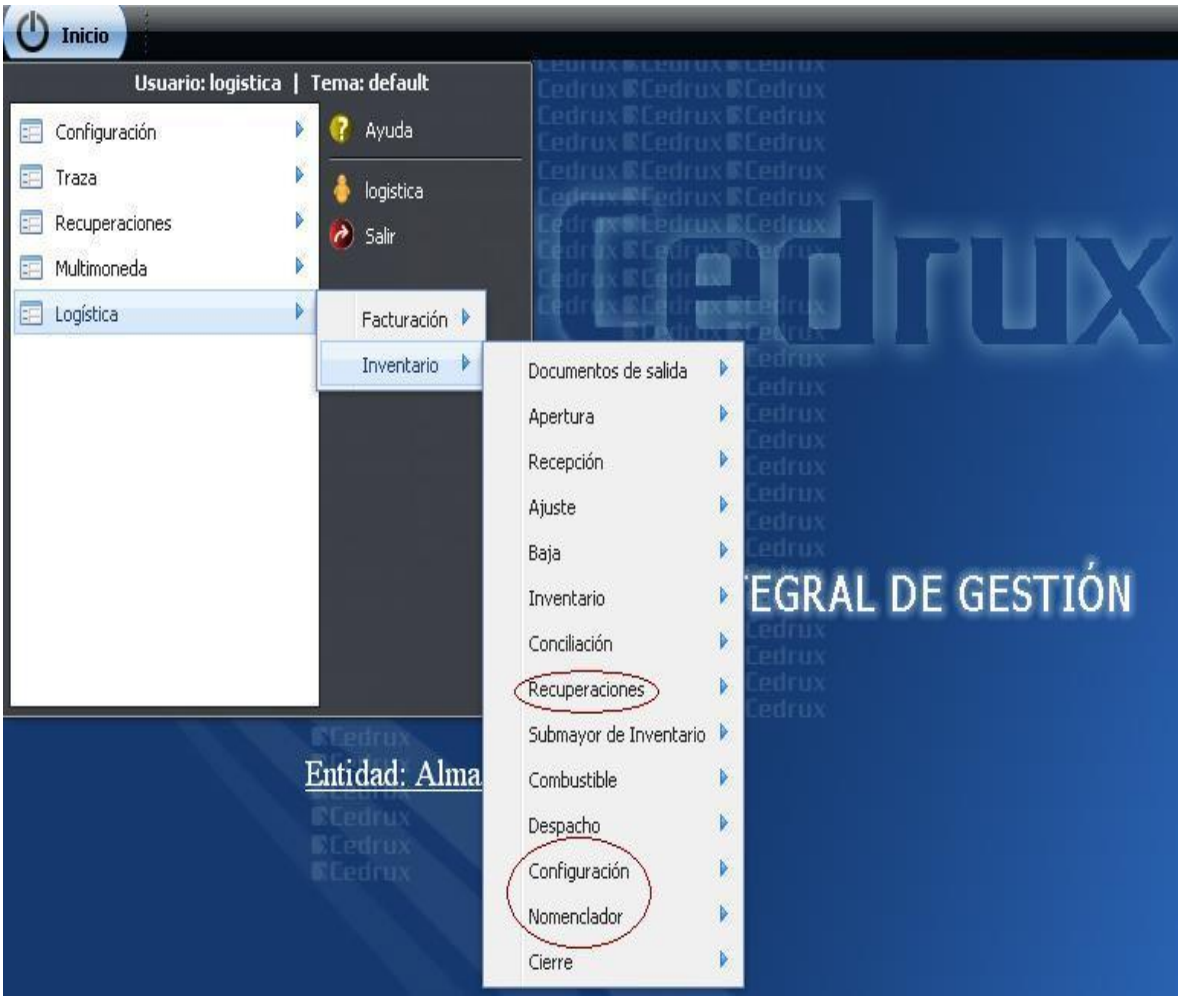
Anexo 2. Interfaz de la Recuperación Documentos emitidos y recibidos



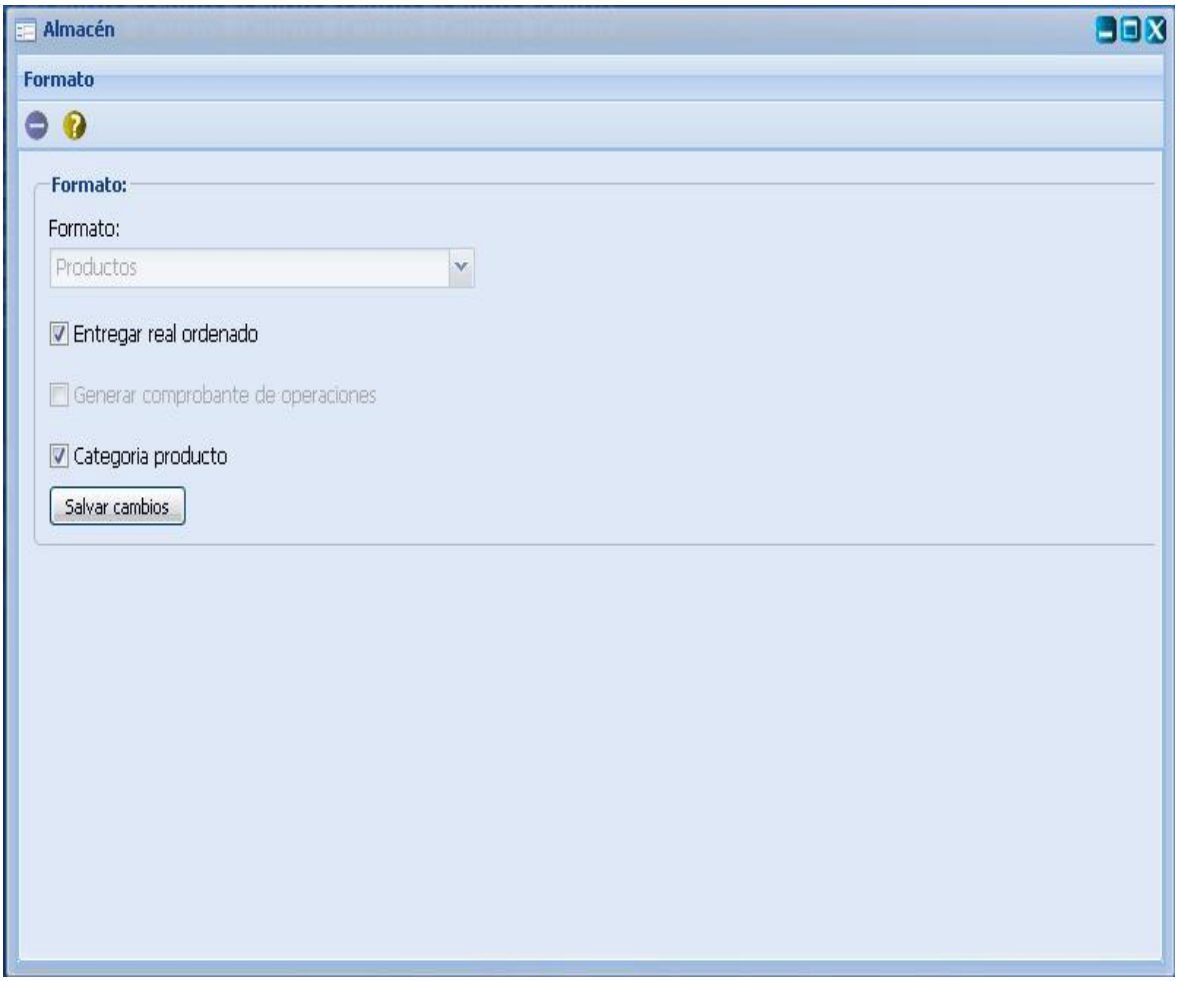
Anexo 3. Interfaz de la Recuperación Documentos pendientes



Anexo 4. Interfaz para el acceso a los componentes

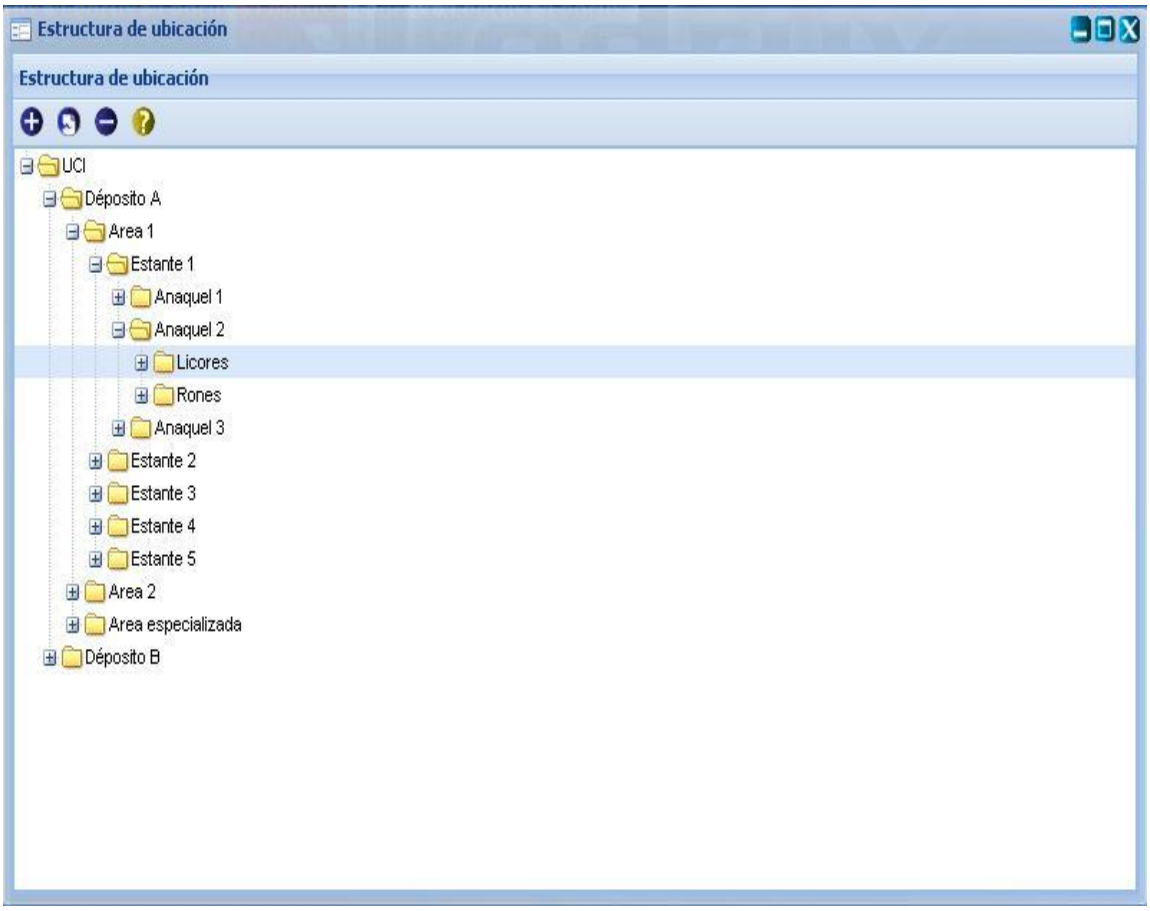


Anexo 5. Gestionar formato del almacén

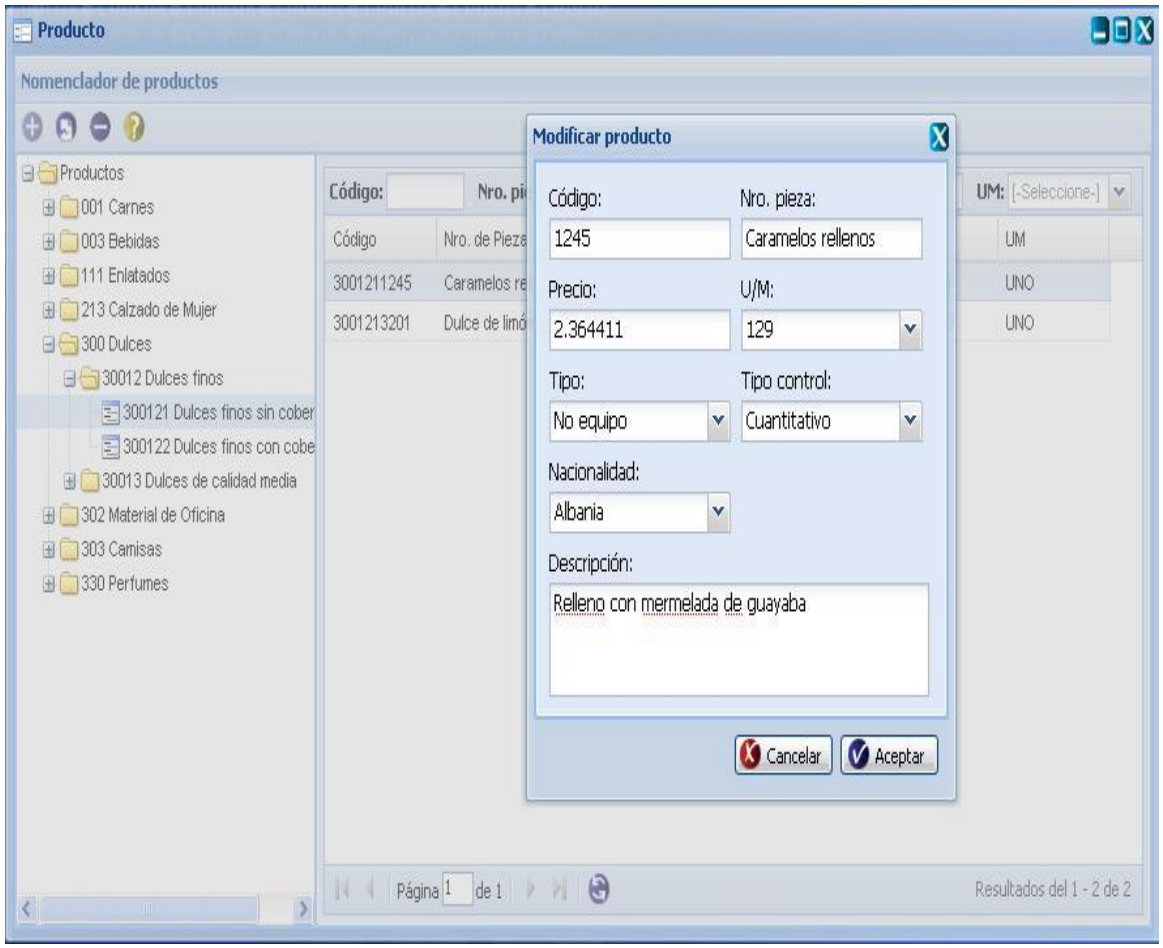


.

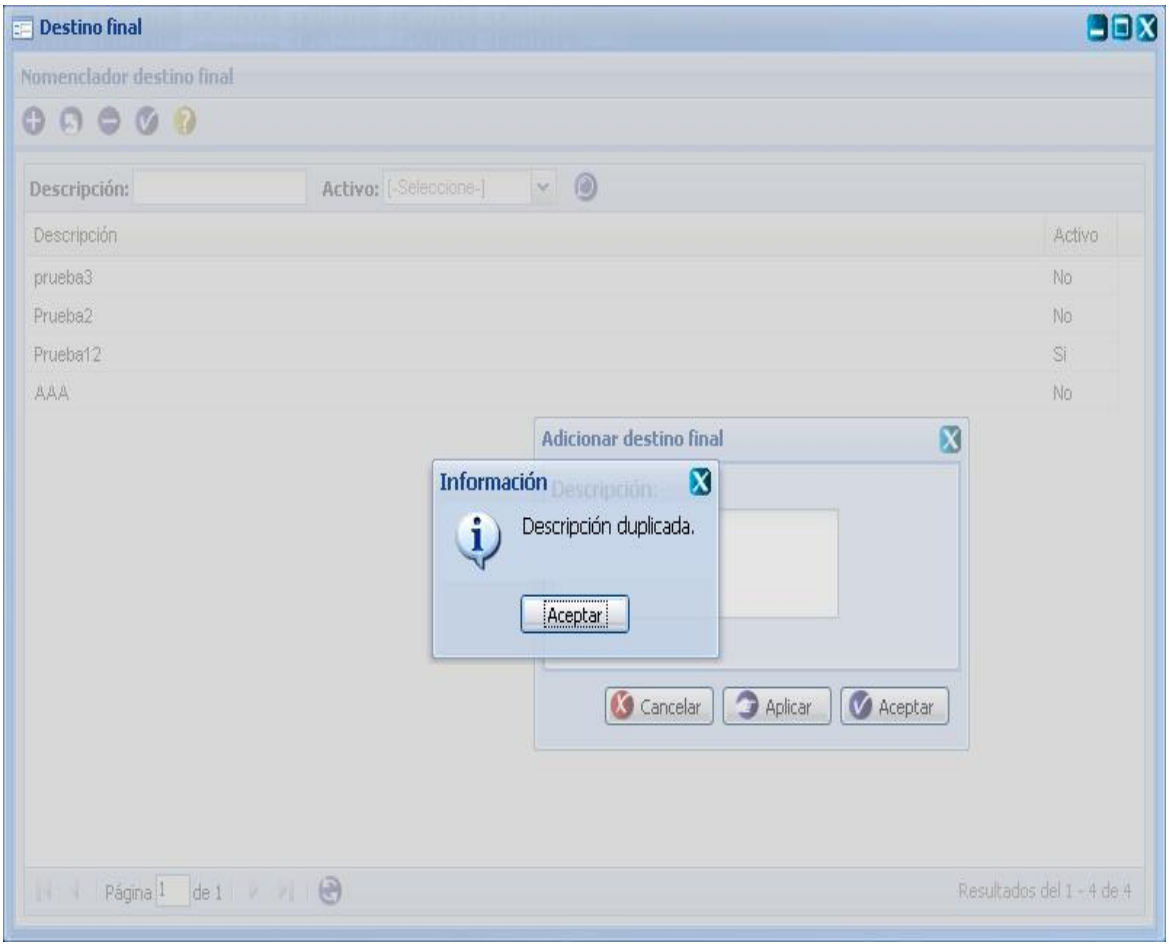
Anexo 6. Estructura de ubicación dentro del almacén



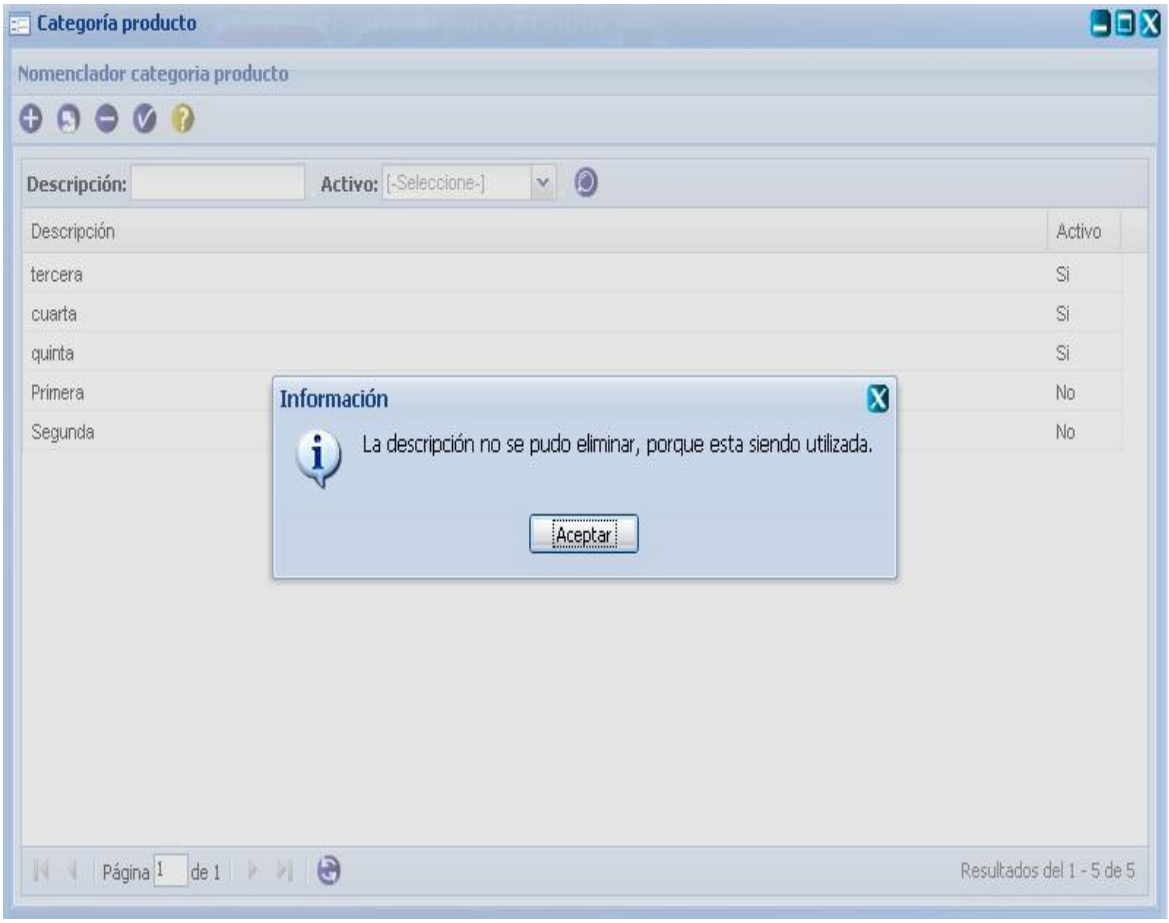
Anexo 7. Modificar nomenclador de producto



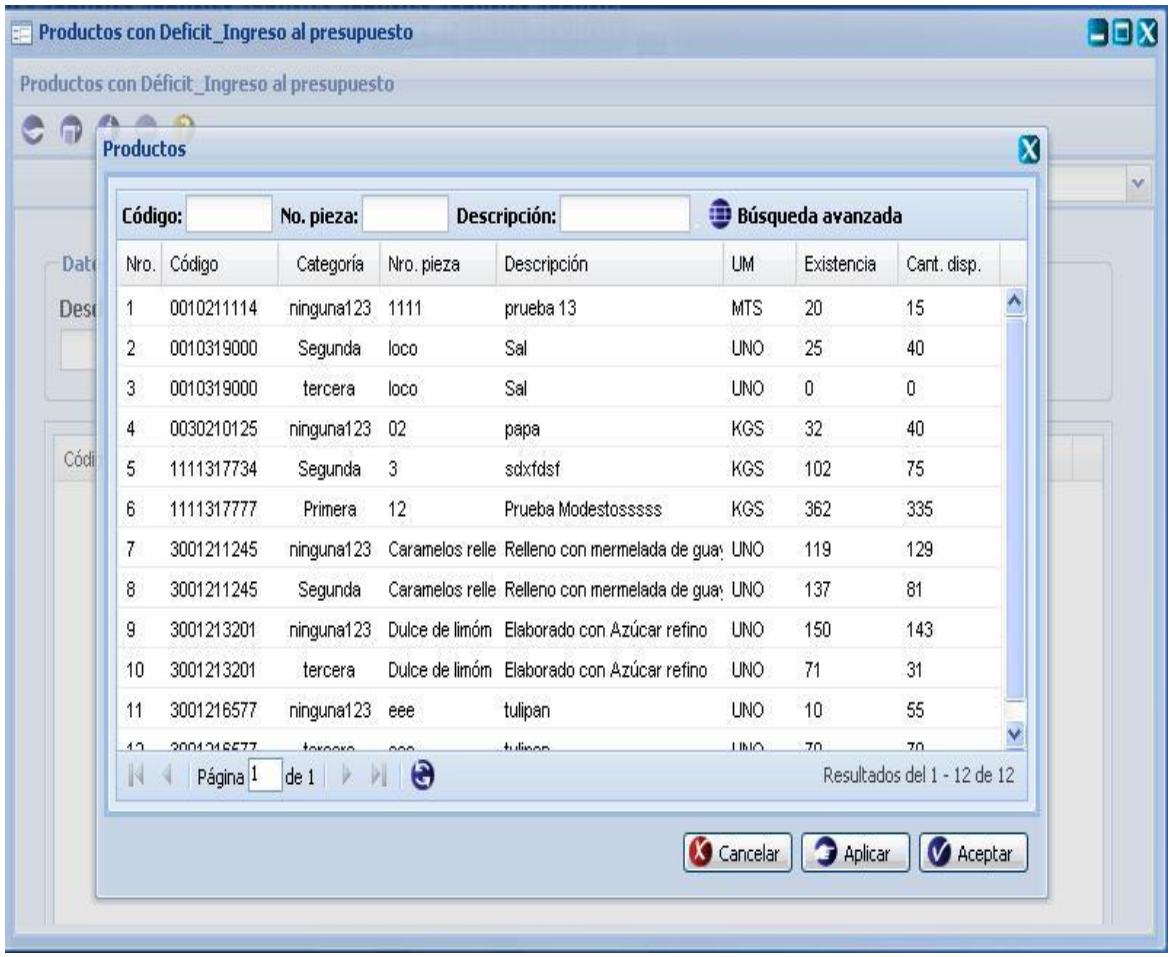
Anexo 8. Tratamiento de errores. Componente nomencladores



Anexo 9. Tratamiento de errores. Componente nomencladores



Anexo 10. Acceso al componente Producto para la selección de los mismos



Anexo 11. Instrumento de medición de la métrica Tamaño operacional de clase (TOC)

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

Tabla 12. Rango de valores de para la evaluación técnica de los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización) relacionados con la métrica TOC.

No	Componente	Clase	Procedimientos	Responsabilidad	Complejidad	Reutilización
1	Configuración	GestcfgalmacenController	7	Baja	Baja	Alta
2	Configuración	GestconfalmacenController	3	Baja	Baja	Alta
3	Configuración	GestdesbloqueodocController	4	Baja	Baja	Alta
4	Configuración	GestestructubicacionController	14	Media	Media	Media
5	Configuración	GestcfgalmacenModel	5	Baja	Baja	Alta
6	Configuración	GestconfalmacenModel	2	Baja	Baja	Alta
7	Configuración	GestestructubicacionModel	9	Media	Media	Media
8	Configuración	CfgAlmacen	5	Baja	Baja	Alta
9	Configuración	DatArea	6	Baja	Baja	Alta
10	Configuración	DatEstructuraubicacion	16	Media	Media	Media
11	Configuración	NomUmedida	8	Baja	Baja	Alta
12	Nomencladores	GestnomcalidadmovdiferenciaController	6	Baja	Baja	Alta
13	Nomencladores	GestnomcategoriaController	6	Baja	Baja	Alta
14	Nomencladores	GestnomconceptoplanController	6	Baja	Baja	Alta
15	Nomencladores	GestnomdestfinalController	6	Baja	Baja	Alta
16	Nomencladores	GestnomgrupoproductoController	6	Baja	Baja	Alta
17	Nomencladores	GestnomproductoController	18	Alta	Alta	Baja
18	Nomencladores	GestConceptoModel	4	Baja	Baja	Alta
19	Nomencladores	GestDestfinalModel	5	Baja	Baja	Alta

20	Nomencladores	GestNomCalidadmovdiferenciaModel	5	Baja	Baja	Alta
21	Nomencladores	GestNomCategoriaModel	7	Baja	Baja	Alta
22	Nomencladores	GestNomGrupoProductoModel	5	Baja	Baja	Alta
23	Nomencladores	GestnomproductoModel	11	Media	Media	Media
24	Nomencladores	NomCalidadmovdiferencia	9	Media	Media	Media
25	Nomencladores	NomCategoriaproducto	12	Media	Media	Media
26	Nomencladores	NomConceptoplan	9	Media	Media	Media
27	Nomencladores	NomDestinofinal	10	Media	Media	Media
28	Nomencladores	NomGrupoproducto	10	Media	Media	Media
29	Nomencladores	NomProducto	24	Alta	Alta	Baja
30	Recuperaciones	GestdocemitrecController	8	Baja	Baja	Alta
31	Recuperaciones	GestdocpendientesController	9	Media	Media	Media
32	Recuperaciones	GestentprodController	6	Baja	Baja	Alta
33	Recuperaciones	GestproddefingresoController	6	Baja	Baja	Alta
34	Recuperaciones	GestprodendocController	9	Media	Media	Media
35	Recuperaciones	gestvalormovController	3	Baja	Baja	Alta
36	Recuperaciones	GestdocemitrecModel	2	Baja	Baja	Alta
37	Recuperaciones	GestdocpendientesModel	4	Baja	Baja	Alta
38	Recuperaciones	GestentprodModel	1	Baja	Baja	Alta
39	Recuperaciones	GestprodendocModel	4	Baja	Baja	Alta
40	Recuperaciones	OlereadModel	6	Baja	Baja	Alta
41	Recuperaciones	ReaderModel	16	Media	Media	Media
42	Recuperaciones	RecupevalormovimientosModel	2	Baja	Baja	Alta
43	Recuperaciones	RecupvalormovModel	2	Baja	Baja	Alta

Tabla 13. Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad (Responsabilidad, Complejidad de Implementación y Reutilización)

Anexo 12 Instrumento de medición de la métrica Relaciones entre clases (RC)

Atributo	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad Mant.	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

Tabla 14. Rango de valores de para la evaluación técnica de los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas) relacionados con la métrica RC

No	Componente	Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad	Reutilización	Cantidad de Pruebas
1	Configuración	GestcfgalmacenController	2	Medio	Media	Media	Media
2	Configuración	GestconfalmacenController	0	Ninguno	Baja	Alta	Baja
3	Configuración	GestdesbloqueodocController	1	Bajo	Baja	Alta	Baja
4	Configuración	GestestructubicacionController	4	Alto	Alta	Baja	Alta
5	Configuración	GestcfgalmacenModel	1	Bajo	Baja	Alta	Baja
6	Configuración	GestconfalmacenModel	1	Bajo	Baja	Alta	Baja
7	Configuración	GestestructubicacionModel	1	Bajo	Baja	Alta	Baja
8	Configuración	CfgAlmacen	0	Ninguno	Baja	Alta	Baja
9	Configuración	DatArea	0	Ninguno	Baja	Alta	Baja
10	Configuración	DatEstructuraubicacion	0	Ninguno	Baja	Alta	Baja
11	Configuración	NomUmedida	0	Ninguno	Baja	Alta	Baja
12	Nomencladores	GestnomcalidadmovdiferenciaController	2	Medio	Media	Media	Media

13	Nomencladores	Gestnomcategoria Controller	2	Medio	Media	Media	Media
14	Nomencladores	Gestnomconceptop lanController	2	Medio	Media	Media	Media
15	Nomencladores	GestnomdestfinalC ontroller	2	Medio	Media	Media	Media
16	Nomencladores	Gestnomgrupoprod uctoController	2	Medio	Media	Media	Media
17	Nomencladores	GestnomproductoC ontroller	2	Medio	Media	Media	Media
18	Nomencladores	GestConceptoMod el	1	Bajo	Baja	Alta	Baja
19	Nomencladores	GestDestfinalModel	1	Bajo	Baja	Alta	Baja
20	Nomencladores	GestNomCalidadm ovdiferenciaModel	1	Bajo	Baja	Alta	Baja
21	Nomencladores	GestNomCategoria Model	1	Bajo	Baja	Alta	Baja
22	Nomencladores	GestNomGrupoPro ductoModel	1	Bajo	Baja	Alta	Baja
23	Nomencladores	GestnomproductoM odel	1	Bajo	Baja	Alta	Baja
24	Nomencladores	NomCalidadmovdif erencia	0	Ninguno	Baja	Alta	Baja
25	Nomencladores	NomCategoriaprod ucto	0	Ninguno	Baja	Alta	Baja
26	Nomencladores	NomConceptoplan	0	Ninguno	Baja	Alta	Baja
27	Nomencladores	NomDestinofinal	0	Ninguno	Baja	Alta	Baja
28	Nomencladores	NomGrupoproducto	0	Ninguno	Baja	Alta	Baja
29	Nomencladores	NomProducto	0	Ninguno	Baja	Alta	Baja
30	Recuperaciones	GestdocemitrecCon troller	1	Bajo	Baja	Alta	Baja
31	Recuperaciones	Gestdocpendientes Controller	1	Bajo	Baja	Alta	Baja
32	Recuperaciones	GestentprodControl ler	1	Bajo	Baja	Alta	Baja
33	Recuperaciones	Gestproddefingreso Controller	0	Ninguno	Baja	Alta	Baja
34	Recuperaciones	GestprodendocCon troller	1	Bajo	Baja	Alta	Baja
35	Recuperaciones	gestvalormovContr oller	1	Bajo	Baja	Alta	Baja
36	Recuperaciones	GestdocemitrecMo del	1	Bajo	Baja	Alta	Baja
37	Recuperaciones	Gestdocpendientes Model	1	Bajo	Baja	Alta	Baja
38	Recuperaciones	GestentprodModel	1	Bajo	Baja	Alta	Baja
39	Recuperaciones	GestprodendocMod el	1	Bajo	Baja	Alta	Baja
40	Recuperaciones	OlereadModel	0	Ninguno	Baja	Alta	Baja
41	Recuperaciones	ReaderModel	0	Ninguno	Baja	Alta	Baja

42	Recuperaciones	RecupevalormovimientosModel	3	Alto	Media	Media	Media
43	Recuperaciones	RecupvalormovModel	3	Alto	Media	Media	Media

Tabla 15. Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad (Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas)

GLOSARIO DE TÉRMINOS

1. **Algoritmo**

   Es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema.

2. **Base de Datos**

   Es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

3. **Componentes**

   El componente es la unidad de construcción elemental del diseño físico. Las características de un componente son:

   - Se define según cómo interactúa con otros
   - Encapsula sus funciones y sus datos
   - Es reusable a través de las aplicaciones
   - Puede verse como una caja negra
   - Puede contener otros componentes

4. **Fichero**

   Es una manera particular de codificar información para almacenarla en un archivo informático.

5. **Internet Explorer**

   Es un navegador web producido por Microsoft para el sistema operativo Windows desde 1995 y más tarde para Sun Solaris y Apple Macintosh, estas dos últimas discontinuadas en el 2002 y 2006 respectivamente. Ha sido el navegador web más utilizado desde 1999.

6. **Implementación**

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

7. **Inventario**

Por **inventario** se define al registro documental de los bienes y demás cosas pertenecientes a una persona o comunidad, hecho con orden y precisión.

En el campo de la gestión empresarial, un inventario registra el conjunto de todos los bienes propios y disponibles para la venta a los clientes, considerados como activo corriente.

8. **Logística**

Conjunto de medios y métodos necesarios para llevar a cabo la organización de una empresa, o de un servicio.

9. **Métricas**

Las métricas son un buen medio para entender, monitorizar, controlar, predecir y probar el desarrollo software y los proyectos de mantenimiento.

10. **Módulo**

A efectos prácticos, hablar de *programas* es lo mismo que hablar de *productos de software* o hablar de *módulos de software*. Cada *módulo* es una parte del sistema, que se instala y funciona por separado, entrelazándose con otros módulos con los que intercambia información.

11. **Nomencladores**

Tabla de valores definidos por el usuario, que luego no pueden ser modificados.

12. **Objeto**

Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad. Corresponden a los objetos reales del mundo que nos rodea, o a objetos internos del sistema.

13. **Plataformas**

Entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones.

14. Seguridad

La **seguridad informática** consiste en asegurar que los recursos del sistema de información (material informático o programas) de una organización sean utilizados de la manera que se decidió y que el acceso a la información allí contenida así como su modificación sólo sea posible a las personas que se encuentren acreditadas y dentro de los límites de su autorización.

15. SGBD

Programas que permiten almacenar y posteriormente acceder a los datos de forma rápida y estructurada.

16. Software

Se refiere a los programas y datos almacenados en un ordenador.

- Los programas dan instrucciones para realizar tareas al hardware o sirven de conexión con otro software.
- Los datos solamente existen para su uso eventual por un programa.

17. Software Libre

Es la denominación del software que brinda libertad a los usuarios sobre su producto adquirido y por tanto, una vez obtenido, puede ser usado, copiado, estudiado, modificado y redistribuido libremente.

18. SQL

El Lenguaje de consulta estructurado (Structured Query Lenguaje) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos,

así como también hacer cambios sobre la misma. Es un lenguaje de cuarta generación (4GL).

19. TCP/IP

Sistema de protocolos establece las bases de la mayor parte del universo de Internet. El TCP se encarga en fragmentar la información en pequeños paquetes para después volverlos a juntar en un destino final. El IP tiene como función el checar que estos paquetes vayan dirigidos correctamente hacia un mismo destino.